International Journal of Fuzzy Logic and Intelligent Systems, vol. 9, no. 4, December 2009 pp. 327−332

Short-term Electrical Load Forecasting Using Neuro-Fuzzy Model with Error Compensation

Bo-Hyeun Wang

Department of Electrical Engineering, Kangnung-Wonju National University
123, Chibyun-Dong, Kangnung, Kangwon, Korea

Abstract

This paper proposes a method to improve the accuracy of a short-term electrical load forecasting (STLF) system based on neuro-fuzzy models. The proposed method compensates load forecasts based on the error obtained during the previous prediction. The basic idea behind this approach is that the error of the current prediction is highly correlated with that of the previous prediction. This simple compensation scheme using error information drastically improves the performance of the STLF based on neuro-fuzzy models. The viability of the proposed method is demonstrated through the simulation studies performed on the load data collected by Korea Electric Power Corporation (KEPCO) in 1996 and 1997.

Key Words : Load forecasting, Neuro-fuzzy model, Structure identification, Compensation by prediction error

1. Introduction

The short-term electrical load forecasting systems that provide the prediction of the system load over an interval ranging from one hour to one week are used for hydro scheduling, unit commitment, and economic load dispatch. The accuracy and reliability requirements of the short-term load forecasting (STLF) systems have become stricter since optimal utilization of generators and power stations completely depends on the performance of the STLF.

In order to cope with such stricter requirements, a number of techniques to build STLF models have been suggested in the last few decades. They include statistical techniques [1-2], expert system based approaches [3] and artificial neural network based models [4-7].

Recently, artificial neural networks have been considered as a very promising approach for developing STLF models of enhanced performance. Park et al. [4] presented that multilayer feedforward neural networks could be applied to construct STLF models. In their method, the multilayer feedforward networks learned the relationship between the input variables and the system loads. Chow et al. [5] investigated a method to improve the performance of STLF models by using weather factors for the inputs of the multilayer feedforward neural networks. Khotanzad et al. [6] proposed an artificial neural network approach in which multiple modules were elaborately designed to predict weekday loads as well as weekend and special day loads.

In order to overcome the black box drawbacks of the multilayer feedforward neural networks, Bakirtzis et al. [8] presented a STLF model of 24-hours lead-time by using neuro-fuzzy models. In their approach, 168 separate neuro-fuzzy models were constructed in which each neuro-fuzzy model was trained for a particular hour of the day. Mori et al. [9] developed an one-hour ahead load forecasting system by using a neuro-fuzzy model of 36 fuzzy in-then rules.

Park and Wang [10] showed that a proper initialization scheme for neuro-fuzzy models can improve the performance of load forecasting system. Shim and Wang [11] proposed a systematic method to compute a reliability measure for STLF using neuro-fuzzy models.

This paper further improves the accuracy performance of STLF using neuro-fuzzy models incorporating the error information acquired over the past prediction procedure with the current prediction. The idea behind the error compensation comes from the bias normally observed on the prediction models trained through the past data. The proposed enhanced scheme is tested on the data obtained from KEPCO in 1997 and shows a considerable improvement can be made, suggesting higher possibility of practical use of the proposed STLF based on neuro-fuzzy models.

2. Neuro-Fuzzy Model based Short-term Load Forecasting

2.1 Basics of neuro-fuzzy models

A neuro-fuzzy model can represent a fuzzy rulebase that consists of a set of fuzzy if-then rules of the following form:

$$\begin{aligned}
&\text{Rule } 1: \text{If } (x_1 \text{ is } A_1^1) \text{ and } (x_2 \text{ is } A_1^1) \text{ and } \ldots \text{and } (x_n \text{ is } A_n^1), \text{ then } y \text{ is } q^1,\\
&\text{Rule } 2: \text{If } (x_1 \text{ is } A_1^2) \text{ and } (x_2 \text{ is } A_1^2) \text{ and } \ldots \text{and } (x_n \text{ is } A_n^2), \text{ then } y \text{ is } q^2,\\
&\qquad\qquad \text{M}\\
&\text{Rule } p: \text{If } (x_1 \text{ is } A_1^p) \text{ and } (x_2 \text{ is } A_1^p) \text{ and } \ldots \text{and } (x_n \text{ is } A_n^p), \text{ then } y \text{ is } q^p,
\end{aligned} \tag{1}$$

Manuscript received Aug. 17, 2009; revised Nov. 30, 2009.
This work was supported by the 2008 Kangnung National University Research Foundation Grant..
The author would like to thank Korea Electric Power Corporation for providing the system load data.

where x_j ($1 \le j \le n$) are the input variables, y is the output variable, and A_j^i and q^i ($1 \le i \le p$) are fuzzy sets characterized by the membership functions.

