Fuzzy neural networks for time-series forecasting of electric load

P.K. Dash
G. Ramakrishna
A.C. Liew
S. Rahman

Indexing terms: Neural networks, Load prediction, Time-series forecasting, Supervised learning, Unsupervised learning

Abstract: Three computing models, based on the multilayer perceptron and capable of fuzzy classification of patterns, are presented. The first type of fuzzy neural network uses the membership values of the linguistic properties of the past load and weather parameters and the output of the network is defined as fuzzy-class-membership values of the forecast load. The backpropagation algorithm is used to train the network. The second and third types of fuzzy neural network are developed based on the fact that any fuzzy expert system can be represented in the form of a feedforward neural network. These two types of fuzzy-neural-network model can be trained to develop fuzzy-logic rules and find optimal input/output membership values. A hybrid learning algorithm consisting of unsupervised and supervised learning phases is used to train the two models. Extensive tests have been performed on two-years of utility data for generation of peak and average load profiles 24 hours and 168 hours ahead, and results for typical winter and summer months are given to confirm the effectiveness of the three models.

1 Introduction

The application of artificial-neural-network- (ANN) and fuzzy-logic-based decision-support systems to time-series forecasting has gained attention recently [7–12]. ANN-based load forecasts give large errors when the weather profile changes very fast. Also, extremely slow training or even training failure occurs in many cases owing to difficulties in selecting proper structures of the neural-network paradigm being used, and owing to the errors in associated parameters such as learning rates, activation functions etc. which are fundamental to any back-propagation neural network. On the other hand, the development of a fuzzy decision system (fuzzy expert system) for load forecasting requires detailed analysis of data and the fuzzy-rule base has to be developed heuristically for each season. The rules fixed in this way may not always yield the best forecast. The shortcomings of the neural-network paradigm can be partly remedied by the recognition of the fact that the learning speed and accuracy of an ANN may often be enhanced by utilising the knowledge of neural-network expertise in a specific application. This human knowledge can be encoded by fuzzy expert systems, which are integrated into the fuzzy neural network (FNN).

The present work is aimed at achieving a robust load forecast with much improved accuracy using three different models of FNNs. For the neural network to be called a FNN, the signal and/or the weights should be fuzzified [14]. The first type of FNN, abbreviated FNN_1, is based on the multilayer perceptron, using the backpropagation algorithm. The input vector consists of the membership values of linguistic properties of the past load and weather parameters and the output vector is defined in terms of fuzzy-class-membership values of the forecast load. The second and third types of FNN, abbreviated as FNN_2 and FNN_3, are based on the argument that any fuzzy expert system employing one block of rules may be approximated by a neural network (feedforward, multilayered). The input vector to FNN_2 and FNN_3 consists of differences in weather parameters between the present and forecast instant. The output of the FNN_2 and FNN_3 gives the load correction which, when added to the past load, gives the forecast load. The learning algorithm for FNN_2 and FNN_3 combines unsupervised and supervised learning procedures to build the rule nodes and train the membership functions. The supervised learning procedure for FNN_2 uses the gradient-descent backpropagation algorithm [5] for finding the optimum weights and membership functions, while for FNN_3 the supervised learning procedure comprises of Kalman-filter-based algorithm [16] which is similar to the least-square adaptive techniques. The least-square adaptive filtering techniques are known to have rapid convergence properties over the backpropagation algorithm. In this paper FNNs using fuzzy weights are not considered.

A few examples of peak-load forecasting and average-load forecasting using the above techniques for a typical utility with 24-hour and 168-hour lead times in the months of winter and summer are presented, and results obtained using the two FNN models are compared.

© IEE, 1995

Paper 1807C (P10), first received 6th June and in revised form 30th November 1994

P.K. Dash and G. Ramakrishna are with the Department of Electrical Engineering, Regional Engineering College, Rourkela 769 008, India

A.C. Liew is with the Department of Electrical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 0511

S. Rahman is with the Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, USA

The authors acknowledge funds from the US National Science Foundation (NSF grant INT-9209103 and INT-9117624).

2 Type 1: fuzzy neural network (FNN_1) for load forecasting

The network model for FNN_1 is shown in Fig. 1. The FNN_1 model is similar to ANN with the input and

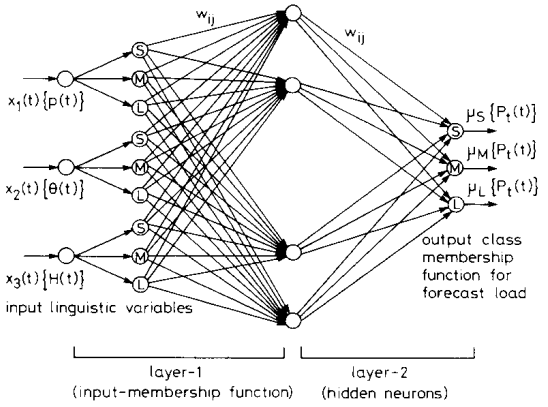

Fig. 1 *Type 1 fuzzy neural network (FNN_1) for load forecasting*
S = small, M = medium, L = large
P = load, Q = temperature, H = humidity, t = iteration, W_{ij} = weights

output parameters fuzzified. This is very important for load forecasting since there are so many fuzzy factors which are difficult to characterise by a number. An instance of this could be weather conditions such as temperature, humidity, cloud cover etc. The FNN_1 clusters the input parameters such as load of ith day, maximum and minimum temperatures and humidities of the ith day and $(i + n)$th day into fuzzy spaces for forecasting load on the $(i + n)$th day (n is the lead time for the load forecast, i.e. $n = 24$ for 24 h ahead forecast, $n = 168$ for 168 h ahead forecast etc.). The load, temperature and humidity are classified into three categories, i.e. small, medium and large. The output nodes of FNN_1 represents the class-membership function of the forecast load. The classification of input and output linguistic variables into fuzzy spaces involves an increase in the amount of computation required compared with the ANN. This is suitably offset by the fact that the conventional crisp backpropagation algorithm may not necessarily converge when the training patterns are nonseparable with overlapping fuzzy classes. Further, in the proposed FNN_1 the error back-propagated has more weight for nodes with higher membership values and hence induces greater weight corrections for that class. Thus the ambiguity in modelling the uncertain vectors is automatically reduced.

In many cases it is convenient to express the membership function of a fuzzy subset in terms of a standard nonlinear function. The Gaussian membership function is used for the input and output linguistic parameters of the FNNs in this study:

$$\mu(x; a, b) = \exp\left\{\frac{-(x - a)^2}{b}\right\} \quad (1)$$

Here, a and b are the centre and width of the Gaussian function, respectively.

