

Short-Term Load Forecasting Via Fuzzy Neural Network With Varied Learning Rates

Rong-Jong Wai¹, Yi-Chang Chen¹ and Yung-Ruei Chang²

¹ Department of Electrical Engineering, Yuan Ze University, Chung Li, Taiwan, R.O.C.

² Institute of Nuclear Energy Research, Atomic Energy Council, Taoyuan, Taiwan, R.O.C.

Abstract—Due to the lack of natural resources, the majority of energy in many countries must depend on import, and the corresponding cost is expensive and affected by international market fluctuation and control. In recent years, an intelligent micro-grid system composed of renewable energy sources is becoming one of the interesting research topics. The forecasting of short-term loads enables the intelligent micro-grid system to manipulate an optimized loading and unloading control by measuring the electrical supply each hour for achieving the best economical and power efficiency. Therefore, this study investigates a fuzzy neural network (FNN) with varied learning rates for the short-term load forecasting (STLF), and compares its better forecasting performance with a conventional neural network (NN) by numerical simulations of a real case in Taiwan campus.

Keywords—Fuzzy neural network, varied learning rates, short-term load forecasting.

I. INTRODUCTION

Nowadays, the research topic of short-term load forecasting (STLF) becomes an important issue for power system operations. In general, the objective of high-precision STLF is difficult to reach due to complex effects on load by a variety of factors. In the past researches, various STLF methods have been investigated [1], including exponential smoothing [2], regression [3], Box-Jenkins models [4], Kalman filter [5], state space model [6], and time series techniques [6]. These frameworks were developed based on statistical methods and proven to work well under normal conditions. However, these methods show some deficiency in the presence of an abrupt change in environment or sociological variables, which are believed to affect load patterns. Besides, the employed techniques for these models in [1]–[6] use a large number of complex relationships and require a long computational time, so that the desired forecasting accuracy can not be achieved.

In recent years, artificial intelligent methods have attracted more attention in application to load forecasting, such as pattern recognition [7], expert system techniques [8]–[10], neural network (NN) [11]–[14] and fuzzy logic system [15], [16]. Amjadi, Daraeepour, and Keynia [11] comprised a modified version of a relief algorithm for the feature selection and a hybrid neural network to forecast the day-ahead electricity price. Bashir and El-Hawary [12] proposed adaptive artificial neural networks with particle

swarm optimization to forecast the hourly load. In the previous literatures [15], [16], fuzzy logic was designed for short-term and midterm load forecasting. Unfortunately, the detailed action behavior in the NN is difficult to handle clearly, and how to build appropriate rules in the fuzzy system is the major problem to be solved. Therefore, the concept of incorporating fuzzy logic into a neural network has grown into a popular research topic [17]. The integrated fuzzy-neural-network (FNN) system possesses the merits of both fuzzy systems (e.g., humanlike IF-THEN rules thinking and ease of incorporating expert knowledge) and neural networks (e.g., learning and optimization abilities, and connectionist structures). By this way, one can bring the low-level learning and computational power of neural networks into fuzzy systems and also high-level, humanlike IF-THEN rule thinking and reasoning of fuzzy systems into neural networks [18]–[20].

This study investigates a FNN forecaster with varied learning rates for the STLF model, and compares its better forecasting performance with a conventional NN forecaster. The back-propagation algorithm is used to train the FNN on line. Moreover, to guarantee the convergence of forecasting error, analytical methods based on a discrete-type Lyapunov function are proposed to determine the varied learning rates of the FNN. This study is organized into five sections. Following the introduction, the network structure of a four-layer FNN is described, and the on-line training algorithm for the FNN with varied learning rates is derived in Section II. Moreover, the proposed STLF model via the FNN forecaster is illustrated in Section III. In Section IV, numerical simulations are given to verify the effectiveness of the proposed strategy in terms of mean average percentage error (MAPE). Finally, some conclusions are drawn in Section V.

II. FUZZY NEURAL NETWORK

A. Description of Network Structure

A general four-layer fuzzy neural network (FNN) as shown in Fig. 1, which is composed of the input (the i layer), membership (the j layer), rule (the k layer) and output layers (the o layer), is adopted to implement the proposed FNN forecaster. Nodes in the input layer represent the input linguistic variables. Nodes in the membership layer act as the membership functions. Moreover, all the nodes in the rule layer form a fuzzy rule base. The signal propagation and the basic function of each layer are introduced as follows.

This work was supported by the Institute of Nuclear Energy Research of Taiwan, R.O.C. through grant number 1002001INER065.

Layer 1: input layer

For every node i in this layer, the net input and the net output can be represented as

$$net_i^1 = x_h^1, \quad y_i^1 = f_i^1(net_i^1) = net_i^1 \quad (1)$$

where x_h^1 represents the i th input to the node of layer 1.

