Recognition of Speech Commands Using a Modified Neural Fuzzy Network and an Improved GA[1]

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

Abstract: This paper presents the recognition of speech commands using a modified neural-fuzzy network. To train the parameters of the network, an improved genetic algorithm is proposed. As an application example, the proposed speech recognition approach is implemented in an Electronic Book experimentally to illustrate the design and its merits.

I. Introduction

When we want to communicate with a machine using speeches, it is difficult to let the machine fully recognize our spoken words. In general, the solution to this problem involves two main procedures: feature extraction and classification. Feature extraction is a preprocessing procedure in a speech recognition system. It is used to extract the specific voice features from the speech signals. In a noise free environment, each word or phoneme has its corresponding formant frequencies. However, when the environment is noisy, the speech signals are impure, and it is difficult to identify their corresponding features. The problem becomes more complicated when the speeches to be recognized have close phonemes. Thus, researchers worked on developing some distinctive feature extraction techniques. The most commonly used approaches are filter bank modeling and linear predictive coding (LPC) analysis [1]. Filter-bank modeling involves a bank of band-pass filters, which are used to model the characteristics of human ears. LPC analysis [1] approximates the current sampled speech as a linear combination of its past samples. The time-domain speech signals are first windowed into frames, and the autocorrelation coefficients between frames are obtained. This approach mimics the human vocal tract.

Classification is the next procedure to identify the input speech based on the feature parameters. Speech signal classification can be done in either a pattern recognition approach or a statistical approach. Artificial neural networks (ANN) and hidden Markov model (HMM) are commonly employed in the pattern recognition approach and statistical approach respectively [1, 8]. ANN is distinct in discrimination [5], and the classification can be done by measuring the closeness of the testing template to the trained templates. However, a large number of mathematical operations will be required if the number of speech samples is large and the duration of the speech is long. HMMs are good at statistical modeling of continuous speech signals. The states in the HMM characterize the phonemes. The speech is formulated into a sequence of states.

Cantonese digit speech recognition is a challenge task. Cantonese is a nine-tonal and syllabic language [2]. Some digits are difficult to discriminate when they are spoken in Cantonese, such as the digits '1' and '7'. Other human factors will introduce additional difficulties in obtaining a good performance in Cantonese speech recognition. Good algorithms for the speech feature extraction and classification are therefore important to give a high success rate on recognizing the spoken words.

An electronic book (eBook) reader should have no keyboard or mouse. The main input device is a touch screen. As many functions are implemented in a single eBook Reader, it is not convenient to access these functions through menus and hot keys alone. By using a small microphone, a one-step commanding process using speeches is proposed for eBooks. To realize speech recognition for commanding, a modified neural fuzzy network (NFN) trained by an improved GA is proposed in this paper. The proposed NFN consists of two NFNs such that one NFN is responsible for providing the parameters of another NFN. In this way, the trained NFN will have dynamic parameters. Effectively, the rule base for each recognized pattern will change according to the pattern itself. On applying the proposed NFN, the performance of speech recognition is improved, and the time of training is shortened. The proposed training algorithm has the advantage of offering a global solution in a faster rate. In this paper, the modified NFN is used to recognize ten Cantonese digit speeches, and implemented in an eBook reader practically.

II. Modified Neural Fuzzy Network

A modified neural fuzzy network is proposed to recognize speeches. Referring to Fig. 1, the proposed NFN consists of two NFNs, namely a tuner NFN and a classifier NFN. The parameters of traditional NFNs are usually fixed after the training. In the proposed NFN, some parameters of the classifier NFN are adjusted by the tuner NFN (which have fixed parameters after training) to cope with the changing environment during the operation. For example, when there are two sets of input-output data, namely S1 and S2, separated in a far distance within a large spatial domain shown in Fig. 2(a) and (b), it may be difficult for an NFN with fixed parameters and a limited number of rules to identify the features of the data. By using the proposed method, the rules of the classifier NFN are governed by the tuner NFN and are changed according to the network inputs. As a result, a set of input data will have a set of rules in the classifier NFN to handle them.