Fig. 2 shows the membership functions for peak load, maximum temperature and maximum humidity in winter. The backpropagated error is computed with respect to the desired class-membership values for the output and each weight is updated using the gradient-descent backpropagation algorithm [5]. The input layer consists of nodes equal to the product of the input linguistic-pattern points and the fuzzy-term sets (i.e. three term sets for each pattern point). The output layer con-

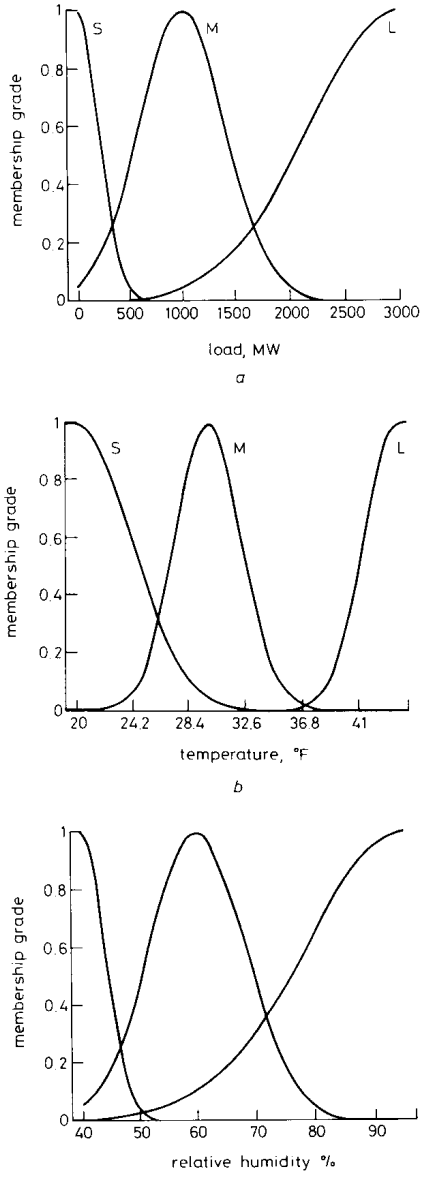

Fig. 2 *Peak-load, maximum-temperature and maximum-humidity membership functions using FNN_1 in winter*
S = small, M = medium, L = large
a Peak load
b Maximum temperature
c Maximum relative humidity

sists of three terms sets for the forecast load. The number of hidden nodes is fixed empirically during training.

After the training phase is over, the input consists of load, temperature and humidities of the ith and the forecast temperature and humidity values of the $(i + n)$th day. The output of FNN_1 gives the class-membership values of forecast load of the $(i + n)$th day. The final forecast load is obtained by using the centre-of-gravity defuzzification technique [13–15].

3 Type 2: fuzzy neural network (FNN_2) for load forecasting

An alternative to the neural-network-based load forecast is the expert-system approach. A fuzzy expert system for load forecasting consists of a collection of fuzzy IF–THEN rules showing the relationship between load and weather variables. One of the difficulties with the fuzzy expert system is the rule matching and composition time, apart from the time-consuming process of adapting the rules. The neural network eliminates the rule-matching process and stores the knowledge in the link weights. The decision signals can be output immediately after the input data are fed in. Fig. 3 shows the proposed

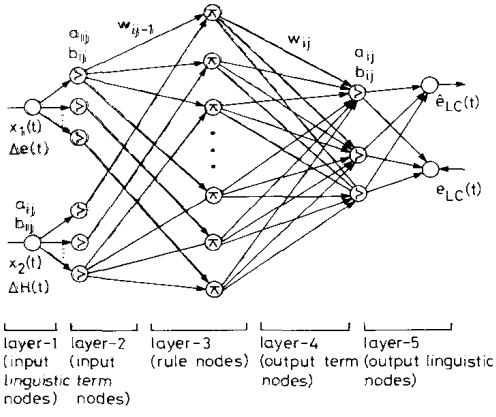

Fig. 3 *Type 2 and type 3 fuzzy neural network (FNN_2 and FNN_3) for load forecasting*

$\Delta\theta$ = differential temperature
ΔH = differential humidity
t = iteration number
W_{ij} = weights
$\hat{e}_{LC}$ = actual load correction
e_{LC} = desired load correction

FNN_2 to model the fuzzy expert system in the form of an FNN using the ANN architecture. The FNN_2 clusters the differential temperatures and humidities of the ith and $(i + n)$th day into fuzzy-term sets. The output of FNN_2 is the final crisp-load correction ($\hat{e}_{LC}$). Hence the forecast load on the $(i + n)$th day $[P_f(i + n)]$ is given by

$$P_f(i + n) = P(i) + \hat{e}_{LC}(i) \tag{2}$$

where n is the lead time for the forecast.

FNN_2 has a total of five layers. Nodes at layer one are the input linguistic nodes. Layer five is the output layer and consists of two nodes [one is for the actual load correction ($\hat{e}_{LC}$) and the other is the desired load correction (e_{LC})]. Nodes at layers two and four are term nodes which act as membership functions to represent the term sets of the respective linguistic variables. Each node at layer three represents the preconditions of the rule nodes, and layer-four links define the consequences of the rules. The functions of each layer is described as follows:

(*a*) layer 1: the nodes in this layer just transmit the input feature x_i, $i = 1, 2$ to the next layer;

(*b*) layer 2: each input feature x_i, $i = 1, 2$ is expressed in terms of membership values $\mu_x^j(a_{ij}, b_{ij})$, where i corresponds to the input feature and j corresponds to the number of term sets for the linguistic variable x_i. The membership function $\mu_{x_i}^j$ uses the Gaussian membership function given in eqn. 1;

(*c*) layer 3: the links in this layer are used to perform precondition matching of fuzzy-logic rules. Hence the rule nodes perform the product operation (or AND operation):

$$\mu_{R_P} = \Pi\mu_{x_i}^j \tag{3}$$

where $R_P = 1, 2, \ldots, n$. R_P corresponds to the rule node and n is the maximum number of rule nodes. However, if the fuzzy AND operation is used

$$\mu_{R_P} = \min\{\mu_{x_i}^j\} \tag{4}$$

(*d*) layer 4: the nodes in this layer have two operations, i.e. foreward and backward transmission. In forward-transmission mode, the nodes perform the fuzzy OR operation to integrate the fired rules which have the same consequence

$$\mu^4 = \sum_{i=1}^{P} u_i^4 \tag{5}$$

where p corresponds to the links terminating at the node. In the backward-transmission mode, the links function exactly the same as the layer-two nodes.

(*e*) layer 5: there are two nodes in this layer for obtaining the actual and desired output-load correction, respectively. The desired output-load correction (e_{LC}) is fed into the FNN_2 during learning whereas the actual load correction ($\hat{e}_{LC}$) is obtained by using the centroid defuzzification method.

3.1 Hybrid learning algorithm for FNN_2

The hybrid learning scheme consists of unsupervised- and supervised-learning phases. In the unsupervised phase, the initial membership functions of the input and output linguistic variables are fixed and an initial form of the network is constructed. Then, during the learning process, some nodes and links of this initial network are deleted or combined to form the final structure of the network. In the supervised-learning phase, the input and output membership functions are optimally adjusted to obtain the desired outputs.

3.1.1 Unsupervised-learning phase

Given the training input data $x_i(t)$, $i = 1, 2$, the desired output-load correction $e_{LC}(t)$ and the fuzzy partitions $|\mu_{x_i}^j|$, we wish to locate the membership functions (i.e. a_{ij} and b_{ij}) and find the fuzzy-logic rules.

Kohonen's feature-maps algorithm [13] is used to find the values for a_{ij} and b_{ij}:

$$\|x(t) - a_{i,\,closest}(t)\| = \min_{1 \le j \le t}\{\|x_i(t) - a_{ij}(t)\|\} \tag{6}$$

$$a_{i,\,closest}(t + 1) = a_{i,\,closest}(t) + \eta(t)\{x_i(t) - a_{i,\,closest}(t)\} \tag{7}$$

$$a_{ij}(t + 1) = a_{ij}(t) \quad \text{for } a_{ij} \neq a_{i,\,closest} \tag{8}$$

where $\eta(t)$ is the monotonically decreasing learning rate and t is the number of term sets for the linguistic variable x_i.