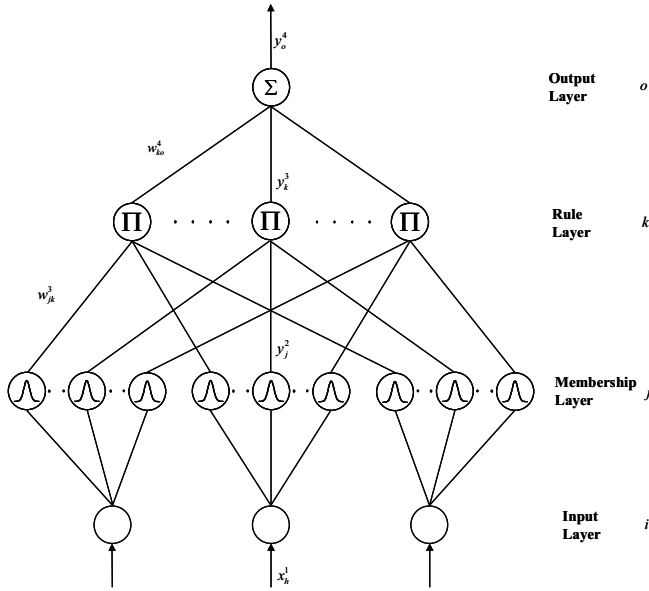


Fig. 1. Structure of four-layer fuzzy neural network (FNN).

Layer 2: membership layer

In this layer, each node performs a membership function. The Gaussian function is adopted as the membership function. For the j th node

$$net_j^2 = -\frac{(x_i^2 - m_{ij})^2}{(\sigma_{ij})^2}, \quad y_j^2 = f_j^2(net_j^2) = \exp(net_j^2) \quad (2)$$

where m_{ij} and σ_{ij} are, respectively, the mean and the standard deviation of the Gaussian function in the j th term of the i th input linguistic variable x_i^2 to the node of layer 2.

Layer 3: rule layer

Each node k in this layer is denoted by Π , which multiplies the incoming signal and outputs the result of product. For the k th rule node

$$net_k^3 = \prod_j w_{jk}^3 x_j^3, \quad y_k^3 = f_k^3(net_k^3) = net_k^3 \quad (3)$$

where x_j^3 represents the j th input to the node of layer 3, and w_{jk}^3 is assumed to be unity.

Layer 4: output layer

The single node o in this layer is labeled as Σ , which computes the overall output as the summation of all incoming signals.

$$net_o^4 = \sum_k w_{ko}^4 x_k^4, \quad y_o^4 = f_o^4(net_o^4) = net_o^4 \quad (4)$$

where the link weight w_{ko}^4 is the output action strength of the o th output associated with the k th rule; x_k^4 represents the k th input to the node of layer 4, and y_o^4 is the output

of the FNN forecaster.

B. On-Line Learning Algorithm

The central part of the learning algorithm for an FNN concerns how to recursively obtain a gradient vector in which each element in the learning algorithm is defined as the derivative of an energy function with respect to a parameter of the network. This is done by means of the chain rule, and the method is generally referred to as the back-propagation learning rule, because the gradient vector is calculated in the direction opposite to the flow of the output of each node [17]. To describe the on-line learning algorithm of the FNN using the supervised gradient decent method, first the energy function E is defined as

$$E = (x_f - x_l)^2 / 2 = e^2 / 2 \quad (5)$$

where x_f and x_l represent the forecasting load and the actual load, and e denotes the output error between the forecasting load and the actual load. Then, the learning algorithm is described as follows.

Layer 4:

The error term to be propagated is given by

$$\delta_o^4 = -\frac{\partial E}{\partial y_o^4} = \left[-\frac{\partial E}{\partial e} \frac{\partial e}{\partial y_o^4} \right] = \left[-\frac{\partial E}{\partial e} \frac{\partial e}{\partial x_l} \frac{\partial x_l}{\partial y_o^4} \right] \quad (6)$$

and the weight is updated by the amount

$$\Delta w_{ko}^4 = -\eta_w \frac{\partial E}{\partial w_{ko}^4} = \left[-\eta_w \frac{\partial E}{\partial y_o^4} \right] \left(\frac{\partial y_o^4}{\partial net_o^4} \frac{\partial net_o^4}{\partial w_{ko}^4} \right) = \eta_w \delta_o^4 x_k^4 \quad (7)$$

The weights of the output layer are updated according to the following equation:

$$w_{ko}^4(N+1) = w_{ko}^4(N) + \Delta w_{ko}^4 \quad (8)$$

where the factor η_w is the learning-rate parameter of the weight, and N denotes the number of iteration.