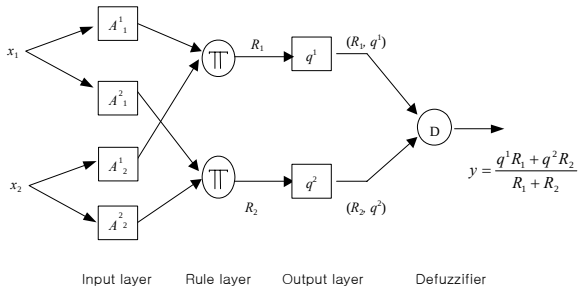


Fig. 1 Architecture of a neuro-fuzzy model.

A class of feedforward neural networks can implement the neuro-fuzzy systems [12, 13]. Suppose that we have two fuzzy rules where each fuzzy rule has two inputs and a single output. The architecture of the corresponding neuro-fuzzy model is shown in Fig. 1. The input term nodes in the first layer represent the membership functions given in the fuzzy values of the premise part. The input term node denoted as A_j^i accepts x_j as the input and produces the degree of matching between x_j and its corresponding membership function. If we use the Gaussian membership functions for A_j^i, the output of the input term node is given by

$$\mu_{A_j^i}(x_j) = \exp\left[-\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right], \tag{2}$$

where c_j^i and σ_j^i are called the premise parameters.

The ith node in the second layer produces the output that represents the firing strength of the ith rule:

$$R_i(x) = \prod_{j=1}^{n} \mu_{A_j^i}(x_j) \tag{3}$$

The output term node in the third layer computes the value of the normalized firing strengths:

$$\overline{R}_i = \frac{R_i}{\sum_{j=1}^{n} R_j} \tag{4}$$

The last layer acts as the defuzzifier. If we use the center of gravity defuzzification from local centroids, the output of this layer can be written as

$$y = \sum_{i=1}^{p} q^i \overline{R}_i . \tag{5}$$

2.2 Structure identification of NFM

Neuro-fuzzy modeling involves structure learning and parameter learning. Structure learning of the NFM deals with two sub-problems: input variables selection and input space partitioning. The problem of input variables selection is to select a set of relevant input variables from finite number of possible candidates. The problem of input space partitioning, on the other hand, is to find a set of initial fuzzy partitions. We can initialize the NFM by using the initial fuzzy partitions obtained since they correspond to the parameters of the input term nodes of the NFM.

A number of attempts have been made to partition the input space of the NFM. Sun [14] proposed fuzzy k-d trees which partition the input space from a series of guillotine cuts. Sugeno and Yasukawa [15] developed a scatter partitioning method that used the fuzzy c-means algorithm. Kubat [16] developed a method to initialize the parameters of the radial basis function networks by using the decision tree learning.

This paper partitions the input space by the method of Kubat since the decision tree learning is very fast. Fig. 5 shows the procedure of the input space partition of NFM using the decision tree. The procedure first performs the clustering on the output data and projects the resultant clusters into the input space. At this point, the problem of function approximation is converted to that of classification. The procedure next applies the decision tree learning to the converted classification problem and initializes the NFM using the results of the decision tree learning. Since the goal of the input space partitioning is to provide a better initial guess for the parameter learning, its performance has to be evaluated on the model resulted from the parameter learning. For this, the method starts with the minimum number of clusters, i.e., c=2 and then increments it until the identified model satisfies the prespecified conditions.