We use a fuzzy associative memory (FAM) [3-4, 18] type of rule base for both the tuner and classifier NFNs. An FAM is formed by partitioning the universe of discourse of each fuzzy variable according to the level of fuzzy resolution chosen for the antecedents, thereby generating a grid of FAM elements. The entry at each grid element in the FAM corresponds to a fuzzy premise. An FAM is thus interpreted as a geometric or tabular representation of a fuzzy logic rule base. The tuner and classifier NFNs share the same structure as shown in Fig. 3. We define the input and output variables as x_i and y_j respectively; where $i = 1, 2, \ldots, n_{in}$; n_{in} is the number of input variables; $j = 1, 2, \ldots, n_{out}$; n_{out} is the number of output variables. The behavior of y_j of the NFN is governed by m_f fuzzy rules of the following format;

[1] The work described in this paper was fully supported by a grant from the Centre for Multimedia Signal Processing, The Hong Kong Polytechnic University (project number A432).

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.

R_g: IF $x_1(t)$ is $A_{1g}(x_1(t))$ AND $x_2(t)$ is $A_{2g}(x_2(t))$ AND ... AND $x_{n_m}(t)$ is $A_{n_m g}(x_{n_m}(t))$

THEN $y_j(t)$ is w_{jg}, $g = 1, 2, \ldots, m_f$; $t = 1, 2, \ldots, n_d$ (1)

where n_d denotes the number of input-output data pairs; $A_{ig}(x_i(t))$ is the fuzzy term corresponding to $x_i(t)$; w_{jg} is the output singleton of the rule g. The grade of membership of each rule is defined as,

$$\mu_g(t) = A_{1g}(x_1(t)) \times A_{2g}(x_2(t)) \times \ldots \times A_{n_m g}(x_{n_m}(t)) \geq 0,$$
$$g = 1, 2, \ldots, m_f \quad (2)$$

The j-th output of the NFN, $y_j(t)$, is defined as,

$$y_j(t) = \frac{\sum_{g=1}^{m_f} \mu_g(t) w_{jg}}{\sum_{g=1}^{m_f} \mu_g(t)} \quad (3)$$

It should be noted that for this partially connected neural-fuzzy network, the number of rules m_f is equal to number of membership functions used for each input variables. Referring to Fig. 1, the structures of both the tuner and classifier NFNs are the same, with the outputs governed by (3). The main difference is the outputs of the tuner NFN are the values of the classifier NFN's output singleton, w_{jg}. Hence, the rule base of the classifier NFN will change with respect to the input $\mathbf{x}(t)$.

III. Improved Genetic Algorithm

Genetic algorithm (GA) is a directed random search technique [6] that is widely applied in optimization problems [6-8, 10]. This is especially useful for complex problems where the number of parameters is large and the analytical global solutions are difficult to obtain. GA has been applied in different areas such as fuzzy control [11-13, 17], path planning [14], greenhouse climate control [15], modeling and classification [16] etc.

A lot of research efforts have been spent to improve the performance of GA. Different selection schemes and genetic operators have been proposed. Selection schemes such as rank-based selection, elitist strategies, steady-state election and tournament selection were reported [21]. There are two kinds of genetic operations, namely crossover and mutation. Apart from random mutation and crossover, other crossover and mutation algorithms have been proposed. For crossover, two-point crossover, multipoint crossover, arithmetic crossover and heuristic crossover have been reported [6, 20-22]. For mutation, boundary mutation, uniform mutation and non-uniform mutation can be found [6, 20-22].

The standard GA process [6-7, 10] is shown in Fig. 4. In this paper, the standard GA is modified and new genetic operations [23] are introduced to improve its performance. The improved GA process is shown in Fig. 5. Its details will be given as follows.

A. Initial Population

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

$$P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\} \quad (4)$$

$$\mathbf{p}_i = [p_{i_1} \quad p_{i_2} \quad \cdots \quad p_{i_j} \quad \cdots \quad p_{i_{no\_vars}}],$$
$$i = 1, 2, \ldots, pop\_size; j = 1, 2, \ldots, no\_vars \quad (5)$$
$$para^j_{\min} \leq p_{i_j} \leq para^j_{\max} \quad (6)$$

where *pop_size* denotes the population size; *no_vars* denotes the number of variables to be tuned; p_{i_j}, $i = 1, 2, \ldots$, *pop_size*; $j = 1, 2, \ldots$, *no_vars*, are the parameters to be tuned; $para^j_{\min}$ and $para^j_{\max}$ are the minimum and maximum values of the parameter p_{i_j} respectively for all i. It can be seen from (4) to (6) 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,

$$fitness = f(\mathbf{p}_i) \quad (7)$$

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 by the method of spinning the roulette wheel [6]. It is believed that high potential parents will produce better offspring (survival of the best ones). The chromosome having a higher fitness value should therefore have a higher chance to be selected. First, assign a probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i = \frac{f(\mathbf{p}_i)}{\sum_{k=1}^{pop\_size} f(\mathbf{p}_k)}, \; i = 1, 2, \ldots, pop\_size \quad (8)$$