Layer 3:

Since the weights in this layer are unity, only the error term needs to be calculated and propagated.

$$\delta_k^3 = -\frac{\partial E}{\partial net_k^3} = \left[-\frac{\partial E}{\partial y_o^4} \right] \left(\frac{\partial y_o^4}{\partial net_o^4} \frac{\partial net_o^4}{\partial y_k^3} \frac{\partial y_k^3}{\partial net_k^3} \right) = \delta_o^4 w_{ko}^4 \quad (9)$$

Layer 2:

The multiplication operation is done in this layer. The error term is computed as follows:

$$\begin{aligned} \delta_j^2 &= -\frac{\partial E}{\partial net_j^2} \\ &= \left[-\frac{\partial E}{\partial y_o^4} \frac{\partial y_o^4}{\partial net_o^4} \frac{\partial net_o^4}{\partial y_k^3} \frac{\partial y_k^3}{\partial net_k^3} \right] \left[\frac{\partial net_k^3}{\partial y_j^2} \frac{\partial y_j^2}{\partial net_j^2} \right] = \sum_k \delta_k^3 y_k^3 \end{aligned} \quad (10)$$

Moreover, the variations of m_{ij} and σ_{ij} can be expressed as

$$\Delta m_{ij} = -\eta_m \frac{\partial E}{\partial m_{ij}} = \eta_m \delta_j^2 \frac{2(x_i^2 - m_{ij})}{(\sigma_{ij})^2} \quad (11)$$

$$\Delta \sigma_{ij} = -\eta_\sigma \frac{\partial E}{\partial \sigma_{ij}} = \eta_\sigma \delta_j^2 \frac{2(x_i^2 - m_{ij})^2}{(\sigma_{ij})^3} \quad (12)$$

where η_m and η_σ are the learning-rate parameters of the mean and the standard deviation of the Gaussian function. The mean and standard deviation of the membership layer are updated as follows:

$$m_{ij}(N+1) = m_{ij}(N) + \Delta m_{ij} \quad (13)$$

$$\sigma_{ij}(N+1) = \sigma_{ij}(N) + \Delta \sigma_{ij} \quad (14)$$

C. Convergence Analyses

Selection of the values for the learning-rate parameters has a significant effect on the network performance. In order to train the FNN effectively, three varied learning rates, which guarantee convergence of the output error based on the analyses of a discrete-type Lyapunov function, are derived in this section. The convergence analyses in this study are to derive specific learning-rate parameters for specific types of network parameters to assure convergence of the output error.

Theorem 1: Let η_w be the learning-rate parameter of the FNN weights and let P_{wmax} be defined as $P_{wmax} \equiv \max_N \|P_w(N)\|$, where $P_w(N) = \partial y_o^4 / \partial w_{ko}^4$ and $\|\cdot\|$ is the Euclidean norm in $\Re^n$. The convergence is guaranteed if η_w is chosen as $\eta_w = \lambda / (P_{wmax}^2) = \lambda / R_u$, in which λ is a positive constant gain, and R_u is the number of rules in the FNN.

Proof: Since

$$P_w(N) = \frac{\partial y_o^4}{\partial w_{ko}^4} = x_k^4 \quad (15)$$

Thus

$$\|P_w(N)\| < \sqrt{R_u} \quad (16)$$

Then, a discrete-type Lyapunov function is selected as

$$V(N) = e^2(N) / 2 \quad (17)$$

The change in the Lyapunov function can be obtained by

$$\Delta V(N) = V(N+1) - V(N) = [e^2(N+1) - e^2(N)] / 2 \quad (18)$$

The error difference can be represented by

$$e(N+1) = e(N) + \Delta e(N) = e(N) + \left[\frac{\partial e(N)}{\partial w_{ko}^4} \right]^T \Delta w_{ko}^4 \quad (19)$$

where Δw_{ko}^4 represents a weight change in the output layer.

By using (6), (7) and (19), then

$$\frac{\partial e(N)}{\partial w_{ko}^4} = \frac{\partial e(N)}{\partial y_o^4} \frac{\partial y_o^4}{\partial w_{ko}^4} = -\frac{\delta_o^4}{e(N)} P_w(N) \quad (20)$$

$$e(N+1) = e(N) - \left[\frac{\delta_o^4}{e(N)} P_w(N) \right]^T \eta_w \delta_o^4 P_w(N) \quad (21)$$

Then

$$\|e(N+1)\| \leq \|e(N)\| \left\| 1 - \eta_w \left(\frac{\delta_o^4}{e(N)} \right)^2 P_w^T(N) P_w(N) \right\| \quad (22)$$

If η_w is chosen as $\eta_w = \lambda / (P_{wmax}^2) = \lambda / R_u$, the term

$\left\| 1 - \eta_w \left(\frac{\delta_o^4}{e(N)} \right)^2 P_w^T(N) P_w(N) \right\|$ in (22) is less than 1.

Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ is guaranteed. The output error between the forecasting load and the actual load will converge to zero as $t \rightarrow \infty$. This completes the proof of Theorem 1.

In order to prove Theorem 2, the following lemmas [18] will be used.

Lemma 1 : Let $p(z) = z \exp(-z^2)$. Then $|p(z)| < 1$, $\forall z \in \Re$.

Lemma 2 : Let $q(z) = z^2 \exp(-z^2)$. Then $|q(z)| < 1$, $\forall z \in \Re$.

Theorem 2: Let η_m and η_σ be the learning-rate parameters of the mean and the standard deviation of the Gaussian function for the FNN; let P_{mmax} be defined as

$P_{mmax} \equiv \max_N \|P_m(N)\|$, where $P_m(N) = \partial y_o^4 / \partial m_{ij}$; let $P_{\sigma max}$ be defined as $P_{\sigma max} \equiv \max_N \|P_\sigma(N)\|$, where $P_\sigma(N) = \partial y_o^4 / \partial \sigma_{ij}$; $\|\cdot\|$ is the Euclidean norm in $\Re^n$. The convergence is guaranteed if η_m and η_σ is chosen as $\eta_m = \eta_\sigma = \eta_w [w_{ko max}^4 (2/\sigma_{ij min})]^{-2}$, in which $\eta_w = \lambda / R_u$; λ is a positive constant gain; $w_{ko max}^4 = \max_N [w_{ko}^4(N)]$; $\sigma_{ij min} = \max_N [\sigma_{ij}(N)]$; $|\cdot|$ is the absolute value.

Proof: According to lemma 1,

$\left| [(x_i^2 - m_{ij}) / \sigma_{ij}] \exp\{-(x_i^2 - m_{ij}) / \sigma_{ij}\} \right| < 1$. Since

$$\begin{aligned} P_m(N) &= \frac{\partial y_o^4(N)}{\partial m_{ij}} = w_{ko}^4 \frac{\partial y_j^2}{\partial m_{ij}} \\ &< w_{ko}^4 \left\{ \max \left(\frac{2}{\sigma_{ij}} \right) \right\} = w_{ko}^4 \left(\frac{2}{\sigma_{ij min}} \right) \end{aligned} \quad (23)$$

Thus,

$$\|P_m(N)\| < \sqrt{R_u} w_{ko max}^4 (2/\sigma_{ij min}) \quad (24)$$

The error difference can also be represented by

$$e(N+1) = e(N) + \Delta e(N) = e(N) + \left[\frac{\partial e(N)}{\partial m_{ij}} \right]^T \Delta m_{ij} \quad (25)$$

where Δm_{ij} represents a change of the mean of the Gaussian function in the membership layer. By using (6), (11) and (25), then

$$\frac{\partial e(N)}{\partial m_{ij}} = \frac{\partial e(N)}{\partial y_o^4} \frac{\partial y_o^4}{\partial m_{ij}} = -\frac{\delta_o^4}{e(N)} P_m(N) \quad (26)$$

$$e(N+1) = e(N) - \left[\frac{\delta_o^4}{e(N)} P_m(N) \right]^T \eta_m \delta_o^4 P_m(N) \quad (27)$$

Then

$$\|e(N+1)\| \leq \|e(N)\| \left\| 1 - \eta_m \left(\frac{\delta_o^4}{e(N)} \right)^2 P_m^T(N) P_m(N) \right\| \quad (28)$$

If η_m is chosen as $\eta_m = \lambda/R_u [w_{ko\max}^4 (2/\sigma_{ij\min})]^{-2} = \eta_w [w_{ko\max}^4 (2/\sigma_{ij\min})]^{-2}$, $\|1 - \eta_m (\delta_o^4/e(N))^2 P_m^T(N) P_m(N)\|$ in (28) is less than 1. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ by (17) and (18) is guaranteed. The output error between the forecasting load and the actual load will converge to zero as $t \rightarrow \infty$.