Initial clustering and projection: The initial clustering clusters the output data and projects the resultant clusters into the input space. For the initial clustering we adopt the method presented in [17] which uses the fuzzy c-means algorithm [18]. As a result of the clustering, every output data y^i is associated with the grade of membership belonging to the fuzzy clusters $\tilde{O}_j$'s where i=1,2,…,N and j=1,2,…,c:

$$[y^i; \mu_{\tilde{O}_1}(y^i), \mu_{\tilde{O}_2}(y^i), \mathrm{K}, \mu_{\tilde{O}_c}(y^i)], \tag{6}$$

where $\mu_{\tilde{O}}(y^i)$ is the grade of the ith data belonging to the jth cluster, N is the number of data to be clustered, and c is the number of clusters.

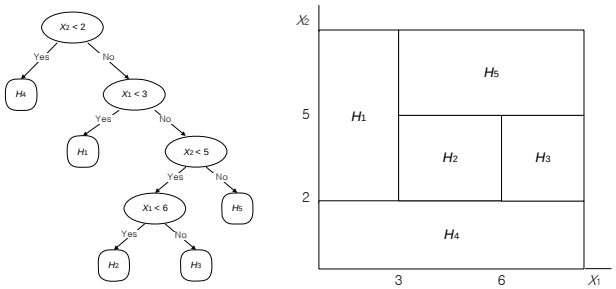
Once we complete the clustering, we project the fuzzy clusters in the output space into the input space. For this, we first transform the fuzzy clusters into the crisp clusters. The crisp cluster to which y^i belongs is determined by taking the fuzzy cluster whose grade of membership of y^i is the maximum:

$$y^i \in O_j \text{ if } \mu_{\tilde{O}_j}(y^i) = \max\{\mu_{\tilde{O}_1}(y^i), \mu_{\tilde{O}_2}(y^i), \mathrm{K}, \mu_{\tilde{O}_c}(y^i)\}. \tag{7}$$

A projected input cluster P_j is found by identifying a group of the input data, which are associated with all the output data belonging to a crisp cluster O_j. This leads us to c groups of the input data.

Decision tree learning: The groups of the input data resulted from the projection have arbitrary shapes. It is necessary to refine the shapes of the groups, since the neuro-fuzzy models

can implement only hyper-rectangular partitions. The decision tree learning can serve as a tool for the shape refinement. Suppose that we apply the decision tree learning to a classification problem and the tree learning produces the binary tree shown in Fig. 2.(a). We can interpret the binary tree as the partitions of the input space as shown in Fig. 2.(b). It should be noted that a leaf node in the binary tree corresponds to a fuzzy rule of the NFM and the number of leaf nodes is that of the fuzzy rules of the NFM. For the simulations in Section 4, C4.5 algorithm developed by Quinlan is used [19].



(a) (b)

Fig. 2 A binary decision tree and the associated input space partitions.

Initialization of NFM: We initialize the premise parameters of the neuro-fuzzy model based on a binary tree produced by the decision tree learning. Suppose that we try to initialize a NFM that has n input variables. A fuzzy rule in the NFM is represented by n-dimensional hyper-rectangle in the input space. The initialization is done by assigning a membership function to each edge of the hyper-rectangle. If we assume that the Gaussian membership functions are employed, the centers of the Gaussian membership functions for the ith hyper-rectangle H_i are simply given by

$$c_j^i = x_{j\min}^i + \frac{x_{j\max}^i - x_{j\min}^i}{2}, \text{for } i = 1,2,\text{K}\ , p \text{ and } j = 1,2,\text{K}\ , n, \quad (8)$$

where $x^i_{j\min}$ and $x^i_{j\max}$ are the values of the left edge and the right edge in the x_j-direction, respectively. The width of the membership function is given by

$$\sigma_j^i = \frac{x_{j\max}^i - c_j^i}{[\ln(1/\alpha)]^{1/2}}, \text{ for } i = 1,2,\text{K}\ , p \text{ and } j = 1,2,\text{K}\ , n, \quad (9)$$

where α is the value of the α-cut fuzzy set, which is used for constructing the hyper-rectangle. The consequent parameter associated with H_i is initialized by using the centers of the output fuzzy clusters:

$$q^i = v_m, \quad (10)$$

if the number of data belonged to the mth cluster in H_i is the maximum.

