A Neural Fuzzy Network with Optimal Number of Rules for Short-Term Load Forecasting in an Intelligent Home

S.H. Ling, H.K. Lam, F.H.F. Leung and P.K.S. Tam
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong

Abstract

In this paper, a short-term home daily load forecasting realized by a neural fuzzy network (NFN) and an improved genetic algorithm (GA) is proposed. It can forecast the daily load accurately with respect to different day types and weather information. It will also be shown that the improved GA performs better than the traditional GA on some benchmark test functions. By introducing switches in the links of the neural fuzzy network, the optimal network structure can be found by the improved GA. The membership functions and the number of rules of the neural fuzzy network can be generated automatically. Simulation results for a short-term daily load forecasting in an intelligent home will be given.

I. Introduction

Modern homes should have smart features to ensure a higher degree of home security, entertainment and comfort. To realize these features, reliable channels for the communication among electrical appliances and users should be present. Appliances should be used in an efficient way to reduce the wastage of energy. This paper is based on an intelligent home system [3]. In this system, the AC power line network is used not only for supplying electrical power, but also serving as the data communication channel for electrical appliances. With this AC power line data network, a short-term load forecasting can be realized. An accurate load forecasting can bring the following benefits to the intelligent home: 1) Increasing the reliability [4] of the AC power line data network, and 2) Optimal load scheduling.

Computational intelligence techniques have been applied in load forecasting. Artificial neural networks have been considered as a very promising approach to short-term load forecasting [5-6], but its slow convergence time and poor ability of processing linguistic information may cause some problems. In recent year, fuzzy logic has been used to deal with variable linguistic information in load forecasting [7]. By processing fuzzy information, reasoning with respect to a linguistic knowledge base can be done. In [5-6], gradient-descent (GD) algorithm was used to train the neural network parameters. However, the common problems of convergence to local minima and sensitivity to initial values persist. Global search technique such as Genetic Algorithm (GA) [1] may solve these problems. The contributions of this paper are five-fold. First, we develop a neural fuzzy system with the improved GA for short-term daily load forecasting in an intelligent home. Simulation results will be given. Second, new genetic operators are introduced in the improved GA. It will be shown that the improved GA performs better than the traditional GA based on the benchmark De Jong's test functions [2]. Third, the improved GA is implemented in floating-point numbers; hence, the processing time is shorter than that of the traditional GA. Fourth, the improved GA needs only one user-input parameter (population size), instead of three, for its implementation. This makes the improved GA simple and easy to use, especially for the users who do not have too much knowledge on tuning. Fifth, a neural fuzzy network (NFN) with switches is proposed. By using the improved GA, the optimal number of fuzzy rules can be found.

II. Improved Genetic Algorithm

Genetic algorithms (GAs) [1] are powerful searching algorithms to handle optimization problems. In this paper, the traditional GA is modified and new genetic operators are introduced to improve its performance. The probabilities of crossover and mutation in the traditional GA are no longer needed. Only the population size has to be defined. The improved GA process is shown in Fig. 1.

A. Initial Population

The initial population is a potential solution set P. The first set of population is usually generated randomly.

$$P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\}, \quad (1)$$

$$\mathbf{p}_i = \begin{bmatrix} p_{i_1} & p_{i_2} & \cdots & p_{i_j} & \cdots & p_{i_{no\_vars}} \end{bmatrix}, \quad (2)$$

$$para_{\min}^{j} \le p_{i_j} \le para_{\max}^{j},\ i = 1, \dots, pop\_size;\ j = 1, \dots, no\_vars \quad (3)$$

where *pop_size* denotes the population size; *no_vars* denotes the number of variables to be tuned; p_{i_j} are the parameters to be tuned; $para_{\min}^{j}$ and $para_{\max}^{j}$ are the minimum and maximum values of the parameter p_{i_j}. It can be seen from (1) to (3) that the potential solution set P contains some candidate solutions $\mathbf{p}_i$ (chromosomes). The chromosome $\mathbf{p}_i$ contains some variables p_{i_j} (genes).

B. Evaluation

Each chromosome in the population will be evaluated by a defined fitness function. The better chromosomes will return higher values in this process. The fitness function to evaluate a chromosome in the population can be written as,

$$fitness = f(\mathbf{p}_i) \quad (4)$$

The form of the fitness function depends on the application.