According to lemma 2, $|(x_i^2 - m_{ij})/\sigma_{ij}|^2 \exp\{-(x_i^2 - m_{ij})/\sigma_{ij}\} < 1$. Since

$$P_\sigma(N) = \frac{\partial y_o^4(N)}{\partial \sigma_{ij}} = w_{ko}^4 \frac{\partial y_j^2}{\partial \sigma_{ij}} \quad (29)$$

$$< w_{ko}^4 \left\{ \max \left(\frac{2}{\sigma_{ij}} \right) \right\} = w_{ko}^4 \left(\frac{2}{\sigma_{ij\min}} \right)$$

Thus,

$$\|P_\sigma(N)\| < \sqrt{R_u} [w_{ko\max}^4 (2/\sigma_{ij\min})] \quad (30)$$

The error difference can also be represented by

$$e(N+1) = e(N) + \Delta e(N) = e(N) + \left[\frac{\partial e(N)}{\partial \sigma_{ij}} \right]^T \Delta \sigma_{ij} \quad (31)$$

where $\Delta \sigma_{ij}$ represents a standard deviation change of the Gaussian function in membership layer. By using (6), (12) and (31), then

$$\frac{\partial e(N)}{\partial \sigma_{ij}} = \frac{\partial e(N)}{\partial y_o^4} \frac{\partial y_o^4}{\partial \sigma_{ij}} = -\frac{\delta_o^4}{e(N)} P_\sigma(N) \quad (32)$$

$$e(N+1) = e(N) - \left[\frac{\delta_o^4}{e(N)} P_\sigma(N) \right]^T \eta_\sigma \delta_o^4 P_\sigma(N) \quad (33)$$

Then

$$\|e(N+1)\| \leq \|e(N)\| \left\| 1 - \eta_\sigma \left(\frac{\delta_o^4}{e(N)} \right)^2 P_\sigma^T(N) P_\sigma(N) \right\| \quad (34)$$

If η_σ is chosen as $\eta_\sigma = \lambda/R_u [w_{ko\max}^4 (2/\sigma_{ij\min})]^{-2} = \eta_w [w_{ko\max}^4 (2/\sigma_{ij\min})]^{-2}$, $\|1 - \eta_\sigma (\delta_o^4/e(N))^2 P_\sigma^T(N) P_\sigma(N)\|$ in (34) is less than 1. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ by (17) and (18) is guaranteed. The output error between the forecasting load and the actual load will converge to zero as $t \rightarrow \infty$. This completes the proof of Theorem 2.

III. SHORT-TERM LOAD FORECASTING MODEL

In general, the historical load data are usually used as the input for the short-term load forecasting (STLF) model. However, how to properly select characteristic data is a key question. To forecast the load at the next hour, the common practice is to use the most recently available load data, i.e., the yesterday load on the same hour. In the past research, the load data at the closed hour was selected to be the input for the STLF model. But, this selection method is not enough to completely appear the features for the forecasting hour. In order to improve the forecasting performance, a similar hour method is used to select the input variable for

the STLF model.

The load patterns for weekdays are almost similar while those for weekends are different from weekdays due to social activities. Therefore, load information data for an entire week are considered. Moreover, all hours for a day will be considered because the daytime and night have different load features. In addition, a sudden load variation has a large influence for the forecasting performance. In order to prevent decreasing the forecasting accuracy from the sudden load variation, the data at the closed hours have to be considered.

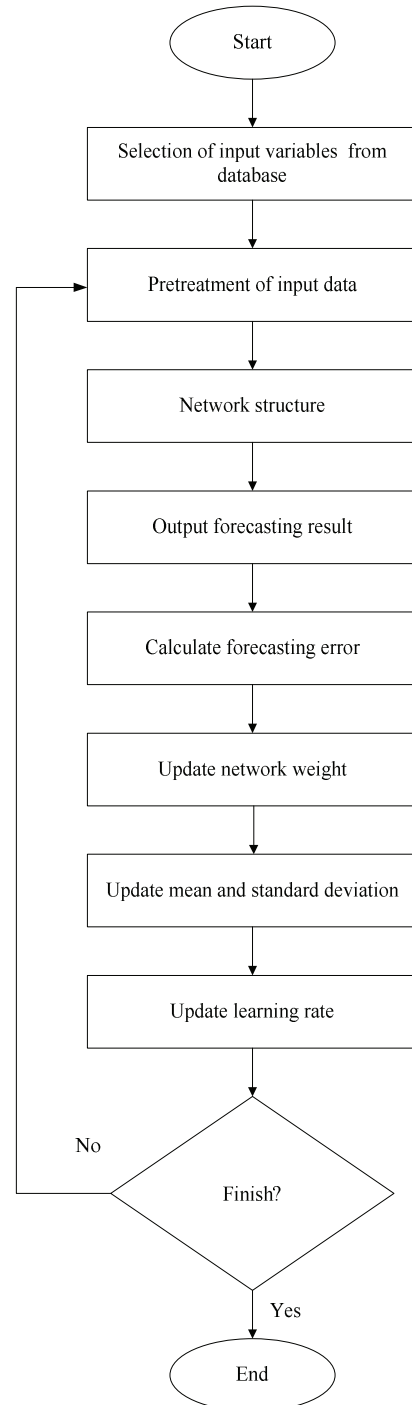


Fig. 2. Flowchart of neural-network-based short-term load forecasting (STLF).