The cumulative probability $\hat{q}_i$ for the chromosome $\mathbf{p}_i$ is

$$\hat{q}_i = \sum_{k=1}^{i} q_k, \; i = 1, 2, \ldots, pop\_size \quad (9)$$

Randomly generate a nonzero real number, $d \in [0 \quad 1]$. Then, the chromosome $\mathbf{p}_i$ is chosen if $\hat{q}_{i-1} < d \leq \hat{q}_i$ ($\hat{q}_0 = 0$). Therefore, a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected. Hence, the best chromosomes will get more offspring, the average will stay and the worst will die off. In the selection process, two chromosomes are selected to undergo the genetic operations.

D. Genetic Operations

The genetic operations are to generate some new chromosomes (offspring) from their parents. They include crossover and mutation operations.

Crossover

The crossover operation exchanges information from the two parents, chromosomes $\mathbf{p}_1$ and $\mathbf{p}_2$, obtained in the selection process. The two parents will produce one offspring. First, four chromosomes will be generated according to the following equations:

$$\mathbf{os}^1_c = [os^1_1 \quad os^1_2 \quad \cdots \quad os^1_{no\_vars}] = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \quad (10)$$

$$\mathbf{os}^2_c = [os^2_1 \quad os^2_2 \quad \cdots \quad os^2_{no\_vars}] = \mathbf{p}_{\max}(1-w) + \max(\mathbf{p}_1, \mathbf{p}_2)w \quad (11)$$

$$\mathbf{os}^3_c = [os^3_1 \quad os^3_2 \quad \cdots \quad os^3_{no\_vars}] = \mathbf{p}_{\min}(1-w) + \min(\mathbf{p}_1, \mathbf{p}_2)w \quad (12)$$

$$\mathbf{os}^4_c = [os^4_1 \quad os^4_2 \quad \cdots \quad os^4_{no\_vars}] = \frac{(\mathbf{p}_{\max} + \mathbf{p}_{\min})(1-w) + (\mathbf{p}_1 + \mathbf{p}_2)w}{2} \quad (13)$$

$$\mathbf{p}_{\max} = [para^1_{\max} \quad para^2_{\max} \quad \cdots \quad para^{no\_vars}_{\max}] \quad (14)$$

$$\mathbf{p}_{\min} = [para^1_{\min} \quad para^2_{\min} \quad \cdots \quad para^{no\_vars}_{\min}] \quad (15)$$

where $w \in [0 \quad 1]$ denotes the weight to be determined by

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.

users, $\max(\mathbf{p}_1, \mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum among the corresponding element of $\mathbf{p}_1$ and $\mathbf{p}_2$. For instance, $\max([1 \ -2 \ 3],[2 \ 3 \ 1]) = [2 \ 3 \ 3]$. Similarly, $\min(\mathbf{p}_1, \mathbf{p}_2)$ gives a vector by taking the minimum value. For instance, $\min([1 \ -2 \ 3],[2 \ 3 \ 1]) = [1 \ -2 \ 1]$. Among $\mathbf{os}_c^1$ to $\mathbf{os}_c^4$, the one with the largest fitness value is used as the offspring $\mathbf{os}$ of the crossover operation:

$$\mathbf{os} \equiv [os_1 \ \ os_2 \ \ \cdots \ \ os_{no\_vars}] = \mathbf{os}_c^{i_{os}} \quad (16)$$

i_{os} denotes the index i which gives a maximum value of $f(\mathbf{os}_c^i)$, $i = 1, 2, ,3 ,4$.

If the crossover operation can provide a good offspring, a higher fitness value can be reached in less iteration. In general, two-point crossover, multipoint crossover, arithmetic crossover or heuristic crossover can be employed to realize the crossover operation [6, 20-22]. The offspring generated by these methods, however, may not be better than that from our approach. As seen from (10) to (13), the potential offspring after the crossover operation spreads over the domain. While (10) and (13) result in searching around the centre region of the domain (a value of w near to 1 in (13) can move $\mathbf{os}_c^4$ to be near $\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$), (11) and (12) move the potential offspring to be near the domain boundary (a small value of w in (11) and (12) can move $\mathbf{os}_c^2$ and $\mathbf{os}_c^3$ to be near $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ respectively).