This adaptive formulation runs independently for each input linguistic variable x_i.

The width b_{ij} is determined heuristically at this stage [13] as follows:

$$b_{ij} = \frac{|a_{i,j} - a_{i,\,closest}|}{r} \tag{9}$$

where r is an overlap parameter. After the parameters of the membership functions have been found, the weights in layer four are obtained by using the competitive-learning algorithm [6] as follows:

$$W_{ij} = LI_j^4(LI_j^3 - W_{ij}) \tag{10}$$

where LI_j^3 serves as the win–loss index of the rule node at layer three and LI_j^4 serves as the win–loss index of the jth term node at layer four, respectively.

After the competitive learning through the whole training data set, the link weights at layer four represent the strength of the existence of the corresponding rule consequences. If a link weight between a rule node and the term node of the output linguistic node is very small, all the corresponding links are deleted, meaning that this rule node has little or no relation to the output.

Once the consequences of rule nodes have been determined, the rule combination is performed to reduce the number of rules in the following manner. The criterion for the choice of rule nodes is

(i) they have the same consequences;

(ii) some preconditions are common to all the rule nodes in this set; and

(iii) the union of other preconditions of these rule nodes composes the whole term set of some input linguistic variables.

The rule nodes which satisfy these criteria are replaced by a new rule node with common preconditions.

3.1.2 Supervised-learning phase

Once the fuzzy-logic rules have been found, supervised learning is used to find the optimum weights and the input and output membership functions by using the gradient-descent backpropagation algorithm. The detailed steps are given in Appendix 9.1.

The hybrid learning procedure is summarised in Fig. 4. The convergence speed of the supervisory-learning

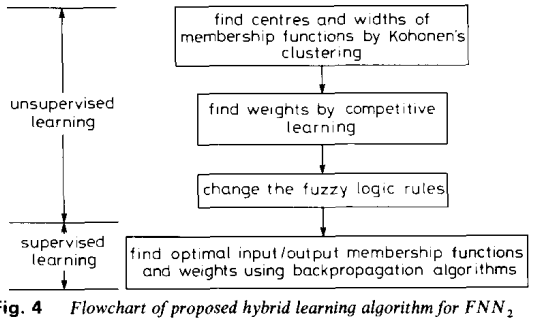

Fig. 4 *Flowchart of proposed hybrid learning algorithm for FNN_2*

scheme for FNN_2 is found to be superior to that of the supervisory-learning scheme for FNN_1, since the unsupervised-learning process for FNN_2 had carried out much of the learning process in advance. The convergence speed of the supervised-learning process can be further improved by solving the weight-update equations at layer three and the input- and output-membership functions at layers one and two by linear Kalman-filter equations [16].

4 Type 3: fuzzy neural network (FNN_3) for load forecasting

Referring to Fig. 1, the tuning of the Gaussian membership function at layers two and four (a_{ij}, b_{ij}) is similar to the weight-update equations at layer three. The unsupervised-learning phase of the FNN_3 model is the same as for the FNN_2 model. The supervised-learning phase of the FNN_3 model uses the linear Kalman-filter equations for updating the weights and the membership function. Unlike the backpropagation technique, this algorithm assumes that the estimated weight matrix is nonstationary and hence will allow the tracking of time-varying data such as those of load forecasting.

This algorithm defines locally a gradient based on present and past data at each node, and updates the weights of each node using the linear Kalman-filter equations so as to bring this gradient identically to zero whenever an update is made. Performing the update thus, and defining the gradient in this manner, ensures that maximum use is made of available information.

The gradient for the linear combiner at each node is defined as

$$\boldsymbol{G} = \boldsymbol{RW} - \boldsymbol{C} \tag{11}$$

Here R is the autocorrelation matrix for each layer and is calculated as

$$\boldsymbol{R} = \sum_{np=1}^{NP} ff^{NP-np} \boldsymbol{x}_{np} \boldsymbol{x}_{np}^T \tag{12}$$

and $\boldsymbol{C}$ is the crosscorrelation matrix and is given by

$$\boldsymbol{C} = \sum_{np=1}^{NP} ff^{NP-np} \boldsymbol{d}_{np} \boldsymbol{x}_{np}^T \tag{13}$$

where NP denotes the total number of patterns and ff denotes the forgetting factor. d_{np} and x_{np} are the summation output and the output of the nonlinearity (Gaussian membership function) for the layer-two and layer-five nodes, respectively. As the layer-four nodes contain no nonlinearity term, therefore $d_{np} = x_{np}$.

The weight vector which makes $\boldsymbol{G} = \boldsymbol{RW} - \boldsymbol{C}$ zero is the solution to the equations. The detailed weight-update algorithm using the linear Kalman filter is given in Appendix 9.2.

5 Implementation results

To evaluate the performance of the FNN models, load forecasting is performed on typical utility data. The models ANN, FNN_1, FNN_2 and FNN_3 are tested on two years of utility data for generating peak and average load profiles and some of the results are given in the subsequent subsections. In References 7–9 it has been shown that ANN gives the best prediction and accuracy compared with conventional approaches. Therefore in this paper the results of FNN_1, FNN_2 and FNN_3 are compared with those of the ANN approach.

The training sets are formed separately for each of the seven day types (i.e. Tuesdays through Thursdays, Mondays, Fridays, Saturdays, Sundays, holidays). The selection of training patterns is given in Reference 9.

5.1 Peak-load forecasting

For peak-load forecasting, the following training data are used for ANN and FNN_1:

Input pattern: $P_{max}(i)$, $\Theta_{max}(i)$, $H_{max}(i)$, $\Theta^f_{max}(i+n)$, $H^f_{max}(i+n)$

Output pattern: $P_{max}(i+n)$ and $\mu\{P_{max}(i+n)\}$ for ANN and FNN_1, respectively.

where P, Θ and H stand for load, temperature and humidity, respectively. The superscript f denotes the forecast values for Θ and H; n is the lead time for forecast ($n = 24$ for 24 h-ahead forecast, $n = 168$ for 168 h-ahead forecast).

The forecast value $P_{max}(i+n)$ for FNN_1 is obtained from class-memberships values using the defuzzification procedure given in Reference 15.

For FNN_2 and FNN_3, the training patterns used are:

Input pattern: $\Delta\Theta_{max}(i, i+n)$ and $\Delta H_{max}(i, i+n)$

Output pattern: $e_{LC}(i)$, the desired load correction

Here again, the weather variables used for the $(i+n)$th day are the forecast values. The $P_{max}(i+n)$ for FNN_2 and FNN_3 are obtained using eqn. 2.

Table 1 gives the membership functions learned using FNN_2 for 24 h-ahead peak-load forecasting in winter. For example, rule 0 is interpreted as

Table 1: Learned fuzzy logic rules for 24 h-ahead peak-load forecasting using FNN_2 in winter

Rule	Term sets		
	Preconditions		Consequence
	$\Delta\Theta_{max}(i, i+1)$	$\Delta H_{max}(i, i+1)$	$\hat{e}_{LC}(i)$
0	0	3	7
1	0	4	7
2	1	0	8
3	1	1	7
4	1	2	7
5	1	3	6
6	1	4	6
7	2	0	8
8	2	1	7
9	2	2	7
10	2	3	6
11	2	4	7
12	3	1	2
13	3	2	4
14	3	3	6
15	4	2	5
16	4	3	6
17	4	4	1
18	5	0	3
19	5	1	2
20	5	2	1
21	5	3	1
22	5	4	0
23	6	0	1
24	6	1	1
25	6	2	0

RO: IF $\Delta\Theta_{max}$ is term 0 and ΔH_{max} is term 3 THEN $\hat{e}_{LC}$ is term 7.