C. Selection

Two chromosomes in the population will be selected to undergo genetic operations for reproduction. The chromosome having a higher fitness value should have a higher chance to be selected. The selection can be done by assigning a probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i = f(\mathbf{p}_i) \Big/ \sum_{j=1}^{pop\_size} f(\mathbf{p}_j),\ i = 1, 2, \ldots, pop\_size \quad (5)$$

The cumulative probability $\hat{q}_i$ for $\mathbf{p}_i$ is defined as,

$$\hat{q}_i = \sum_{j=1}^{i} q_j,\ i = 1, 2, \ldots, pop\_size \quad (6)$$

The selection process randomly generates a nonzero floating-point number, $d \in [0\ \ 1]$, for each chromosome. Then, the chromosome $\mathbf{p}_i$ is chosen if $\hat{q}_{i-1} < d \le \hat{q}_i$, $i = 1,2, \ldots, pop\_size$, and $\hat{q}_0 = 0$. It can be observed from this selection process that a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected.

D. Genetic Operations

The genetic operations are the averaging and the mutation operations. If two selected chromosomes are $\mathbf{p}_1$ and $\mathbf{p}_2$, the offspring generated by the averaging process is given by,

$$\mathbf{os} = [os_1\ \ os_2\ \ \cdots\ \ os_{no\_vars}] = (\mathbf{p}_1 + \mathbf{p}_2)/2 \quad (7)$$

This offspring (7) will then undergo the mutation operation. Three new offspring will be generated:

$$\mathbf{nos}_j = [os_1^j\ \ os_2^j\ \ \cdots\ \ os_{no\_vars}^j] + [b_1\Delta nos_1\ \ b_2\Delta nos_2\ \ \cdots\ \ b_{no\_vars}\Delta nos_{no\_vars}],\ j = 1, 2, 3 \quad (8)$$

where b_i, $i = 1, 2, \ldots, no\_vars$, can only take the value of 0 or 1, Δnos_i, $i = 1, 2, \ldots, no\_vars$, are randomly generated floating-point numbers with $para^i_{\min} \le os^j_i + \Delta nos_i \le para^i_{\max}$. The first new offspring ($j = 1$) has only one $b_i = 1$ (i being randomly generated within the range) and all the others are zeros. The second new offspring has some b_i randomly set to 1 and others are zeros. The third new offspring has all $b_i = 1$. These three new offspring will then be evaluated using the fitness function of (4). The one with the largest fitness value f_l will replace the chromosome with the smallest fitness value f_s in the population if $f_l > f_s$. After the operation of selection, averaging, and mutation, a new population is generated. This new population will repeat the same process. Such an iterative process can be terminated when the result reaches a defined condition, e.g., the change of the fitness values between the current and the previous iteration is less than 0.001.

III. Benchmark Test Functions

De Jong's Test Functions [2] are used as the benchmark test functions to examine the applicability and efficiency of the improved GA. Five test functions, $f_i(\mathbf{x})$, $i = 1, 2, 3, 4, 5$, will be used, where $\mathbf{x} = [x_1\ \ x_1\ \ \cdots\ \ x_{no\_x}]$. no_x is an integer denoting the dimension of the vector $\mathbf{x}$.
The five test functions are defined as follows,

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2,\ -5.12 \le x_i \le 5.12 \quad (9)$$

where $n = 3$ and the minimum point is at $f_1(0, 0, 0) = 0$

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1}\left(100\times\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right),\ -2.048 \le x_i \le 2.048 \quad (10)$$

where $n = 2$ and the minimum point is at $f_2(0, 0) = 0$.

$$f_3(\mathbf{x}) = 6\times n + \sum_{i=1}^{n} floor(x_i),\ -5.12 \le x_i \le 5.12 \quad (11)$$

where $n = 5$ and the minimum point is at $f_3([-5.12, -5], \ldots, [-5.12, -5]) = 0$.
The floor function, $floor(\cdot)$, is to round down the argument to an integer.