Mutation

The offspring (16) will then undergo the mutation operation that changes the genes of the chromosomes. Then, the features of the chromosomes inherited from their parents can be changed. In general, various methods like boundary mutation, uniform mutation or non-uniform mutation [6, 20-22] can be employed to realize the mutation operation. In this paper, a different process of mutation is proposed. Every gene of the offspring $\mathbf{o}_s$ of (16) will have a chance to mutate governed by a probability of mutation, $p_m \in [0 \ \ 1]$, which is defined by the user. This probability gives an expected number ($p_m \times no\_vars$) of genes that undergo the mutation. For each gene, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, the operation of mutation will take place on that gene. The gene of the offspring of (16) is then mutated by:

$$\hat{o}_{s_k} = \begin{cases} o_{s_k} + \Delta o_{s_k}^U \text{ if } f(\mathbf{o}_s + \Delta \mathbf{o}_{s_k}^U) \geq f(\mathbf{o}_s - \Delta \mathbf{o}_{s_k}^L) \\ o_{s_k} - \Delta o_{s_k}^L \text{ if } f(\mathbf{o}_s + \Delta \mathbf{o}_{s_k}^U) < f(\mathbf{o}_s - \Delta \mathbf{o}_{s_k}^L) \end{cases},$$

$k = 1, 2, \ldots, no\_vars$ (17)

$$\Delta o_{s_k}^U = w_{m_k} r(para_{\max}^k - o_{s_k}) \quad (18)$$

$$\Delta o_{s_k}^L = w_{m_k} r(o_{s_k} - para_{\min}^k) \quad (19)$$

$$\Delta \mathbf{o}_{s_k}^U = [0 \ \ 0 \ \ \cdots \ \ \Delta o_{s_k}^U \ \ \cdots \ \ 0] \quad (20)$$

$$\Delta \mathbf{o}_{s_k}^L = [0 \ \ 0 \ \ \cdots \ \ \Delta o_{s_k}^L \ \ \cdots \ \ 0] \quad (21)$$

$r \in [0 \ \ 1]$ is a randomly generated number; $w_{m_k} \in (0 \ \ 1]$ is a weight governing the magnitudes of $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$. The value of weight w_{m_k} is varied by the value of $\frac{\tau}{T}$, where τ is the iteration number and T is the total number of iterations. The value of weight w_{m_k} should be small as $\frac{\tau}{T}$ increases in order to reduce the significance of the mutation. Based on this idea, a monotonic decreasing function governing w_{m_k} is proposed as follows,

$$w_{m_k} = w_f \left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \geq 0 \quad (22)$$

where $w_f \in [0 \ \ 1]$ and $w_\tau > 0$ determine the initial value and the decay rate respectively. Their values are chosen by the user. For a large value of w_f, it can be seen from (18) and (19) that $\Delta o_{s_k}^U \approx r(para_{\max}^k - o_{s_k})$ and $\Delta o_{s_k}^L \approx r(o_{s_k} - para_{\min}^k)$ initially as $\left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 1$ which ensure a large search space. When the value of $\left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 0$, the values of $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$ are small to ensure a small search space for fine-tuning.

E. Reproduction

The new offspring will be evaluated using the fitness function of (7). This new offspring will replace the chromosome with the smallest fitness value among the population if a randomly generated number between 0 and 1 is smaller than $p_a \in [0 \ \ 1]$, which is the probability of acceptance defined by users. Otherwise, the new offspring will replace the chromosome with the smallest fitness value only if the fitness value of the offspring is greater than the fitness value of that chromosome in the population. p_a is effectively the probability of accepting a bad offspring in order to reduce the chance of converging to a local optimum. After the operations of selection, crossover, mutation and reproduction, a new population is generated. This new population will repeat the same process. Such an iterative process can be terminated when it meets a defined condition, e.g. a defined number of iteration has been reached.

F. Benchmark Test Functions

Some benchmark test functions [8-9, 19] are used to examine the applicability and efficiency of the improved GA. Six test functions, $f_i(\mathbf{x})$, $i = 1, 2, 3, 4, 5, 6$ will be used, where $\mathbf{x} = [x_1 \ \ x_2 \ \ \cdots \ \ x_n]^T$, n is an integer denoting the dimension of the vector $\mathbf{x}$. The six test functions are defined as follows,

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, \ -5.12 \leq x_i \leq 5.12 \quad (23)$$

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

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right), \ -2.048 \leq x_i \leq 2.048 \quad (24)$$

where $n = 2$ and the minimum point is at $f_2(1, 1) = 0$.