Fig. 5 gives the membership functions learned for FNN_2 after the first (unsupervised-learning) and second (supervised-learning) phases. Fig. 6 gives the plot of mean absolute percentage errors (MAPEs) against the number of iterations for the ANN, FNN_1, FNN_2 and FNN_3 models, respectively. The results in Figs. 5 and 6 were obtained for 24 h-ahead peak-load forecasting in winter.

From Fig. 6 we see that FNN_3 gives a extremely fast rate of convergence followed by FNN_2, ANN and FNN_1, respectively. The linear Kalman-filter equations and the variable-forgetting factor used for the training of FNN_3 are instrumental in driving the MAPE low during the first few hundred iterations until the bias caused by the initial parameters, arbitrarily chosen, is eliminated. Also the convergence speed of FNN_1 is found to be slower than that of ANN, even though it is trained with the same backpropagation algorithm, because of the increased amount of computation involved in classifying the input and output of FNN_1 into fuzzy-term sets, hence requiring a greater number of weight updates. However, FNN_1 converged to a lower MAPE compared with ANN.

Tables 2 and 3 give the 24 h- and 168 h-ahead peak-load-forecastig results, the number of iterations for convergence and the MAPEs for the month of January (winter) using the ANN and three hybrid models.

From Tables 2 and 3, we see that FNN_3 gives a very accurate prediction, followed in accuracy by FNN_2, FNN_1 and ANN, respectively. Also we find that the 168 h-ahead-prediction results are comparable with the

Table 2: Peak-load forecasting in January (winter) using 24 h-ahead forecast

Day	Actual load	Peak forecast (NN)	PE (NN)	Peak forecast (FNN_1)	PE (FNN_1)	Peak forecast (FNN_2)	PE (FNN_2)	Peak forecast (FNN_3)	PE (FNN_3)
	(MW)	(MW)		(MW)		(MW)		(MW)	
1	2690	2556.6	4.96	2592.4	3.63	2659.3	1.14	2733.2	−1.61
2	2628	2547.0	3.08	2729.5	−3.86	2581.7	1.76	2668.1	−1.52
3	2703	2676.8	0.97	2674.5	1.05	2681.0	0.81	2692.5	0.39
4	2592	2602.6	−0.41	2592.3	−0.01	2604.2	−0.47	2566.3	0.99
5	2530	2547.6	−0.69	2507.0	0.91	2517.2	0.51	2542.6	−0.50
6	2574	2578.1	−0.16	2595.3	−0.83	2581.1	−0.27	2534.6	1.53
7	2389	2382.0	0.29	2400.9	−0.50	2424.0	−1.47	2435.8	−1.96
8	2513	2534.6	−0.86	2487.8	1.00	2496.3	0.66	2528.1	−0.60
9	2500	2394.0	4.24	2396.3	4.15	2594.8	−3.79	2566.9	−2.68
10	2450	2381.1	2.81	2406.1	1.80	2483.0	−1.35	2427.5	0.91
11	2551	2513.9	1.46	2547.0	0.16	2529.5	0.84	2559.8	−0.34
12	2763	2869.2	−3.85	2850.0	−3.15	2845.7	−2.99	2807.4	−1.61
13	2603	2544.2	2.26	2581.0	0.85	2609.6	−0.25	2619.8	−0.64
14	2914	2783.1	4.50	2987.0	−2.50	2883.7	1.04	2909.9	0.14
15	2761	2845.0	−3.05	2719.0	1.53	2754.5	0.24	2740.8	0.73
16	2514	2435.4	3.13	2588.5	−2.97	2562.7	−1.94	2476.9	1.47
17	2543	2648.2	−4.14	2606.4	−2.50	2527.2	0.62	2565.9	−0.90
18	2435	2498.8	−2.62	2488.9	−2.21	2474.3	−1.61	2442.2	−0.30
19	2496	2513.7	−0.72	2493.4	0.10	2490.6	0.21	2501.8	−0.23
20	2551	2485.6	2.56	2524.0	1.06	2593.6	−1.67	2572.9	−0.86
21	2813	2881.9	−2.45	2879.7	−2.37	2831.8	−0.67	2811.5	0.05
22	2537	2435.9	3.99	2483.0	2.16	2582.7	−1.80	2559.8	−0.90
23	2381	2310.5	2.96	2324.0	2.39	2326.7	2.28	2422.3	−1.74
24	2459	2533.8	−3.04	2448.7	0.42	2451.5	0.31	2437.0	0.89
25	2505	2509.3	−0.17	2487.2	0.71	2520.6	−0.62	2512.1	−0.28
26	2429	2506.8	−3.20	2450.2	−0.87	2381.2	1.97	2384.5	1.83
27	2438	2329.1	4.47	2367.4	2.89	2488.4	−2.07	2396.0	1.72
28	2748	2780.6	−1.19	2786.1	1.39	2761.9	−0.51	2741.6	0.23
29	2388	2502.9	−4.82	2334.1	2.26	2358.0	1.26	2421.6	−1.41
30	2175	2211.5	−1.68	2214.6	−1.82	2148.9	1.20	2209.7	−1.60
31	2539	2480.4	2.31	2490.6	1.90	2522.0	0.67	2522.7	0.64
MAPE			2.48		1.74		1.19		1.01
Iterations required			1100		1940		700		490

results obtained for 24 h-ahead predictions as the load forecasting is performed as one-step process and hence the forecast error for past days do not add up to the final

where n is the lead time for the forecast as given in Section 5.1. The forecast load from FNN_1 is obtained in a manner similar to that in Section 5.1.