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} i\times x_i^4 + Gauss(0,1),\ -1.28 \le x_i \le 1.28 \quad (12)$$

where $n = 30$ and the minimum point is at $f_4(0, \ldots, 0) = 0$.
The $Gauss(0, 1)$ is a function to generate uniformly a random floating-point number between 0 and 1 inclusively.

$$f_5(\mathbf{x}) = \frac{1}{k} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\quad -65.356 \le x_i \le 65.356 \quad (13)$$

where $\{a_{ij}\} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & \cdots & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & \cdots & \cdots & 32 & 32 & 32 \end{bmatrix}$

$k = 500$ and the minimum point is at $f_5(-32,-32) \approx 1$.
The fitness function for f_1 to f_4 is defined as,

$$fitness = 1/1 + f_i(\mathbf{x}),\ i = 1, 2, 3, 4. \quad (14)$$

and the fitness function for f_5 is defined as,

$$fitness = 1/f_5(\mathbf{x}) \quad (15)$$

The improved GA goes through these 5 test functions. The results are compared with those obtained by the traditional GA [1]. Each simulation takes 500 iterations and the population size is 20. Each parameter of the traditional GA is encoded into a 40-bit chromosome and the probabilities of crossover and mutation are 0.25 and 0.03 respectively. For tests 1 to 5, the initial value are $[1\ \ 1\ \ 1]$, $[0.5\ \ 0.5]$, $[1\ \ \cdots\ \ 1]$, $[0.5\ \ \cdots\ \ 0.5]$ and $[10\ \ \cdots\ \ 10]$ respectively. The results of the average fitness values over 30 times of simulations are tabulated in Table I. It can be seen that the performance of the improved GA is better than that of the traditional GA. From Table I, the processing time of the improved GA is much shorter than that of the traditional GA.

IV. Tuning Membership Functions and Rules

A. Neural Fuzzy Network with Rule Switches

We use a fuzzy associative memory (FAM) [8] rule base type for the NFN. For an NFN, the number of possible rules may be too large. This makes the network complex while some rules may not be necessary. Thus, an NFN is proposed which can have an optimal number of rules and membership functions. A unit step function is introduced to each rule:

$$\delta(\varsigma) = \begin{cases} 0 \text{ if } \varsigma \le 0 \\ 1 \text{ if } \varsigma > 0 \end{cases},\ \varsigma \in \Re \quad (16)$$

This is equivalent to adding a switch to each rule in the NFN. Referring to Fig. 2, we define the input and output variables as x_i, $i = 1, 2, \ldots, n$, and y, respectively; where n is the number of input variables. The behaviour of the NFN is governed by p fuzzy rules in the following format;

R_g: IF $x_1(t)$ is $A_{1g_1}(x_1(t))$ AND $x_2(t)$ is $A_{2g_2}(x_2(t))$ AND … AND $x_n(t)$ is $A_{ng_n}(x_n(t))$
THEN $y(t)$ is w_g, $t = 1, 2, \ldots, u$ (17)

where u is the number of input-output data pairs; g = 1, 2, ..., p, is the rule number; w_g is the output singleton of rule g.

$$p=\prod_{i=1}^{n} m_i \tag{18}$$

where m_i is the number of membership function of input variable x_i and $g_i \in [1,\ldots,m_i]$, $i=1,\ldots,n$.

The membership function is bell-shaped as given by,

$$A_{ig_i}(x_i(t))=e^{\frac{-(x_i(t)-\bar{x}_{ig_i})^2}{2\sigma_{ig_i}^2}} \tag{19}$$

where parameter $\bar{x}_{ig_i}$ and σ_{ig_i} are the mean value and the standard deviation of the membership function respectively. The grade of the membership of each rule is defined as,

$$\mu_g(t) = A_{1g_1}(x_1(t)) \cdot A_{2g_2}(x_2(t)) \cdot \ldots \cdot A_{ng_n}(x_n(t)) \tag{20}$$

The output of the neural fuzzy network $y(t)$ is defined as,

$$y(t)=\sum_{g=1}^{p} \mu_g(t) w_g \delta(\varsigma_g) \Big/ \sum_{g=1}^{p} \mu_g(t) \tag{21}$$

where ς_g denotes the rule switch parameter of the g-th rule.

B. Tuning

The proposed NFN can be employed to learn a multi-input-single-output relationship in an application using the improved GA. The desired input-output relationship is described by,

$$y^d(t)=\mathbf{q}(\mathbf{z}^d(t)),\ t = 1, 2, \ldots, u \tag{22}$$

where $y^d(t)$ is the desired output for the input $\mathbf{z}^d(t)=[z_1^d(t)\ \ z_2^d(t)\ \cdots\ z_v^d(t)]$ and $\mathbf{q}(\cdot)$ is an unknown non-linear function. The fitness function is defined as,

$$fitness = 1/1+err \tag{23}$$

where $$err = 1/u \sum_{t=1}^{u} \left| y^d(t) - y(t) \right| / y^d(t) \tag{24}$$

The objective is to minimize the value of (24) using the improved GA by setting the chromosome to be $[\bar{x}_{ig_i}\ \sigma_{ig_i}\ \varsigma_g]$ for all i, g_i, g. The range of fitness in (23) is [0,1]. A larger value of the *fitness* indicates a smaller *err*. Thus, an optimal neural fuzzy network in terms of the number of rules and the membership functions can be obtained.