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} floor(x_i + 0.5)^2, \ -5.12 \leq x_i \leq 5.12 \quad (25)$$

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

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + Gauss(0,1),\ -1.28 \le x_i \le 1.28 \qquad (26)$$

where $n = 3$ and the minimum point is at $f_4(0, 0, 0) = 0$. $Gauss(0, 1)$ is a function to randomly generate a floating-point number between 0 and 1.

$$f_5(\mathbf{x}) = \frac{1}{k} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6},\ -65.356 \le x_i \le 65.356 \qquad (27)$$

where

$$\mathbf{a} = \{a_{ij}\} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & \\ 32 & 32 & 32 & 32 & 32 & -16 & -16 & -16 & -16 & -16 & \\ -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix}$$

$k = 500$ and the maximum point is at $f_5(32, 32) \approx 1$.

$$f_6(\mathbf{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right],\ -5.12 \le x_i \le 5.12 \qquad (28)$$

where $n = 3$ and the minimum point is at $f_6(0, 0, 0) = 0$.

It should be noted that the minimum values of all functions in the defined domain are zero except for $f_5(\mathbf{x})$. The fitness functions for f_1 to f_4 and f_6 are defined as,

$$fitness = \frac{1}{1 + f_i(\mathbf{x})},\ i = 1, 2, 3, 4, 6. \qquad (29)$$

and the fitness function for f_5 is defined as,

$$fitness = f_5(\mathbf{x}) \qquad (30)$$

The proposed GA goes through these 6 test functions. The results are compared with those obtained by the standard GA with arithmetic crossover and non-uniform mutation [6, 20-22]. The control parameters of the proposed GA and the standard GAs are tabulated in Table I. These control parameters are selected by trial and error through experiments for good performance. The initial values of $\mathbf{x}$ in the population for a test function are set to be the same for both the proposed and the standard GAs. The number of iteration for each test function is listed in Table II. For test functions 1 to 6, the initial values are $[1\ 1\ 1]$, $[0.5\ 0.5]$, $[1\ \cdots\ 1]$, $[0.5\ \cdots\ 0.5]$, $[10\ \cdots\ 10]$ and $[1\ 1\ 1]$ respectively. The results of the average fitness values over 50 times of simulations based on the proposed and standard GAs are also tabulated in Table II. The performance of the proposed GA is better than that of the standard GA.

IV. Speech Command Recognition in an eBook

A block diagram of the speech recognition system is shown in Fig. 6. Speech signals are recorded from the microphone of the eBook. A speech vector, which records the speech amplitude against time, is obtained. It then undergoes a feature extraction process using the uniform filter-bank technique to extract the specific frequency components of the speech. The frequency components are obtained by Fast Fourier Transform. The uniform filter-bank is used to transform the large amount of frequency components into several speech frames. The number of speech frame is equal to the number band-pass filters used to model the whole speech, and the band-pass filters are evenly distributed along the frequency spectrum. The speech feature parameters are obtained by the following equations:

$$c_o(\alpha) = \frac{10 \log \sum_{i=1}^{m} s_{\alpha i}}{m},\ \alpha = 1, 2, 3 \ldots, k. \qquad (31)$$

$$d_o(\alpha + 1) = c_o(\alpha + 1) - c_o(\alpha),\ \alpha = 1, 2, 3 \ldots, k. \qquad (32)$$

$$d_o(1) = c_o(1) \qquad (33)$$

where c_o is the mean amplitude value of the frame α, α is the speech frame number, k denotes the total number of frames, m denotes the size of the frame, $s_{\alpha i}$ denotes the i-th element of the speech signal amplitude of frame α in frequency domain, d_o denotes the feature parameters of the frame.

The outputs of the feature extractor will be fed to the modified NFN, which is employed to classify the input speech command. The inputs of the modified NFN are $\mathbf{x} = \frac{\mathbf{d}_o}{\|\mathbf{d}_o\|}$ where $\mathbf{d}_o = [d_o(1)\ \ d_o(2)\ \cdots\ d_o(k)]$, $\|\cdot\|$ denotes the l_2 vector norm. The output values of the modified NFN are to indicate the similarity to each class. The training of the NFN is to maximize the following fitness value:

$$fitness = \frac{1}{1 + err} \qquad (34)$$

$$err = \frac{\sum_{t=1}^{num\_pat} \frac{\|\mathbf{y}_d(t) - \mathbf{y}(t)\|^2}{num}}{num\_pat} \qquad (35)$$

where $\mathbf{y}_d(t)$ is the desired output vector; $\mathbf{y}(t)$ is the network output vector, num denotes the dimension of the output vector (the number of classes to be recognized), num_pat denotes the number of training patterns. The desired output vector of the network is defined as follows.