According to the aforementioned consideration, a similar hour-based fuzzy neural network (FNN) forecaster as shown in Fig. 1 is designed to forecast the next hour's load. The flowchart of the proposed STLFL model is depicted in Fig. 2. Three major factors including the load data at the same hour with the forecasting hour for yesterday, the load data at the same hour with the forecasting hour for the last week, and the load data at the closed hour with the forecasting hour are considered for the FNN forecaster. It will be helpful for simplifying the FNN structure and reducing the computation burden. In the proposed STLFL model, the pretreatment of input data is to use the corresponding change rate in place of real data for normalizing input signals.

In this study, there are three input signals including the load data at the same hour with the forecasting hour for yesterday, and the load data at the same hour with the forecasting hour for the last two week, and one output signal for the FNN forecaster.

$$x_1^1 = [x_i(t-24) - x_i(t-25)] / x_i(t-25) \quad (35)$$

$$x_2^1 = [x_i(t-168) - x_i(t-169)] / x_i(t-169) \quad (36)$$

$$x_3^1 = [x_i(t-336) - x_i(t-337)] / x_i(t-337) \quad (37)$$

where t is the present hour. Because the main target in the proposed STLFL model is to forecast the change rate between $x_i(t-1)$ and $x_i(t)$, the forecasting load $x_f(t)$ at the next hour can be represented as

$$x_f(t) = (1 + y_1^4) x_i(t-1) \quad (38)$$

The output error between $x_f(t)$ and $x_i(t)$ is continuously calculated to update the weights, the means, the standard deviations and the learning rates of the FNN forecaster until this program is finished. The effectiveness of the proposed STLFL model is verified by the following numerical simulations.

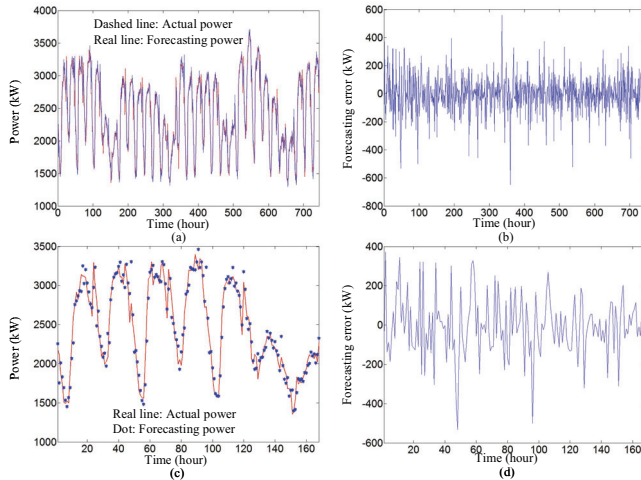


Fig. 3. Numerical simulations of STLFL model with FNN forecaster in March: (a) Month forecasting performance; (b) Month forecasting error; (c) Week forecasting performance; (d) Week Forecasting error.

IV. NUMERICAL SIMULATIONS

Numerical simulations of the proposed short-term load forecasting (STLF) model are implemented via the

MATLAB software based on the schemes as shown in Figs. 1 and 2. In this study, the following mean absolute percentage error (MAPE) is adopted to measure the forecasting performance:

$$MAPE = \frac{1}{M} \sum_{t=1}^M \left[\frac{|x_f(t) - x_i(t)|}{x_i(t)} \right] \times 100\% \quad (39)$$

where M is the number of forecasting loads. In the numerical simulations, real data came from the energy management system in Yuan Ze University, Taiwan is used for examining the forecasting performance. In the fuzzy neural network (FNN) forecaster, there are three inputs in the input layer, nine neurons in the membership layer, twenty-seven neurons in the rule layer, one output in the output layer, and the learning rates are tuned according to Theorems 1 and 2 in Section II-C. For exhibiting the superiority of the proposed FNN forecaster with varied learning rates, a conventional neural network (NN) with the back-propagation learning algorithm is also examined here. In the NN forecaster, there are three inputs in the input layer, nine neurons in the hidden layer, one output in the output layer, and the learning rates are fixed at 0.5. The input and output signals of the NN forecaster are the same as the FNN forecaster as shown in (35)–(38).