2.3 Construction of STLF system using NFM

In [10], Park and Wang applied the input space partitioning method to construct more accurate and reliable NFM based STLF system. This section reviews their approach, the one-hour ahead load forecasting system using NFM. Fig. 3 shows the schematic diagram of the method that consists of the input space partitioning of NFM for constructing initial structure bank, and the parameter learning of NFM for building STLF. An important feature of the method is to decompose the whole prediction problem into a number of smaller sub-problems. The decomposition is supported by the fact that the daily load patterns can be classified into 4 groups [7]. For instance, the hourly load shapes of the weekdays from Tuesday to Friday are similar to each other. However, the load shapes of Saturday, Sunday and Monday are dissimilar to those of the weekdays.

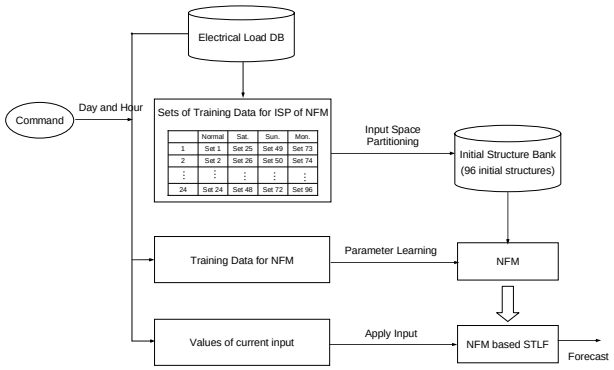


Fig. 3 NFM based short-term load forecasting system.

Table 1. Input variables and output variable.

Input Variables	Output Variable
$p(i, t\text{-}1)$, $p(i\text{-}1, t)$	$p(i, t)$

The method employs two input variables listed in Table 1. In Table 1, i denotes the day of prediction and t represents the time of the day. Given the input variables, we can construct a set of training data from the electrical load database for initial structure bank. It should be noted that the input variable selection is a very important factor in order to obtain a satisfactory performance of a prediction model. It is quite surprising that only two input variables listed in Table 1 can build STLF successfully.

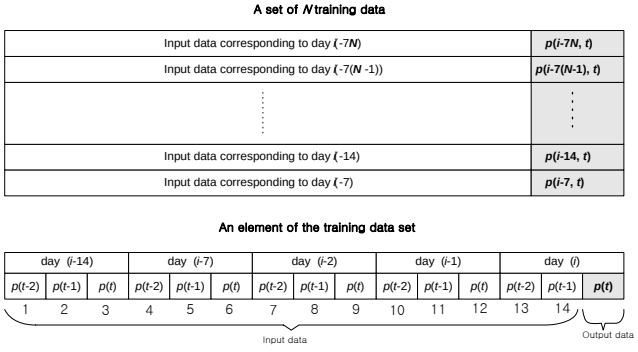


Fig. 4 A set of training data.

The initial structure bank is created by using the procedure shown in Fig. 5. In order to create an initial structure IS(d,t), we initially set the number of clusters to 2 and perform clustering task by using the training data associated with time t of day type d. We next apply the decision tree learning algorithm. If the number of rules generated by the decision tree learning is less than that of predetermined allowable rules, then we initialize a NFM and train the initialized NFM by using the method in [20]. If the performance of the trained NFM is better than that of requirement, we save the set of the initial parameters into the initial structure bank. Otherwise, we increment the number of clusters and repeat the whole procedure. In the case that the number of the resultant rules is greater than that of the allowable rules, we prune the decision tree so that it can generate the initial structure of a fewer number of rules. We save the resulting initial structure into the initial structure bank.

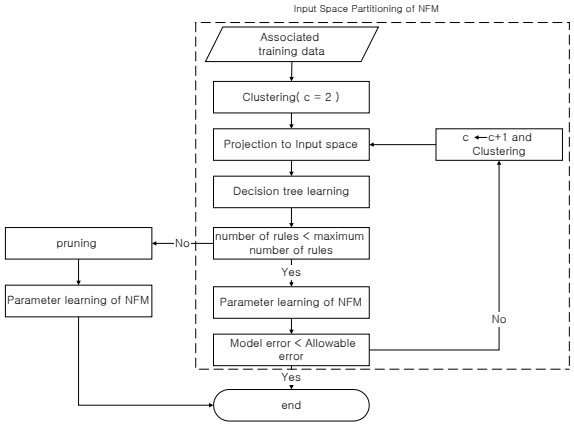


Fig. 5 Algorithm to create an initial structure.