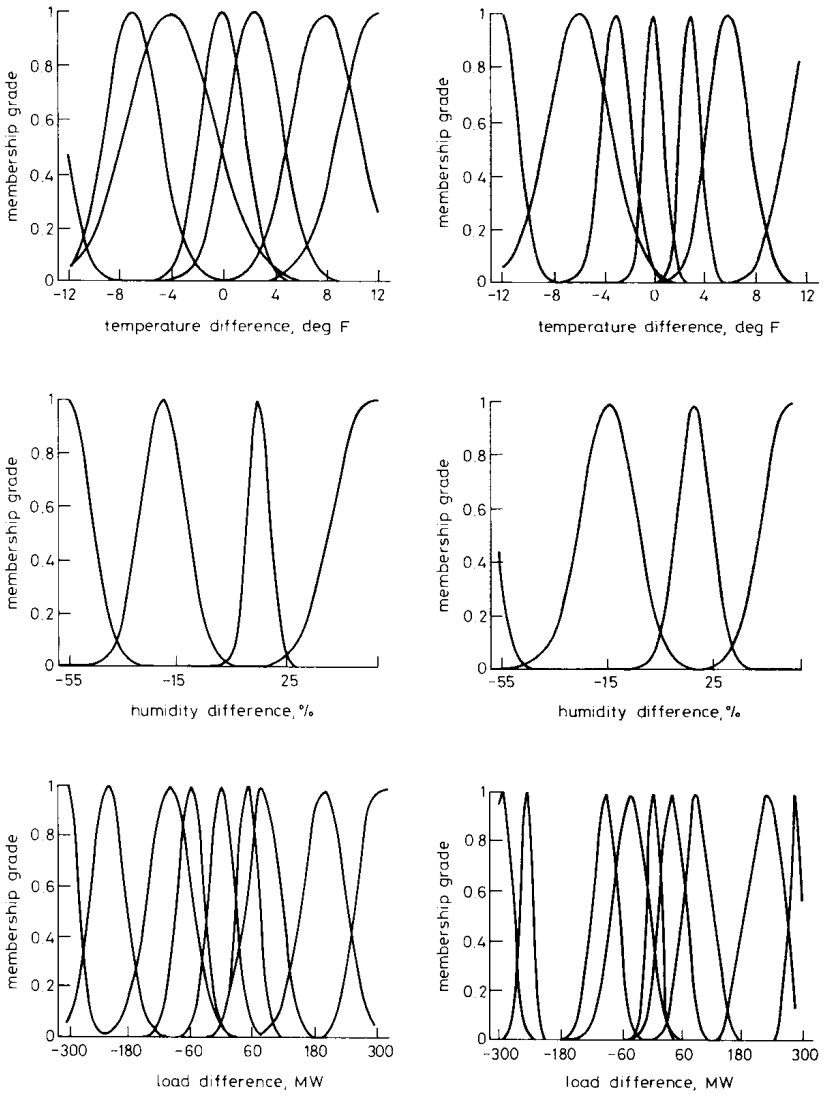

Fig. 5 *Learned membership function for peak-load forecasting in winter using FNN_2*

a Maximum temperature difference (after unsupervised learning)
b Maximum temperature difference (after supervised learning)
c Maximum relative-humidity difference (after unsupervised learning)
d Maximum relative-humidity difference (after supervised learning)
e Maximum peak-load difference (after unsupervised learning)
f Maximum peak-load difference (after supervised learning)

forecast. However, as the lead time is increased to 168, the forecast errors for four days in the month of January using FNN_3 exceeded 4%. As the main purpose of this paper is to make a comparative assessment between FNNs and ANN, no attempt is made to reduce the forecast errors further.

5.2 Average-load forecasting

For average-load forecasting, the following training data are used for ANN and FNN_1:

Input pattern: $P_{av}(i)$, $\Theta_{max}(i)$, $\Theta_{min}(i)$, $H_{max}(i)$, $H_{min}(i)$, $\Theta^f_{max}(i+n)$, $\Theta^f_{min}(i+n)$, $H^f_{max}(i+n)$, $H^f_{min}(i+n)$

Output pattern: $P_{av}(i+n)$ and $\mu\{P_{av}(i+n)\}$ for ANN and FNN_1, respectively,

For FNN_2 and FNN_3, the training patterns used are

Input pattern: $\Delta\Theta_{max}(i, i+n)$, $\Delta\Theta_{min}(i, i+n)$, $\Delta H_{max}(i, i+n)$, $\Delta H_{min}(i, i+n)$

Output pattern: $e_{LC}(i)$, the desired load correction

The $P_{av}(i+1)$ for FNN_2 and FNN_3 is obtained using eqn. 2.

For the average-load forecast also, the forecast temperature and humidity values are used for the day of the forecast.

Table 4 presents the average-load-forecasting results, number of iterations for convergence, PEs and MAPEs for the ANN, FNN_1, FNN_2 and FNN_3 models, respectively, for the month of June using 24 h-ahead predictions. From these results we note the improved performance of

the FNN_3 model in terms of faster convergence and improved overall accuracy, followed by the FNN_2, FNN_1 and ANN models, respectively.

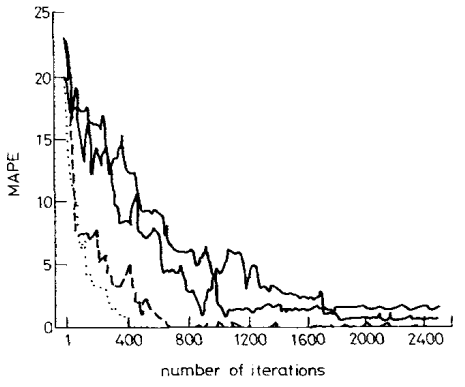

Fig. 6 *Comparison of mean absolute percentage error against iteration number, 24 h-ahead forecast in January (winter)*

—— ANN
—— FNN_1
– – – FNN_2
····· FNN_3

The one-week-ahead average-load forecast is also obtained for the month of June using the above forecasting models, and Table 5 presents these results. The Kalman-filter-based load-forecasting model takes fewer iterations and produces an accurate forecast compared with the other models. It is further observed that the errors in the average-load forecast are comparatively much smaller than for the peak-load forecast.

6 Discussion

The proposed hybrid fuzzy-neural-network models are found to be very powerful in providing an accurate load forecasting. Although the results for two seasons of the year are presented in this paper for validating the effectiveness of this approach, extensive tests have been conducted for other seasons, Sundays, holidays and special days of the year. From the results presented in this paper, it can be observed that significant accuracy can be achieved for 24 h-ahead hourly load forecasts and the PEs can be less than 1. However, the PEs increase for peak-load forecasts and will remain less than 2. If the lead time increases to one week, the Kalman-filter-based hybrid model yields a PE of around 2 for the average-load forecast and around 3 for the peak-load forecast. Further, the results presented in the Table also reveal the superiority of the Kalman-filter-based hybrid-forecasting model over the ANN and other fuzzy-neural-network forecasting models in terms of speed of convergence, MAPE and maximum percentage error.

The accuracy of the hybrid models can be further enhanced by choosing a greater number of fuzzy overlapping sets for fuzzification of input variables instead of the three used for this application. Also, the choice of membershp function is flexible to take into account different seasonal load and weather variables. This increases the