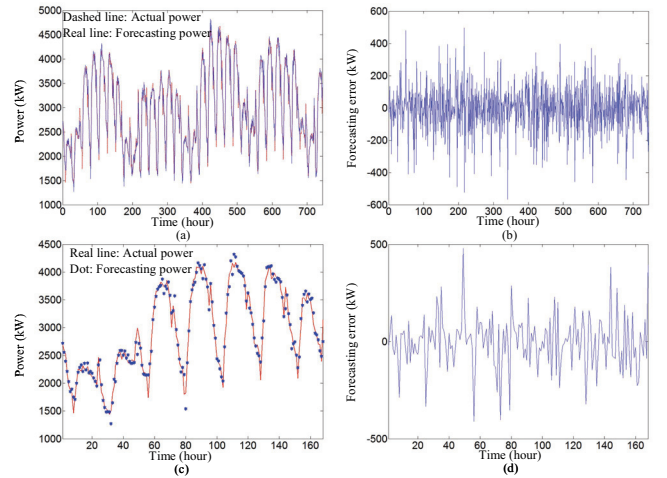


Fig. 4. Numerical simulations of STLFL model with FNN forecaster in May: (a) Month forecasting performance; (b) Month forecasting error; (c) Week forecasting performance; (d) Week Forecasting error.

According to season factor and campus environment, the load data at four months (March, May, October and November in 2010) are selected. Numerical illustrations of the STLFL model with the proposed FNN forecaster are depicted in Figs. 3–6. For all illustrations given in Figs. 3–6, the sub-figure (a) is the forecasting performance for a month; the sub-figure (b) is the forecasting error for a month; the sub-figure (c) is the forecasting performance for a week; the sub-figure (d) is the forecasting error for a week. The corresponding MAPE records of the STLFL model with the conventional NN forecaster and the proposed FNN forecaster for different months are summarized in Table I. From these records, the MAPE value of the proposed STLFL model with the FNN forecaster at different testing months is lower than 4.1%, and the average improvement rate in comparison with the conventional NN forecaster is 13.2%.

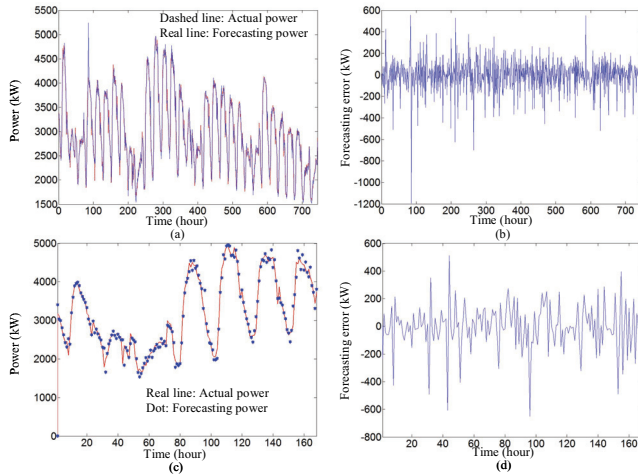


Fig. 5. Numerical simulations of STLF model with FNN forecaster in October: (a) Month forecasting performance; (b) Month forecasting error; (c) Week forecasting performance; (d) Week Forecasting error.

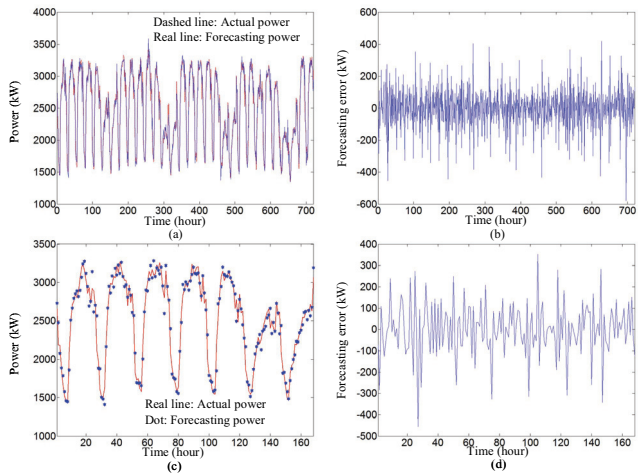


Fig. 6. Numerical simulations of STLF model with FNN forecaster in November: (a) Month forecasting performance; (b) Month forecasting error; (c) Week forecasting performance; (d) Week Forecasting error.

TABLE I. MAPE OF FORECASTING RECORDS FOR DIFFERENT MONTHS

Network structure	March	May	October	November
Fuzzy neural network	3.97%	4.04%	3.69%	3.98%
Neural network	4.8%	4.38%	4.41%	4.48%

V. CONCLUSIONS

This study has successfully developed a short-term load forecasting (STLF) model with a fuzzy neural network (FNN), and applied well to a real case in Taiwan campus. Due to the selection of similar hour data, the network structure can be simplified, and the computation time can be shortened. In order to guarantee the convergence of forecasting error, analytical methods based on a discrete-type Lyapunov function are proposed to determine varied learning rates for the on-line learning algorithm of the FNN forecaster. According to simulated results, the proposed STLF model with the FNN forecaster indeed yields better performance than the one with the conventional NN forecaster, and the average improvement

rate is 13.2%. In the future, the proposed STLF model could be modified and extended to midterm and long-term load forecasting.