In order to compute the one-hour ahead load forecast $\hat{p}(i,t)$, where day i belongs to day type d, we download the initial structure $IS(d,t)$ from the initial structure bank and initialize the parameters of the NFM. Using the training data associated with the prediction time, we optimize the parameters of NFM. It should be noted that the initialization scheme using the initial structure speeds up the parameter optimization. After the parameter learning, we apply the input vector to the trained NFM in order to obtain the load forecast. It should be noted that the proposed method builds STLF with lead-time of one hour by training the NFM every hour.

3. Improving NFM based STLF using Error Compensation

Fig. 6 shows the typical forecasting results of a day in summer season where the prediction performance is severely degraded. In Fig. 6, the 'x' marks are the forecasted loads, the centered solid line is the actual load, and the outer solid lines represent 90% confidence interval. The confidence interval itself is not reliable for a day when a load forecasting is not accurate.

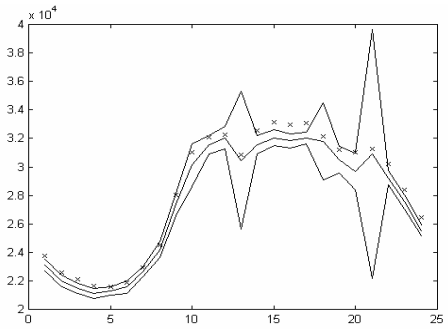


Fig. 6 Sample outputs of the NFM based STLF.

A certain level of bias exists in the forecasting, providing the major cause for the degradation. This observation leads to the improvement scheme that uses the prior forecasting error for adjusting the current load forecasting.

Although there are a lot of candidates to compensate the predicted load, we employ a method reported in [21]:

$$cp(i,t) = p(i,t)\left[1 - \alpha \cdot PE(i,t-1)/100\right] \qquad (11)$$

where $cp(i, t)$ is the compensated load forecast, $PE(i, t\text{-}1)$ is the percent error obtained from the forecast at the previous hour, t-1, and α is the scaling factor ranging from 0 to 1.

The compensation in (11) subtracts a small amount proportional to the percent error obtained at the previous hour. The proportional constant, α, has to be determined by a trial error method. Extensive simulations have been performed in order to find a better constant that produces the best compensation. It turns out that finding a single value of α is not possible, since different α produces the best compensation according to the days and months. There is, however, a certain trend, so that we can set α to the value that produces a better results on average. We will discuss the details of this in Section 4.

4. Simulation Results

The proposed method was implemented and tested on the actual load data collected by KEPCO in 1997. The training data from Jan. 11, 1997 to Jan. 20, 1997 were used for constructing the initial structure bank, shown in Table 2.

Table 2. Training data set for the initial structure bank.

Type	Date	Period	Number of training data
Weekdays	Jan. 14, 1997 ~ Jan. 17, 1997	4 days	200
Saturday	Jan. 11, 1997 Jan. 18, 1997	2 days	100
Sunday	Jan. 12, 1997 Jan. 19, 1997	2 days	100
Monday	Jan. 13, 1997 Jan. 20, 1997	2 days	100

According to the procedure given in Fig. 5, 96 initial structures were generated. Table 3 summarizes the results of the input space partitioning when the maximum number of fuzzy rules and the allowable error are set to 10 and 1%, respectively. The initial structures for the weekdays have 3.9 rules on average, while the average numbers of the rules for Sat., Sun. and Mon. are 4.2, 3.9 and 3.9, respectively. We initialized the NFM by using the initial structure corresponding to the prediction time. We also prepared the training data set consisted of 50 elements for learning the parameters of the initialized NFM.

We initialized the NFM by using the initial structure corresponding to the prediction time. We also prepared the training data set consisted of 50 elements for learning the parameters of the initialized NFM. Its structure is illustrated in Fig. 4. According to the input and output variables, the 12th and the 14th components of the input data in Fig. 4 were picked up to prepare the training data set.

