Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 35 (2008) 1711–1720
Expert Systems with Applications
www.elsevier.com/locate/eswa

A novel hybrid learning algorithm for parametric fuzzy CMAC networks and its classification applications

Cheng-Jian Lin [a,*], Jia-Hong Lee [b], Chi-Yung Lee [c]

[a] Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung City 811, Taiwan, ROC
[b] Department of Information Management, National Kaohsiung First University of Science and Technology, Kaohsiung City 811, Taiwan, ROC
[c] Department of Computer Science and Information Engineering, Nankai Institute of Technology, Nantou 542, Taiwan, ROC

Abstract

This paper shows fundamentals and applications of the parametric fuzzy cerebellar model articulation controller (P-FCMAC) network. It resembles a neural structure that derived from the Albus CMAC and Takagi–Sugeno–Kang parametric fuzzy inference systems. In this paper, a novel hybrid learning which consists of self-clustering algorithm (SCA) and modified genetic algorithms (MGA) is proposed for solving the classification problems. The SCA scheme is a fast, one-pass algorithm for a dynamic estimation of the number of hypercube cells in an input data space. The clustering technique does not require *prior* knowledge such as the number of clusters present in a data set. The number of fuzzy hypercube cells and the adjustable parameters in P-FCMAC are designed by the MGA. The MGA uses the sequential-search based efficient generation (SSEG) method to generate an initial population to determine the most efficient mutation points. Illustrative examples were conducted to show the performance and applicability of the proposed model.

Keywords: Cerebellar model articulation controller (CMAC); Genetic algorithms; TSK-type fuzzy model; Classification; Face detection

1. Introduction

In 1975, the cerebellar model articulation controller (CMAC) developed by Albus (1975a) and Albus (1975b), is an artificial neural network inspired by the cerebellum. The CMAC network (Albus, 1975a) is a local network implies for a given input vector. The input space is quantized into discrete states as well as larger size overlapped areas called *hypercubes*. Each hypercube covers many discrete states and is assigned a memory cell that stores information for it. Only a few of the networks nodes (or hypercube cells) will be active and will effectively contribute to the corresponding network output. The basic idea of the CMAC network is to store learned data into overlapping regions in a way that the data can easily be recalled but use less storage space. Furthermore, the action of storing weight information in the CMAC network is similar to that of the cerebellum in humans. Because of the simple structure and fast learning property of the CMAC network, it has been successfully applied in many fields, such as robot control (Miller, Hewes, Glanz, & Graft, 1990), signal processing (Kolcz & Allinson, 1994), pattern recognition (Glanz, Miller, & Graft, 1991), image coding (Iiguni, 1996), and equalization (Reay, 1995). The advantages of the CMAC network are summarized as follows: a simple local neural network that can treat as a lookup table, fast learning speed, high convergence rate, good generalization capability, and ease of implementation by hardware, etc. However, there are two major limitations for the Albus' CMAC network: difficult in selecting the memory structure parameters and enormous memory size requirement for solving high-dimensional problems. The first problem that is while the conventional CMAC network has a constant value assigned to each hypercube, the data for a quantized state are constant and the derivative information is not preserved. This problem can be solved by using non-constant differentiable basis functions, such as spline functions by

* Corresponding author.
E-mail address: cjlin@nuk.edu.tw (C.-J. Lin).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.08.086

Lane, Handelman, and Gelfand (1992), and fuzzy membership functions by Jou (1992), etc. In this paper, we use mathematical equations to describe the CMAC network with Gaussian basis functions as receptive field functions. The second problem, the choice of clustering technique in the CMAC network is an important consideration. This is due to the use of *partition-based* clustering techniques, such as fuzzy *C*-means (FCM) (Hung et al., 2001), linear vector quantization (LVQ) (He, Liu, & Palm, 1995), fuzzy Kohonen partitioning (FKP), and pseudo FKP (Ang, Quek, & Pasquier, 2003), to perform cluster analysis. However, such clustering techniques require *prior* knowledge such as the number of clusters present in a data set. To solve the above problem, online-based cluster techniques were proposed (Kasabov & Song, 2002; Tung & Quek, 2002). But there are still problems with these methods; that is, the clustering methods (Kasabov & Song, 2002; Tung & Quek, 2002) only consider the total variations of the mean and deviation in all dimensions per input. This is because the cluster numbers increase quickly. In this paper, we propose a new self-clustering algorithm (SCA) to overcome the above-mentioned problem.