$$\mathbf{y}_d = [a_1, a_2, a_3, \ldots, a_{num}] \qquad (36)$$

where a_i, $i = 1, 2, \ldots, num$, describes the target class of the system. Only the value of a_i for a particular class i is equal to 1, and the rest elements of $\mathbf{y}_d$ are all zero. The improved GA will be employed to train the proposed NFN. Referring to (2), (3) and Fig. 3, The parameters of the network, i.e. $[\bar{x}_{ig}^T\ \ \sigma_{ig}^T\ \ \omega_{jg}^T\ \ \bar{x}_{kl}^C\ \ \sigma_{kl}^C\ \ \omega_{ml}^C]$ for all i, j, g, k, l, m; form the chromosomes of the GA process. The superscripts T and C specify the tuner and classifier NFN parameters respectively. During the recognition, the position of the element of $\mathbf{y}(t)$ that has the largest value indicates the possible class of the input pattern.

V. Application and Results

The speech recognition system shown in Fig. 6 has been implemented in a practical eBook Reader. The Cantonese speech signals are sampled at 11kHz, 8-bit mono. The speech samples are recorded from a male speaker and the amplitude of each sample is normalized to lie between −1 to 1. 50 training patterns and 20 testing patterns for each digit are collected. The Cantonese speech digits are '0' to '9' (10 classes). We use (31) to (33) to obtain 20 feature parameters representing each digit. The speech feature coefficients of the ten Cantonese digits are processed by the proposed NFN with 20 inputs and 10 outputs. The membership functions of the proposed NFN are defined as follows.

$$A_{ig}(x_i(t)) = e^{\frac{-(x_i(t) - \bar{x}_{ig})^2}{2\sigma_{ig}^2}} \qquad (37)$$

where the parameters $\bar{x}_{ig}$ and σ_{ig} are the mean and standard deviation of the g-th membership function respectively. The improved GA trains the NFN based on the training patterns in order to maximize the following fitness function:

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.

$$fitness = \frac{1}{1+err} \tag{38}$$

$$err = \frac{\sum_{t=1}^{500} \|\mathbf{y}_d(t) - \mathbf{y}(t)\|^2}{500} \tag{39}$$

The number of training patterns for the modified NFN is 500. The first 50 training patterns are the feature parameters of the Cantonese digit '1'. Thus, the first 50 vectors of $\mathbf{y}_d$ are [1 0 0 0 0 0 0 0 0 0]. Similarly, the next 50 training patterns are the feature parameters of the Cantonese digit '2', and the corresponding 50 vectors of $\mathbf{y}_d$ are [0 1 0 0 0 0 0 0 0 0], and so on. Different numbers of membership functions for each input in the tuner and classifier NFNs have been tested. They are (3,5), (4,5), (5,5), (5,4) and (5,3), where the first and the second number correspond to the tuner NFN and the classifier NFN respectively. The learning process is done by a computer with a P4 1.4 GHz CPU and 256 MB RAM. The number of iteration for training is 50000; $w = 0.5, p_m = 0.01; w_f = 0.5, w_\tau = 5, p_a = 0.1$ for all cases. After training, 200 testing patterns (10 Cantonese digits × 20) are used to test the performance of the proposed NFN. The fitness values for training are tabulated in Table III. The fitness values for testing and the testing errors for each spoken digit are tabulated in Tables IV and V respectively. From Table V, we see that the best result is obtained when we have 5 membership functions for both the tuner NFN and classifier NFNs.

For comparison, a traditional NFN with different number of membership functions trained by GA with arithmetic crossover and non-uniform mutation [10] is also used to perform the recognition. The number of iteration is 50000, the shape parameter, the probability of crossover and the probability of mutation of the GA [10] are chosen to be 1, 0.8 and 0.05 respectively. The results are summarized in Tables VI to VIII. It can be seen that the proposed NFN using the same number of parameters (650) provides a better result. The successful recognition rates are 98.5% for the proposed method and 97% for the traditional method.

VI. Conclusion

A modified neural-fuzzy network has been proposed. An improved genetic algorithm trains the parameters of the proposed network so as to recognize 10 digits in Cantonese. The recognizer is implemented in an eBook practically.