V. Short-Term Load Forecasting System

It is desired to forecast the load demand in a home with respect to the weekday's type number and the hour number. The load forecasting system involves 168 multi-input-single-output NFNs, one for a given weekday's type number and an hour number ($7 \times 24 = 168$). The most important task in the short-term load-forecasting problem is to select the input variables. We use three types of input variables: *Historical load data* – the hourly load values that represent the power consumption habit of the family; *Temperature inputs* – the average temperature at the previous day and the present day; *Rainfall index inputs* – the average rainfall index at previous day and the present day. The range of the rainfall index is set between 0 and 1. 0 represents no rain and 1 represents heavy rain. One of the 168 proposed NFN for daily load forecasting is shown in Fig. 3. It is a 7-input-1-output network with rule switches. The inputs, z_i, of the proposed NFN are: $z_1 = L^d(d-1,h-1)$ which represents the load value at the previous hour of the previous day, $z_2 = L^d(d-1,h)$ which represents the load value at the forecasting hour of the previous day, $z_3 = L^d(d-1,h+1)$ which represents the load value at the next hour of the previous day, z_4 = average temperature at the previous day. z_5 = average temperature at the present day, z_6 = average rainfall index at the previous day, z_7 = average rainfall index at the present day. The output, $y(t) = L(d,h)$, where d=1, 2, ..., 7 is the weekday's type number (e.g. d = 1 for Monday, d = 7 for Sunday), h = 1, 2, ..., 24 is the hour number. One should note the special case that if d = 1, (d-1) should be 7. $L(d,h)$ is the forecasted load for day-d, hour-h.

Real data of 12 weeks (week 1 to week 12) for learning and 2 weeks (week 13 to week 14) for testing are prepared. The number of membership function for each input variables is 2 ($m_i = 2$, i = 1, 2 ,..., 7) such that the numbers of rules is p = 2^7 = 128. From (21),

$$y(t)=\sum_{g=1}^{128} \mu_g(t) w_g \delta(\varsigma_g) \Big/ \sum_{g=1}^{128} \mu_g(t) \tag{25}$$

$$fitness = \frac{1}{1+err} \tag{26}$$

$$err = 1/12 \sum_{t=1}^{12} \left| y^d(t) - y(t) \right| / y^d(t) \tag{27}$$

(25) is for one of the 168 NFNs. The improved GA is employed to tune the parameters and structure of the NFN of (25). The population size is 10. Bounds of parameters are set at $0 \le \bar{x}_{ig_i} \le 1$, $0 \le \sigma_{ig_i} \le 0.4$ and $-1 \le \varsigma_g \le 1$. The chromosomes are $[\bar{x}_{ig_i}\ \sigma_{ig_i}\ \varsigma_g]$, $i = 1,\ldots,7.,g_i = 1,2.,g = 1,\ldots 128.$ The initial values of $\bar{x}_{ig_i}$, σ_{ig_i} , ς_g are 0.5, 0.2 and 1 respectively. The number of the iterations for training is 2000. For comparison, a 7-input-1-output NFN without rule switches trained by the traditional GA (bit-length = 9) is also applied for the load forecasting. The common network parameters are kept unchanged. The probabilities of crossover and mutation are 0.65 and 0.05 respectively. The simulation data are tabulated in Table II. The average number of rules for the proposed NFN is 69.58, implying a 45.64% reduction of the number of rules after learning. Table III show the average training error (MAPE) based on data of week 1 to week 12 and the average forecasting error (MAPE) from week 13 to week 14 for Sunday. The MAPEs are smaller as compared with the values offered by the traditional NFN tuned by the traditional GA. Fig. 4 show the forecasted daily load curve on Sunday at Week 13.

VI. Conclusion

In this paper, an improved GA has been proposed in which new genetic operators have also been introduced. Based on the benchmark, De Jong's test functions, it has been shown that the improved GA performs better than the traditional GA. An NFN has been proposed in which a switch is introduced in each fuzzy rule. Thus, the NFN can be optimized using the improved GA. A short-term load forecasting in an intelligent home has been realized using the proposed network.

ACKNOWLEDGEMENT

The work described in this paper was substantially supported by a Research Grant of The Hong Kong Polytechnic University (project numbers G-V954).