REFERENCES

- [1] N. Amjady and F. Keynia, "Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm," *Energy*, vol. 34, no. 1, pp. 46–57, Jan. 2009.
- [2] B. Billah, M. L. King, R. D. Snyder, and A. B. Koehler, "Exponential smoothing model selection for forecasting," *Int. J. Forecast.*, vol. 22, no. 2, pp. 239–247, June 2006.
- [3] E. E. Elattar, J. Y. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Trans. Syst., Man, Cybern.*, vol. 40, no. 4, pp. 438–447, Jul. 2010.
- [4] Y. Lu and S. M. AbouRizk, "Automated Box – Jenkins forecasting modelling," *Automation in Construction*, vol. 18, no. 5, pp. 547–558, Aug. 2009.
- [5] S. C. Huang, "Online option price forecasting by using unscented Kalman filters and support vector machines," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2819–2825, May 2008.
- [6] M. Shahidehpour, H. Yamin, and Z. Li, *Market Operations in Electric Power Systems*. New York: John Wiley & Sons, 2002.
- [7] A. S. Dehdashti, J. R. Tudor, and M. C. Smith, "Forecasting of hourly load by pattern recognition-a deterministic approach," *IEEE Trans Power Syst.*, vol. 101, no. 9, pp. 3290–3294, Sept. 1982.
- [8] S. Rahman and R. Bhatnagar, "An expert system based algorithm for short term load forecast," *IEEE Trans Power Syst.*, vol. 3, no. 2, pp. 392–399, May 1988.
- [9] S. Rahman and O. Hazim, "Generalized knowledge-based short-term load forecasting technique," *IEEE Trans Power Syst.*, vol. 8, no. 2, pp. 508–514, May 1993.
- [10] K. L. Ho, Y. Y. Hsu, C. F. Chen, T. E. Lee, C. C. Liang, T. S. Lai, and K. K. Chen, "Short term load forecasting of Taiwan power system using a knowledge-based expert system," *IEEE Trans Power Syst.*, vol. 5, no. 4, pp. 1214–1221, Nov. 1990.
- [11] N. Amjady, A. Daraeepour, and F. Keynia, "Day-ahead electricity price forecasting by modified relief algorithm and hybrid neural network," *IET Gener. Transm. Distrib.*, vol. 4, no. 3, pp. 432–444, July 2010.
- [12] Z. A. Bashir and M. E. El-Hawary, "Applying wavelets to short-term load forecasting using PSO-based neural networks," *IEEE Trans Power Syst.*, vol. 24, no. 1, pp. 20–27, Feb. 2009.
- [13] E. González-Romera, M. A. Jaramillo-Morán, and D. Carmona-Fernández, "Monthly electric energy demand forecasting based on trend extraction," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1946–1953, Nov. 2006.
- [14] P. C. Chang, C. Y. Fan, and C. H. Liu, "Integrating a piecewise linear representation method and a neural network model for stock trading points prediction," *IEEE Trans. Syst., Man, and Cybern.*, vol. 39, no. 1, pp. 80–92, Jan. 2009.
- [15] V. H. Hinojosa and A. Hoesse, "Short-term load forecasting using fuzzy inductive reasoning and evolutionary algorithms," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 565–574, Feb. 2010.
- [16] N. E. Charalambos and D. H. Nikos, "An annual midterm energy forecasting model using fuzzy logic," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 469–478, Feb. 2009.
- [17] C. T. Lin and C. S. George Lee, *Neural Fuzzy Systems*. New Jersey: Prentice-Hall, 1996.
- [18] F. J. Lin, R. J. Wai, and C. C. Lee, "Fuzzy neural network position controller for ultrasonic motor drive using push-pull DC-DC converter," *IEE Proc. Control Theory Appl.*, vol. 146, no. 1, pp. 99–107, Jan. 1999.
- [19] R. J. Wai, H. H. Lin, and F. J. Lin, "Hybrid controller using fuzzy neural networks for identification and control of induction servo motor drive," *Neurocomputing*, vol. 35, no. 4, pp. 91–112, Nov. 2000.
- [20] F. J. Lin and R. J. Wai, "Adaptive fuzzy-neural-network control for induction spindle motor drive," *IEEE Trans. Energy Conversion*, vol. 17, no. 4, pp. 507–513, Dec. 2002.