References

[1] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[2] T. Lee, W.K. Lo, P.C. Ching, and H. Meng, "Spoken language resources for Cantonese speech processing," *Speech Communication*, vol. 36, Iss. 3-4, pp. 327-342, March 2002.

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

[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed. Prentice Hall, 1999.

[5] G.P. Zhang, "Neural network for classification: a survey," *IEE Trans. Syst., Man, Cybern. C*, vol. 30, no. 4, Nov. 2000.

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

[7] D.T. Pham and D. Karaboga, *Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 2000.

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

[9] K.A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems* (Ph.D. Thesis). Ann Arbor, MI: University of Michigan, 1975.

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

[11] H.K. Lam, F.H.F. Leung and P.K.S. Tam, "Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm," *IEEE Trans. Syst., Man and Cybern, Part B: Cybernetics* (to be published).

[12] Y.S. Zhou and L.Y. Lai "Optimal design for fuzzy controllers by genetic algorithms," *IEEE Trans. Industry Applications*, vol. 36, no. 1, pp. 93-97, Jan.-Feb. 2000.

[13] C.F. Juang, J.Y. Lin, and C.T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Systems, Man and Cybernetics, Part B*, vol. 30, no. 2, pp. 290–302, April, 2000.

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

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

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

[17] K. Belarbi and F. Titel, "Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 4, pp. 398-405, Aug. 2000.

[18] M. Brown and C. Harris, *Neuralfuzzy Adaptive Modeling and Control*. Prentice Hall, 1994.

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

[20] X. Wang and M. Elbuluk, "Neural network control of induction machines using genetic algorithm training," in *Conf. Record 31st IAS Annual Meeting*, vol. 3, 1996, pp. 1733-1740.

[21] L. Davis, *Handbook of Genetic Algorithms*. NY: Van Nostrand Reinhold, 1991.

[22] M. Srinivas and L.M. Patnaik, "Genetic algorithms: a survey," *IEEE Computer*, vol. 27, issue 6, pp. 17-26, June 1994.

[23] F.H.F. Leung, H.K. Lam, S.H. Ling and P.K.S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 79-88, Jan. 2003.

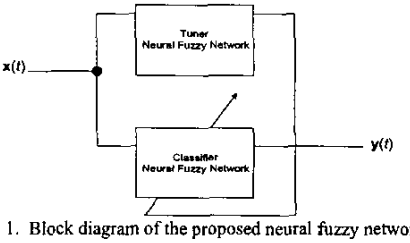

Fig. 1. Block diagram of the proposed neural fuzzy network.

(a) (b)

Fig. 2. (a) A diagram showing two data sets in spatial domain. (b) The feature curves for the two data sets.

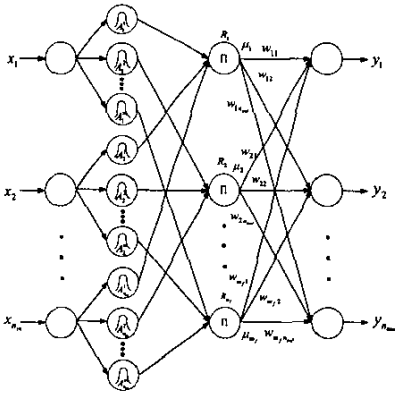

Fig. 3. Three-layer neural fuzzy network.

```
Procedure of the standard GA
begin
    τ←0     // τ: number of iteration
initialize P(τ)  //P(τ): population for iteration τ
    evaluate f(P(τ))  // f(P(τ)):fitness function
while (not termination condition) do
    begin
        τ←τ+1
        select 2 parents p1 and p2 from P(τ−1)
        perform genetic operations (crossover and mutation)
        reproduce a new P(τ)
        evaluate f(P(τ))
    end
end
```

Fig. 4. Procedure of the standard GA.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.

```
Procedure of the improved GA
begin
        τ→0          // τ: number of iteration
        initialize P(τ)  //P(τ): population for iteration τ
        evaluate f(P(τ))          //f(P(τ)):fitness function
while (not termination condition) do
        begin
        τ→τ+1
        select 2 parents p1 and p2 from P(τ−1)
        perform crossover operation according to equations
        (10) - (16)
        perform mutation operation according to equations
        (17) - (22) to generate the offspring os
        // reproduce a new P(τ)
                if random number < pa  // pa: probability of
                        acceptance
                        os replaces the chromosome with the
                        smallest fitness value in the population
                else if f(os) > smallest fitness value in the P(τ−1)
                        os replaces the chromosome with the
                        smallest fitness value
                end
                evaluate f(P(τ))
        end
end
```

Fig. 5. Procedure of the improved GA.