Table 3: Peak-load forecasting in January (winter) using 168 h-ahead forecast

Day	Actual load	Peak forecast (NN)	PE (NN)	Peak forecast (FNN_1)	PE (FNN_1)	Peak forecast (FNN_2)	PE (FNN_2)	Peak forecast (FNN_3)	PE (FNN_3)
	(MW)	(MW)		(MW)		(MW)		(MW)	
1	2690	2534.5	5.78	2578.4	4.15	2630.0	2.23	2742.2	−1.94
2	2628	2556.0	2.74	2531.8	3.66	2564.7	2.41	2571.0	2.17
3	2703	2610.8	3.41	2640.0	2.33	2649.5	1.98	2666.5	1.35
3	2703	2610.8	3.41	2640.0	2.33	2649.5	1.98	2666.5	1.35
4	2592	2651.4	−2.29	2641.0	−1.89	2636.6	−1.72	2551.0	1.58
5	2530	2616.5	−3.42	2586.9	−2.25	2580.1	−1.98	2480.7	1.95
6	2574	2503.5	2.74	2512.0	2.41	2519.9	2.10	2628.8	−2.13
7	2389	2433.2	−1.85	2430.1	−1.72	2351.5	1.57	2354.8	1.43
8	2513	2433.1	3.18	2434.6	3.12	2444.9	2.71	2449.7	2.52
9	2500	2353.8	5.85	2352.8	5.89	2370.8	5.17	2378.5	4.86
10	2450	2324.3	5.13	2332.6	4.79	2340.5	4.47	2360.1	3.67
11	2551	2620.1	−2.71	2502.0	1.92	2510.7	1.58	2519.6	1.23
12	2763	2953.9	−6.91	2944.8	−6.58	2886.0	−4.45	2874.1	−4.02
13	2603	2531.9	2.73	2534.0	2.65	2555.9	1.81	2649.6	−1.79
14	2914	2761.6	5.23	2800.6	3.89	2853.1	2.09	2972.3	−2.00
15	2761	2894.6	−4.84	2878.1	−4.24	2658.3	3.72	2660.2	3.65
16	2514	2389.6	4.95	2389.3	4.96	2408.4	4.20	2415.2	3.93
17	2543	2697.4	−6.07	2675.7	−5.22	2653.9	−4.36	2651.8	−4.28
18	2435	2494.7	−2.45	2401.2	1.39	2454.7	−0.81	2414.8	0.83
19	2496	2463.8	1.29	2481.8	0.57	2509.2	−0.53	2481.5	0.58
20	2551	2456.6	3.70	2453.3	3.83	2493.9	2.24	2514.5	1.43
21	2813	2886.4	−2.61	2878.8	−2.34	2864.5	−1.83	2762.9	1.78
22	2537	2382.0	6.11	2412.4	4.91	2441.9	3.75	2462.2	2.95
23	2381	2279.8	4.25	2281.2	4.19	2285.0	4.03	2285.0	4.03
24	2459	2540.6	−3.32	2537.0	−3.17	2518.5	−2.42	2507.4	−1.97
25	2505	2575.4	−2.81	2573.6	−2.74	2538.8	−1.35	2537.8	−1.31
26	2429	2500.7	−2.95	2478.3	−2.03	2480.5	−2.12	2462.3	−1.37
27	2438	2281.2	6.43	2336.1	4.18	2351.0	3.57	2370.2	2.78
28	2748	2839.0	−3.31	2808.7	−2.21	2808.2	−2.19	2784.3	−1.32
29	2388	2513.8	−5.27	2480.7	−3.88	2481.8	−3.93	2474.2	−3.61
30	2175	2125.4	2.28	2128.9	2.12	2198.9	−1.10	2159.3	0.72
31	2539	2456.2	3.26	2461.1	3.07	2476.3	2.47	2489.2	1.96
	MAPE		3.87		3.30		2.61		2.29
Iterations required			2050		2300		1170		890

Table 4: Average-load forecast in June (summer) using 24 h-ahead forecast

Day	Actual load	Average forecast (NN)	PAE (NN)	Average forecast (FNN_1)	PAE (FNN_1)	Average forecast (FNN_2)	PAE (FNN_2)	Average forecast (FNN_3)	PAE (FNN_3)
	(MW)	(MW)		(MW)		(MW)		(MW)	
1	1521	1527.8	−0.44	1506.5	0.95	1526.5	−0.36	1528.7	−0.51
2	1531	1508.3	1.48	1520.5	0.69	1516.3	0.96	1525.7	0.35
3	1473	1454.4	1.26	1488.0	−1.02	1457.8	1.03	1486.2	−0.90
4	1336	1365.3	−2.20	1308.4	2.06	1361.8	−1.93	1356.1	−1.50
5	1285	1261.7	1.80	1265.3	1.53	1299.0	−1.09	1270.0	1.17
6	1433	1433.2	−0.01	1442.0	−0.63	1436.0	−0.21	1433.7	−0.05
7	1406	1420.4	−1.02	1421.1	−1.10	1415.8	−0.69	1415.3	−0.67
8	1403	1378.3	1.76	1432.2	−2.08	1382.8	1.44	1388.8	1.01
9	1428	1404.5	1.65	1450.0	−1.54	1406.1	1.54	1448.3	−1.42
10	1431	1468.5	−2.61	1473.7	−2.98	1397.7	2.33	1402.3	2.00
11	1301	1280.6	1.57	1321.2	−1.55	1315.1	−1.09	1292.5	0.65
12	1235	1257.5	−1.82	1210.5	1.98	1225.5	0.77	1244.4	−0.76
13	1418	1409.9	0.57	1429.2	−0.79	1421.5	−0.25	1419.8	−0.13
14	1408	1401.4	0.47	1434.4	−1.88	1406.9	0.08	1407.8	0.02
15	1421	1407.2	0.97	1434.4	−0.94	1415.9	0.36	1418.8	0.15
16	1447	1467.8	−1.44	1436.1	0.75	1453.3	−0.43	1437.2	0.68
17	1440	1424.4	1.08	1435.6	0.30	1439.3	0.05	1441.5	−0.11
18	1352	1314.6	2.76	1335.9	1.20	1358.2	−0.46	1358.7	0.49
19	1305	1328.3	−1.79	1327.7	−1.74	1290.1	1.13	1298.0	0.54
20	1430	1444.4	−1.01	1419.4	0.74	1439.7	0.68	1425.9	0.29
21	1383	1373.6	0.68	1379.8	0.23	1370.1	0.93	1390.3	−0.53
22	1414	1417.2	−0.23	1411.5	0.18	1417.9	−0.28	1414.7	−0.05
23	1423	1397.5	1.79	1415.3	0.54	1426.5	−0.25	1421.4	0.11
24	1406	1375.1	2.20	1385.8	1.43	1426.6	−1.46	1389.3	1.19
25	1267	1281.4	−1.14	1260.7	0.50	1255.5	0.91	1269.9	−0.23
26	1221	1245.5	−2.01	1242.9	−1.79	1238.1	−1.40	1235.0	−1.15
27	1397	1380.2	1.20	1405.2	−0.59	1395.9	0.07	1386.2	0.77
28	1401	1436.5	−2.54	1382.2	1.34	1377.0	1.71	1383.4	1.26
29	1403	1401.2	0.13	1408.6	−0.40	1396.5	0.46	1411.2	−0.59
30	1411	1369.7	2.92	1380.6	2.15	1436.5	−1.81	1400.9	0.71
	MAPE		1.42		1.19		0.87		0.67
Iteration required			1480		1800		810		650