Fuzzy modeling (Lin & Lee, 1996) has been recognized as a powerful tool which can facilitate the effective development of models by combining information from different sources. A fuzzy modeling consists of a set of fuzzy IF–THEN rules that describe the input–output mapping relationship of the network and learning algorithms from area of artificial neural networks. It can be better used to explain solutions to users than completely black-box models such as neural networks. The antecedents of fuzzy rules partition the input space into a number of linguistic term sets while the consequent constituent can be chosen as Mamdani-type fuzzy model (Wang & Mendel, 1992) which is a singleton value, or TSK-type fuzzy model which is a function of linear combination of input variables (Jang, 1993; Juang & Lin, 1998; Sugeno & Kang, 1988; Takagi & Sugeno, 1985). Many researchers (Jang, 1993; Juang & Lin, 1998) have been shown that if a TSK-type fuzzy model is used, the network size and learning accuracy is superior to those of Mamdani-type fuzzy model. Recently, several authors (Chen, Cooley, & Zhang, 1999; Jishuang, Chao, & Zhengzhi, 2003) adopt backpropagation (BP) learning algorithm to adjust parameters. Using the steepest descent technique in BP training could minimize the error function, allowing the algorithm to reach the local minima very fast while never finding a global solution. Besides, BP training performance depends on the initial system parameter values. For different network topologies one must derive new mathematical expressions for each network layer. Considering the aforementioned disadvantages, suboptimal performance occurs even for a suitable fuzzy model topology. The techniques capable of training the model parameters and finding the global solution while optimizing the overall structure are needed. In this respect, genetic algorithms (GAs) appear to be better candidates. Several GAs based approaches have appeared in the literatures (Homaifar & McCormick, 1995; Juang, 2002; Karr, 1991; Kusumoputo, Irwanto, & Jatmiko, 2002; Lee & Takagi, 1993). Karr (1991) used a GA to generate membership functions for a fuzzy system. In Karr's work, a user needs to declare an exhaustive rule set and then use GAs to design only the membership functions. In Kusumoputo et al. (2002) and Homaifar and McCormick (1995), a fuzzy controller design method that used GAs to find the membership functions and the rule sets simultaneously was proposed. In Lee and Takagi (1993), a GA was used to tune the consequent parameters of TSK-type fuzzy rules (Takagi & Sugeno, 1985), as well as the membership functions in the precondition parts. Hence, we propose a new network structure which is mainly derived from the CMAC algorithm and Takagi–Sugeno–Kang (TSK) parametric fuzzy inference systems (Merz & Freisleben, 2000; Takagi & Sugeno, 1985), and a novel hybrid learning algorithm. A novel hybrid learning algorithm, which consists of SCA and MGA, is proposed for solving the classification problems. The MGA uses the sequential-search based efficient generation (SSEG) method to generate an initial population to determine the most efficient mutation points. The advantages of the proposed MGA are summarized as follows: (1) It can reduce the population sizes to a minimum size (i.e., only four population sizes); (2) The best chromosome will be chosen to perform the mutation operator in every generation; (3) The MGA method converges more quickly than existing traditional genetic methods. The two examples given confirm the effectiveness of the proposed model.

The rest of this paper is organized as follows. The fundamentals of the P-FCMAC network are detailed in Section 2. Section 3 presents a novel hybrid learning algorithm for the P-FCMAC network. The various applications of the proposed network are shown in the Section 4. The conclusions are summarized in the last section.