REFERENCES

[1] J.H. Holland, *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, MI, 1975.

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

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

[4] D. Liu, E. Flint, B. Gaucher, and Y. Kwark, "Wide band AC power line characterization," *IEEE Trans. Consumer Electronics*, Vol. 45, no.4, pp. 1087-1097, Nov.1999.

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

[6] A.G. Bakirtzls, V. Petridls, S.J. Klartzis, M.C. Alexladls, and A.H. Malssls, "A neural network short term load forecasting model for greed power system", *IEEE Trans. Power Systems*, vol.11, no.2, pp.858-863, May1996.

[7] Y.Y. Hse and K.L. Ho, "Fuzzy expect systems: an application to short-term load forecasting," *IEE Proceedings-C*, vol. 139, no.6, pp. 471-477, Nov 1992.

[8] B. Kosko, *Neural Networks and Fuzzy System: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, 1991.

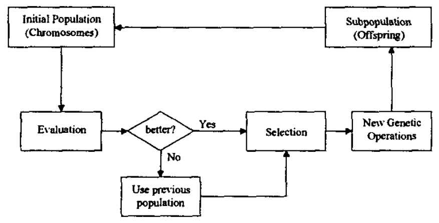

Fig. 1. Improved GA.

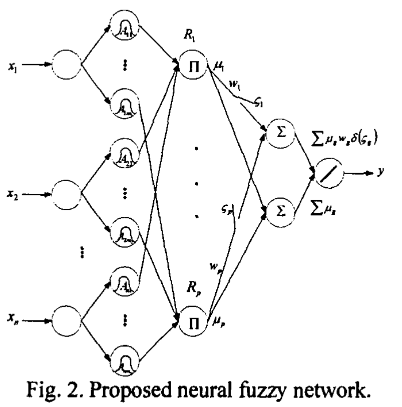

Fig. 2. Proposed neural fuzzy network.

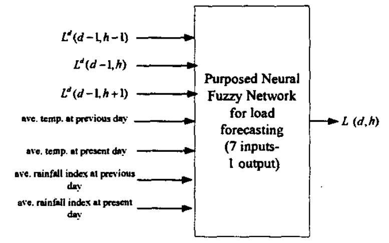
$L^d(d-1,h-1)$
$L^d(d-1,h)$
$L^d(d-1,h+1)$
ave. temp. at previous day
ave. temp. at present day
ave. rainfall index at previous day
ave. rainfall index at present day
Purposed Neural Fuzzy Network for load forecasting (7 inputs-1 output)
$L\ (d,h)$

Fig. 3. Proposed neural fuzzy network for load forecasting

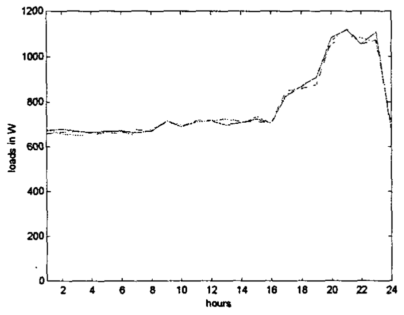

Fig. 4. Actual load (solid line) and forecast results for Sunday (Week13) from the proposed forecasting system (dashed line) and the traditional forecasting system.

TABLE I
SIMULATION RESULTS OF THE IMPROVED AND THE TRADITIONAL GAS BASED ON THE DE JONG'S TEST FUNCTIONS

	Improved GA		Traditional GA	
Test Functions	Fitness Value	Searching Time (s)	Fitness Value	Searchin g Time (s)
$f_1(x)$	0.999955	1.21	0.999382	8.35
$f_2(x)$	0.984039	1.22	0.810813	8.36
$f_3(x)$	0.583333	1.09	0.520833	10.93
$f_4(x)$	0.737526	2.69	0.14211	74.21
$f_5(x)$	0.995509	6.15	0.982912	21.13

TABLE II
SIMULATION DATA OF LOAD FORECASTING ON SUNDAY LEARNING

	Ave. Fitness value	Ave. Learning time	Ave. number of rule
Our Approach	0.984625	1700.92	69.58
Traditional Approach	0.979633	6085.96	128

TABLE III.
TRAINING ERROR AND FORECASTING ERROR (IN MAPE) FOR SUNDAY.

	Ave. Training error (Week 1-12)	Ave. Forecasting error (Week 13-14)
Our Approach	1.5564	1.5034
Traditional Approach	2.0884	2.1035