Table 5: Average-load forecasting in June (summer) using 168 h-ahead forecast

Day	Actual load	Average forecast (NN)	PE (NN)	Average forecast (FNN_1)	PE (FNN_1)	Average forecast (FNN_2)	PE (FNN_2)	Average forecast (FNN_3)	PE (FNN_3)
	(MW)	(MW)		(MW)		(MW)		(MW)	
1	1521	1558.6	−2.47	1550.2	−1.92	1542.9	−1.44	1538.2	−1.13
2	1531	1558.9	−1.82	1555.3	−1.59	1551.8	−1.36	1550.6	−1.28
3	1473	1436.0	2.51	1453.3	1.34	1452.7	1.38	1456.5	1.12
4	1336	1409.3	−5.49	1393.2	−4.28	1388.2	−3.91	1364.9	−2.16
5	1285	1244.1	3.18	1247.6	2.91	1250.8	2.66	1261.6	1.82
6	1433	1409.4	1.65	1440.9	−0.55	1428.4	0.32	1434.6	−0.11
7	1406	1377.5	2.03	1379.0	1.92	1385.3	1.47	1385.9	1.43
8	1403	1441.2	−2.72	1437.5	−2.46	1435.3	−2.30	1431.1	−2.00
9	1428	1484.1	−3.93	1472.8	−3.14	1471.8	−3.07	1448.7	−1.45
10	1431	1469.6	−2.70	1447.6	−1.16	1447.0	−1.12	1416.5	1.01
11	1301	1358.0	−4.38	1347.3	−3.56	1345.8	−3.44	1330.0	−2.23
12	1235	1202.5	2.63	1208.2	2.17	1212.5	1.82	1221.7	1.08
13	1418	1390.3	1.95	1402.0	1.13	1403.0	1.06	1432.0	−0.99
14	1408	1394.9	0.93	1421.5	−0.96	1399.7	0.59	1411.0	−0.21
15	1421	1484.4	−4.46	1465.3	−3.12	1453.7	−2.30	1451.8	−2.17
16	1447	1482.3	−2.44	1478.5	−2.18	1473.2	−1.81	1463.8	−1.16
17	1440	1402.7	2.59	1406.7	2.31	1412.1	1.94	1415.5	1.70
18	1352	1308.5	3.22	1337.1	1.10	1357.7	−0.42	1358.1	−0.45
19	1305	1269.2	2.74	1279.2	1.98	1290.0	1.15	1295.0	0.77
20	1430	1404.0	1.82	1404.5	1.78	1406.3	1.66	1441.9	−0.83
21	1383	1350.4	2.36	1406.4	−1.69	1363.2	1.43	1399.7	−1.21
22	1414	1463.1	−3.47	1457.3	−3.06	1448.6	−2.45	1396.6	1.23
23	1423	1371.5	3.62	1386.4	2.57	1393.0	2.11	1404.8	1.28
24	1406	1359.5	3.31	1361.1	3.19	1372.1	2.41	1389.7	1.16
25	1267	1219.4	3.76	1229.6	2.95	1237.6	2.32	1254.2	1.01
26	1221	1256.7	−2.92	1247.6	−2.18	1241.6	−1.69	1236.1	−1.24
27	1397	1372.7	1.74	1373.7	1.67	1384.6	0.89	1387.9	0.65
28	1401	1373.1	1.99	1371.9	2.08	1379.7	1.52	1380.5	1.46
29	1403	1414.4	−0.81	1391.1	0.85	1405.5	−0.18	1399.5	0.25
30	1411	1359.6	3.64	1380.4	2.17	1374.9	2.56	1378.1	2.33
	MAPE		2.78		2.13		1.76		1.23
Iterations required			2650		2740		2480		1310

number of rules and consequently the rule nodes in the hybrid model. The database used for this study comprises a 14-day period prior to the day of forecast; thus by using a larger database (say four weeks) and an increased number of load and weather parameters as input variables, it is possible to obtain a more accurate and robust forecast for periods between one day and one week ahead.

The authors have also performed extremely short-term predictions from 1 h to 6 h ahead over the next 24-h period using the hybrid models, and the results reveal a significant improvement in accuracy compared with those from 24 h-ahead forecasts. The main features and advantages of the hybrid model are:

(i) it provides us with a general method of combining available numerical information and human linguistic information into a common framework;

(ii) it requires much less construction time than a comparable neural network;

(iii) significant accuracy in predicting chaotic time-series models.

7 Conclusions

This paper presents three fuzzy-neural-network (FNN) models for time-series forecasting of electric load. The first model, FNN_1, uses the fuzzy-membership values of the past load and weather parameters, and the output of the FNN_1 gives the class-membership values of the forecast load. The second and third models, FNN_2 and FNN_3, introduce the low-level learning power of an artificial neural network into a fuzzy expert system and provide high-level human-understandable meaning to the normal neural network. A hybrid learning scheme consisting of a self-organised-learning phase and a supervised-learning phase is used for training FNN_2 and FNN_3. Also the Kalman-filter update equations in the supervised-learning phase of FNN_3 give better convergence and accuracy than the gradient-descent back-propagation algorithm in the supervised-learning phase of FNN_2.

8 References

1 ZADEH, L.A.: 'Fuzzy sets', *Inf. Control*, 1965, **8,** pp. 338–358
2 BOX, G.E.P., and JENKINS, G.M.: 'Time series analysis forecasting and control' (Holden–Day, Oakland, CA, 1976)
3 MOGHRAM, I., and RAHMAN, S.: 'Analysis and evaluation of five short-time load forecasting techniques', *IEEE Trans.*, 1989, **PWRS-4,** (4), pp. 1484–1490
4 BUNN, D.W., and FARMER, E.D.: 'Comparative models for electric load forecasting' (Wiley, 1989)
5 WIDROW, B., and LEHR, M.A.: '30 years of adaptive neural networks: perceptron, madaline, and backpropagation', *Proc. IEEE*, 1990, **78,** (9), pp. 1415–1442
6 RUMELHART, D.E., and ZIPSES, D.: 'Feature discovery by competitive leaning', *Cogn. Sci.*, 1985, **9,** pp. 75–112
7 EL-SHARKAWI, M.A., OH, S., MARKS, R.J., DAMBOURG, M.J., and BRACE, C.M.: 'Short term electric load forecasting using an adaptively trained layered perceptron'. First international forum on Applications of Neural Networks to Power Systems, Seattle, WA, 1991, pp. 3–6
8 LEE, K.Y., CHA, Y.T., and PARK, J.H.: 'Short term load forecasting using an artificial neural network', *IEEE Trans.*, 1992, **PWRS-7,** (1), pp. 124–133
9 PENG, T.M., HUBELE, N.F., and KARADY, G.G.: 'Advancement in the application of neural networks for short term load forecasting', *IEEE Trans.*, 1992, **PWRS-7,** (1), pp. 250–258
10 RAHMAN, S., DREZGA, I., and RAJAGOPALAN, J.: 'Knowledge enhanced connectionist models for short-term electric load forecasting'. International Conference on ANN applications to Power Systems, Japan, April 1993
11 DASH, P.K., DASH, S., RAMA KRISHNA, G., and RAHMAN, S.: 'Forecasting of a load time series using a fuzzy expert system and fuzzy neural networks', *Eng. Int. Syst.*, 1993, **1,** (2), pp. 103–117
12 PARK, J.H., PARK, Y.M., and LEE, K.Y.: 'Composite models for adaptive short term load forecasting', *IEEE Trans.*, 1991, **PWRS-1,** (2), pp. 450–457
13 CHIN-TENG LIN, and LEE, C.S.G.: 'Neural network-based fuzzy logic control and decision system', *IEEE Trans.*, 1991, **C-40,** (12), pp. 1320–1336
14 HAYASHI, Y., BUCKLEY, J.J., and CZOGALA, E.: 'Fuzzy expert systems versus neural networks'. Proceedings of international joint conference on Neural Networks, Baltimore, USA, 1992. Vol. II, pp. 720–726
15 WANG, L.-X., and MENDEL, J.M.: 'Generating fuzzy rules by learning from examples', *IEEE Trans.*, 1992, **SMC-22,** (6), pp. 1414–1427
16 SCALERO, R.S., and TEPEDELENLIOGLU, N.: 'A fast algorithm for training feedforward neural networks', *IEEE Trans.*, 1992, **SSP-40,** (1), pp. 202–210

9 Appendixes

9.1 Appendix 1

This Appendix gives the update equations for FNN_2 (shown in Fig. 3) using the backpropagation algorithm

(*a*) The weight-update equations for layer four is

$$W_{ij}(t) = W_{ij}(t) + \eta\left[\frac{-\partial E}{\partial W_{ij}}\right] \tag{14}$$