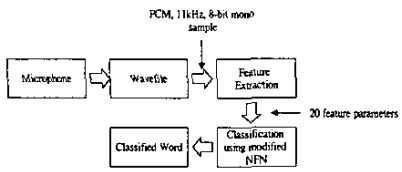

Fig. 6. Block diagram of the Cantonese digit speech recognition system.

Test function	w	p_m	w_f	w_g	p_a
$f_1(x)$	0.1	0.3	0.001	0.001	0.1
$f_2(x)$	0.5	0.5	0.01	10	0.1
$f_3(x)$	0.1	0.8	1	1000	0.1
$f_4(x)$	0.5	0.35	0.001	10	0.1
$f_5(x)$	0.5	0.8	0.1	0.1	0.1
$f_6(x)$	0.01	0.1	0.01	0.01	0.1

(a) Improved GA,

Test function	*b (Shape parameter)*	*p_c (Probability of crossover)*	*p_m (Probability of mutation)*
$f_1(x)$	5	0.7	0.9
$f_2(x)$	1	0.8	0.8
$f_3(x)$	0.1	0.7	0.6
$f_4(x)$	1	0.8	0.35
$f_5(x)$	0.1	0.8	0.5
$f_6(x)$	0.9	0.7	0.4

(b) Standard GA with arithmetic crossover and non-uniform mutation.

Table I. Control parameters of GAs for the benchmark test functions

Test function	*Average fitness value (proposed GA)*	*Average fitness value (standard GA)*	*Number of iteration*
$f_1(x)$	1.0000	1.0000	100
$f_2(x)$	0.9997	0.6393	5000
$f_3(x)$	1.0000	1.0000	200
$f_4(x)$	0.9997	0.8037	500
$f_5(x)$	1.0000	1.0000	250
$f_6(x)$	1.0000	0.7297	200

Table II. Average fitness values obtained from the proposed GA and the traditional GA for the benchmark test functions.

Membership function combination	*(3,5)*	*(4,5)*	*(5,5)*	*(5,4)*	*(5,3)*
Fitness value	0.9968	0.9963	0.9995	0.9969	0.9940
Number of parameters	470	560	650	560	470

Table III. Fitness values under different combinations of numbers of membership functions for the proposed NFN (50 training patterns for each digit).

Membership functions combination	*(3,5)*	*(4,5)*	*(5,5)*	*(5,4)*	*(5,3)*
Fitness value	0.9818	0.9855	0.9952	0.9854	0.9675

Table IV. Fitness values under different combinations of numbers of membership functions for the proposed NFN (20 testing patterns for each digit).

	Membership functions combination				
Digit #	*(3,5)*	*(4,5)*	*(5,5)*	*(5,4)*	*(5,3)*
1	5	5	0	13	4
2	1	1	1	1	3
3	5	3	0	1	1
4	2	1	1	0	3
5	0	0	0	0	0
6	2	1	0	0	1
7	0	2	0	3	15
8	5	3	0	0	13
9	1	1	1	0	1
0	1	0	0	1	1

Table V. Number of recognition errors for Cantonese digits '0'-'9' with the proposed NFN (20 testing patterns for each digit).

Number of membership functions	*9*	*10*	*11*	*12*	*13*
Fitness value	0.9891	0.9930	0.9655	0.9957	0.9962
No. of parameters	450	500	550	600	650

Table VI. Fitness values of the traditional NFN trained by GA with arithmetic crossover and non-uniform mutation (50 training patterns for each digit).

Number of membership functions	*9*	*10*	*11*	*12*	*13*
Fitness value	0.9626	0.9764	0.9895	0.9890	0.9932

Table VII. Fitness values of the traditional NFN trained by GA with arithmetic crossover and non-uniform mutation (20 testing patterns for each digit).

	Number of membership functions				
Digit #	*9*	*10*	*11*	*12*	*13*
1	9	5	10	1	1
2	2	2	2	1	1
3	3	2	2	1	1
4	1	3	1	1	0
5	0	0	2	1	0
6	2	1	0	0	2
7	13	4	2	4	0
8	14	4	0	0	0
9	1	0	1	1	1
0	2	1	2	0	0

Table VIII. Number of recognition errors for Cantonese digits '0'-'9' with the traditional NFN trained by GA with arithmetic crossover and non-uniform mutation (20 testing patterns for each digit).

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on July 7, 2009 at 00:46 from IEEE Xplore. Restrictions apply.