In order to demonstrate the effectiveness of the proposed compensation scheme, we applied it to the period of summer season (four months) where forecasting performance is often degraded. We use mean absolute percentage error (MAPE) for assessing the performance of the STLF:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{p - \hat{p}}{p}\right| \times 100(\%), \qquad (12)$$

where p is the actual load and $\hat{p}$ is the forecasted load.

To find the performance of the NFM based STLF, refer to [10]. The purpose of this paper is to increase the performance of the STLF, assuming that the baseline prediction is given in the approach in [10].

Table 3. Number of fuzzy rules and training error for the initial structure.

Type	Weekdays		Saturday		Sunday		Monday	
Hour	Number of rule	Training error	Number of rule	Training error	Number of rule	Training error	Number of rule	Training error
1	2	0.6061	6	0.7664	6	0.8303	4	0.6362
2	4	0.5489	2	0.4193	5	0.5341	2	0.6314
3	2	0.4658	4	0.4170	4	0.4027	2	0.5708
4	4	0.4404	4	0.3882	4	0.4548	2	0.4423
5	2	0.3618	2	0.4350	2	0.3997	4	0.4901
6	4	0.4751	4	0.3232	4	0.3305	5	0.5273
7	2	0.5359	2	0.5409	4	0.5609	4	0.6033
8	4	0.7335	5	0.6778	7	0.6776	5	0.8469
9	4	1.1010	8	0.9054	4	0.8749	8	0.9529
10	8	1.8935	7	1.4056	4	0.7867	4	0.7743
11	4	0.5368	2	0.5832	2	0.6409	2	0.4778
12	4	0.4102	4	0.5718	2	0.8984	2	0.5801
13	4	0.5889	2	0.7160	2	0.8039	5	0.8933
14	2	0.6960	4	0.8538	2	0.6179	2	0.6272
15	2	0.4705	4	0.6074	2	0.5828	2	0.5392
16	2	0.4264	4	0.7400	4	0.5138	4	0.5195
17	4	0.7911	4	0.6812	5	0.7716	4	0.4805
18	6	1.8384	6	1.2361	7	0.8733	8	0.8326
19	5	1.2601	4	0.8425	4	1.1207	7	0.8646
20	7	0.9879	8	0.7219	6	0.9237	6	0.8625
21	5	0.9325	5	0.7500	6	0.8904	4	0.7906
22	4	0.4177	2	0.3242	4	0.3894	2	0.3976
23	4	0.5792	4	0.8719	2	0.8120	2	0.5942
24	4	0.5586	4	0.6461	2	0.7589	4	0.4214
Ave.	3.9	0.7357	4.2	0.6844	3.9	0.6854	3.9	0.6399

The proposed error compensation scheme is required to set α properly. We tried several values of α and found α =0.9 produced the best compensation on average. It should be noted that in [21] they obtained the best result at α =0.3. The results of the error compensation in load forecasting are listed in Table 4. We included the forecasting without compensation (α =0) and that with compensation of α =0.9. As seen in Table 4, about 38% improvement can be achieved by the proposed compensation scheme which can be considered a lot in short-term load forecasting. Furthermore, standard deviation of MAPE of STLF is also reduced, implying that we can achieve a more reliable forecasting system by the error compensation.

Table 4. Performance comparison: Load forecasting without compensation and with compensation.

		June	July	Aug.	Sept.	Avg.
α =0	MAPE	1.37	2.04	2.28	1.14	1.71
	Std.	1.25	2.23	1.74	1.07	1.57
α =0.9	MAPE	1.03	1.08	1.05	1.04	1.05
	Std.	0.98	1.15	0.89	0.95	0.99

4. Conclusions

In this paper, we propose a method to improve the performance of an existing NFM based STLF by using the error compensation. The proposed method employs the compensation scheme that adjusts the predicted value of current load according to the prior prediction error. Simulation results reveal that this simple compensation can significantly improve the accuracy performance of the NFM based STLF, suggesting higher possibility of the application of the NFM based STLF in the real world.

References

[1] W. R. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Trans. Power Apparatus and Systems*, vol. 90, pp. 900 – 910, 1971.