The error function E is given by

$$E = \tfrac{1}{2}\{e_{LC}(t) - \hat{e}_{LC}(t)\}^2 \tag{15}$$

Since

$$\hat{e}_{LC} = \frac{\sum (a_{ij} b_{ij})\mu_i^5}{\sum b_{ij}\mu_i^5} \tag{16}$$

and using centroid-defuzzification method [15] we obtain

$$\mu_i^5 = \mu^4 = \sum_{i=1}^{p} \mu_i^4 W_{ij} \tag{17}$$

where

$$W_{ij} = 1 \quad \text{for } t = 1$$

Therefore

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial \hat{e}_{LC}} \frac{\partial \hat{e}_{LC}}{\partial \mu_i^5} \frac{\partial \mu_i^5}{\partial W_{ij}} \tag{18}$$

From eqn. 18, we obtain

$$\frac{\partial E}{\partial W_{ij}} = \{e_{LC}(t) - \hat{e}_{LC}(t)\} \times \left\{\frac{a_{ij} b_{ij}(\sum b_{ij}\mu_i^5) - (\sum a_{ij} b_{ij}\mu_i^5) b_{ij}}{(\sum b_{ij}\mu_i^5)^2}\right\}\mu_i^4 \tag{19}$$

(*b*) Training the output membership functions at layer 5:

$$a_{ij}(t+1) = a_{ij}(t) + \eta_1\left[\frac{-\partial E}{\partial a_{ij}}\right] \tag{20}$$

where η is the learning rate. Now

$$\frac{\partial E}{\partial a_{ij}} = \frac{\partial E}{\partial \hat{e}_{LC}} \frac{\partial \hat{e}_{LC}}{\partial a_{ij}} \tag{21}$$

Therefore, using eqns. 15 and 16 we obtain

$$a_{ij}(t+1) = a_{ij}(t) + \eta_1\{e_{LC}(t) - \hat{e}_{LC}(t)\}\left\{\frac{b_{ij}\mu_i^5}{(\sum b_{ij}\mu_i^5)}\right\} \quad (22)$$

Similarly,

$$b_{ij}(t+1) = b_{ij}(t) + \eta_2\left[\frac{-\partial E}{\partial b_{ij}}\right] \quad (23)$$

where η_2 is the learning rate. Now

$$\frac{\partial E}{\partial b_{ij}} = \frac{\partial E}{\partial \hat{e}_{LC}}\frac{\partial \hat{e}_{LC}}{\partial b_{ij}} \quad (24)$$

Therefore, using eqns. 15 and 16 we obtain

$$b_{ij}(t+1) = b_{ij}(t) + \eta_2\{e_{LC}(t) - \hat{e}_{LC}(t)\} \times \left\{\frac{a_{ij}\mu_i^5(\sum b_{ij}\mu_i^5) - (\sum a_{ij}b_{ij}\mu_i^5)\mu_i^5}{(\sum b_{ij}\mu_i^5)^2}\right\} \quad (25)$$

(*c*) Training the input membership functions at layer 2: The error signal δ_i^4 at layer four is given by

$$\delta_i^4 = \{e_{LC}(t) - \hat{e}_{LC}(t)\} \times \left\{\frac{a_{ij}b_{ij}(\sum b_{ij}\mu_i^5) - (\sum a_{ij}b_{ij}\mu_i^5)b_{ij}}{(\sum b_{ij}\mu_i^5)^2}\right\} \quad (26)$$

where (a_{ij}, b_{ij}) correspond to the output-term set.

The error signal at layer three is found by performing the summation over the consequences of a rule node (i.e. layer four). Therefore

$$\delta_i^3 = \sum \delta_i^4 \quad (27)$$

The adaptive rule for a_{ij} (layer 2) is derived as

$$a_{ij}(t+1) = a_{ij}(t) + \eta_3\left[\frac{-\partial E}{\partial a_{ij}}\right] \quad (28)$$

where η_3 is the learning rate. Now

$$\frac{\partial E}{\partial a_{ij}} = \delta_i^2\left[\exp\left\{\frac{-(x_i - a_{ij})^2}{b_{ij}}\right\}\right]\left\{\frac{2(x_i - a_{ij})}{b_{ij}}\right\} \quad (29)$$

Also

$$\delta_i^2 = \sum \delta_i^3 \quad (30)$$

in similar way as for eqn. 27. Therefore

$$a_{ij}(t+1) = a_{ij}(t) - \eta_3\,\delta_i^2\left[\exp\left\{\frac{-(x_i - a_i)^2}{b_{ij}}\right\}\right] \times \left\{\frac{2(x_i - a_{ij})}{b_{ij}}\right\} \quad (31)$$

Similarly, the adaptive rule for b_{ij} (layer 2) is derived as

$$b_{ij}(t+1) = b_{ij}(t) - \eta_4\,\delta_i^2\left[\exp\left\{\frac{-(x_i - a_{ij})^2}{b_{ij}}\right\}\right] \times \left\{\frac{(x_i - a_{ij})^2}{b_{ij}^2}\right\} \quad (32)$$

9.2 Appendix 2

This Appendix gives the update equations for FNN_s using the linear Kalman-filter equations.

(*a*) The weight-update equation for layer four is

$$W_{ij}(t) = W_{ij}(t) + \eta K_j(t)\left[\frac{-\partial E}{\partial W_{ij}}\right] \quad (33)$$

where $(\partial E/\partial W_{ij})$ is given by eqn. 19 and $K_j(t)$ is the Kalman gain and is given by

$$K_j(t) = \left\{\frac{R_j^{-1}(t)x_i(t)}{f_j + x_i^T(t)R_j^{-1}(t)x_i(t)}\right\} \quad (34)$$

where $x_i(t)$ corresponds to the previous layer.

The forgetting factor f_j and the inverse convariance matrix $R_j^{-1}(t)$ are updated using

$$f_j(t+1) = f_0 f_j(t) + (1 - f_0) \quad (35)$$

$$R_j^{-1}(t+1) = \{R_j^{-1}(t) - K_j(t)x_i^T(t)R_j^{-1}(t)\}/f_j \quad (36)$$

(*b*) The update equations for a_{ij} and b_{ij} at layer five are

$$a_{ij}(t+1) = a_{ij}(t) + \eta_1\,K_j(t)\left[\frac{-\partial E}{\partial a}\right] \quad (37)$$

where $(\partial E/\partial a_{ij})$ is given by eqn. 21.

$$b_{ij}(t+1) = b_{ij}(t) + \eta_2\,K_j(t)\left[\frac{-\partial E}{\partial b_{ij}}\right] \quad (38)$$

where $(\partial E/\partial b_{ij})$ is given by eqn. 24.

(*c*) The update equations for a_{ij} and b_{ij} at layer two are

$$a_{ij}(t+1) = a_{ij}(t) + \eta_3\,K_j(t)\left[\frac{-\partial E}{\partial a_{ij}}\right] \quad (39)$$

where $(\partial E/\partial a_{ij})$ is given by eqn. 29. Similarly,

$$b_{ij}(t+1) = b_{ij}(t) - \eta_4\,K_j(t)\delta_i^2\left[\exp\left\{\frac{-(x_i - a_{ij})^2}{b_{ij}}\right\}\right] \times \left\{\frac{(x_i - a_{ij})^2}{b_{ij}^2}\right\} \quad (40)$$

where δ_i^2 is given by eqn. 30.