[2] M. T. Hagan and S. M. Behr, "The time series approach to short-term load forecast," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-2, no. 3, pp. 785-791, 1987.

[3] S. Rahman and O. Hazim, "A generalized knowledge-based short-term load forecasting technique," *IEEE Trans. Power Syst*., vol. PWRS-8, no. 2, pp. 508-514, 1993.

[4] D. C. Park, M. EL-Sharkawi, R. Marks, A. Atlas, and M. Damborg, "Electrical load forecasting using an artificial neural network," *IEEE Trans. Power Syst*., vol. 6, no. 2, pp. 442-449, May 1991.

[5] T. W. Chow and C. T. Leung, "Neural network based short-term load forecasting using weather compensation," *IEEE Trans. Power Syst*., vol. 11, no. 4, pp. 1736-1742, Nov. 1996.

[6] A. Khotanzad, R. C. Hwang, A. Abaye, and D. Maratukulam, "An adaptive modular artificial neural network hourly load forecaster and its implementation at electric utilities," *IEEE Trans. Power Syst*., vol. 10, no. 3, pp. 1716-1722, Aug. 1995.

[7] K. H. Kim, J. K. Park, K. J. Hwang, and S. H. Kim, "Implementation of hybrid short-term load forecasting system using artificial neural networks and fuzzy expert systems," *IEEE Trans. Power Syst*., vol. 10, no. 3, pp. 1534-1539, Aug. 1995.

[8] A. G. Bakirtzis, J. B. Theocharis, S. J. Kiartzis, and K. J. Satsios, "Short-term load forecasting using fuzzy neural networks," *IEEE Trans. Power Syst*., vol. 10, no. 3, pp. 1518-1524, Aug. 1995.

[9] H. Mori and H. Kobayashi, "Optimal fuzzy inference for short-term load forecasting," *IEEE Trans. Power Syst*., vol. 11, no. 1, pp. 390-396, Feb. 1996.

[10] Y. J. Park and B. H. Wang, "Neuro-Fuzzy Model based Electrical Load Forecasting System: Hourly, Daily, and Weekly Forecasting," *Journal of Korean Fuzzy Logic and Intelligent Systems*, vol. 14, no. 5, pp. 533-538, 2004.

[11] H. J. Shim and B. H. Wang, "Reliability Computation of Neuro-fuzzy Model based Short Term Electrical Load Forecasting," *Journal of Korean Institute of Electrical Engineering*, vol. 54, no. 10, pp. 467-474, 2005.

[12] C. T. Lin and C. S. G. Lee, "Neural-network-based-fuzzy logic control and decision system," *IEEE Trans. Comput*., vol. 40, pp. 1320-1336, 1991.

[13] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern*., vol. 23, pp 665-684, 1995.

[14] C. T. Sun, "Rule-based structure identification in an adaptive-network-based fuzzy inference system," *IEEE Trans. Fuzzy Systems*, vol. 2, pp. 7-31, 1994.

[15] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Systems*, vol. 1, pp. 7-31, 1993.

[16] M. Kubat, "Decision trees can initialize radial basis function networks," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 813-821, Sept. 1998.

[17] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Systems*, vol. 1, pp. 7-31, 1993.

[18] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum Press, New York, 1981.

[19] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[20] P. Wasserman, *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, 1993.

[21] H. Yoo and R. L. Pimmel, "Short-term Load Forecasting using a Self-supervised Adaptive Neural Network," *IEEE Trans. Power Syst*., vol. 14, no. 2, pp. 779-784, May 1999.

Bo-Hyeun Wang
received the B.S. degree in Electrical Engineering from Yonsei University in 1987 and the M.S. and the Ph.D. degrees in Electrical Engineering both from Georgia Institute of Technology in 1990 and 1991, respectively. From 1991 to 1998, he was a Senior Research Scientist at LG Electronics Research Center in Seoul, Korea. Since 1998, he has been a faculty member of the Department of Electrical Engineering at Kangnung National University. His research interests include artificial intelligence and machine learning.

Phone : +82-33-640-2384
Fax : +82-33-640-2244
E-mail : bhw@kangnung.ac.kr