2. The parametric fuzzy CMAC (P-FCMAC) network

In this section, we propose a new parametric fuzzy cerebellar model articulation controller (P-FCMAC) network. The architecture of the P-FCMAC network is illustrated in Fig. 1, which consists of the input space partition, association memory selection, and defuzzification. The P-FCMAC network like the conventional CMAC network that also approximates a nonlinear function $y = f(x)$ by using two primary mappings:

$$S : X \Rightarrow A \tag{1}$$

$$P : A \Rightarrow D \tag{2}$$

where X is a N_D-dimensional input space, A is a N_A-dimensional association space, and D is a one-dimensional output space. These two mappings are realized by fuzzy operations. The function $S(x)$ also maps each point x in the input space onto an association vector $\alpha = S(x) \in A$ that has N_L nonzero elements ($N_L < N_A$). Different from conventional CMAC network, the association vector

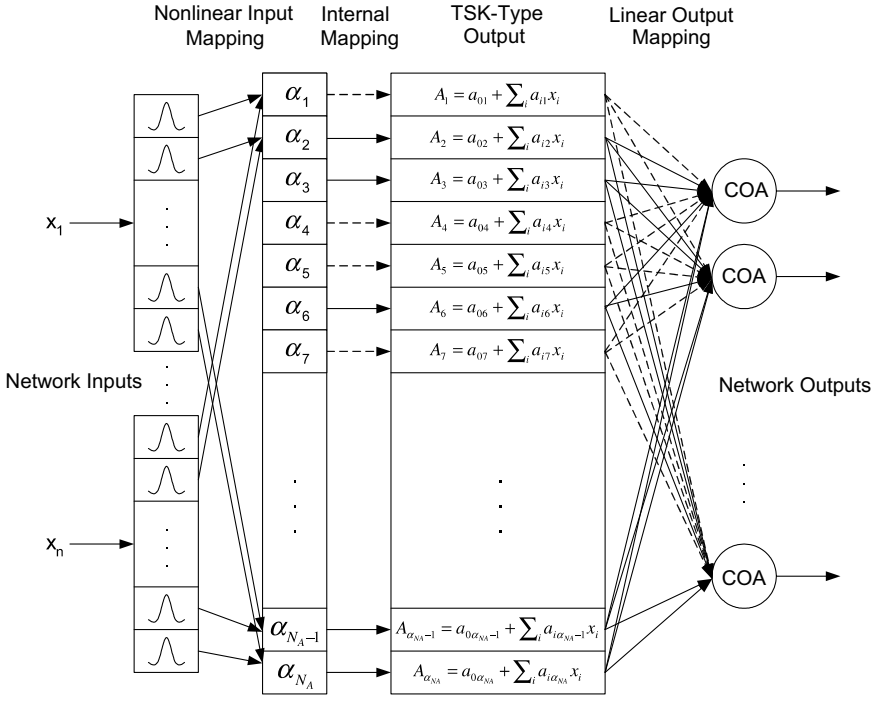

Fig. 1. The architecture of the P-FCMAC network.

$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{N_A})$, where $0 \leqslant \alpha \leqslant 1$ for all components in α, is derived from the composition of the receptive field functions and sensory inputs. Another, several hypercubes is addressed by the input state x that hypercube value is calculated by product operation through the strength of the receptive field functions for each input state. In the P-FCMAC network, we use Gaussian basis function as the receptive field functions and the linear parametric equation of the network input variance as the TSK-type output for learning. Some learned information is stored in the receptive field functions and TSK-type output vectors. A one-dimension Gaussian basis function can be given as follows:

$$\mu(x) = \mathrm{e}^{-((x-m)/\sigma)^2} \tag{3}$$

where x represents the specific input state, m represents the corresponding center, and σ represents the corresponding variance. Let us consider a N_D-dimensional problem. A Gaussian basis function with N_D dimensions is given as follows:

$$\alpha_j = \prod_{i=1}^{N_D} \mathrm{e}^{-((x_i-m_{ij})/\sigma_{ij})^2} \tag{4}$$

where $\prod$ represents the *product* operation, the α_j represents the jth element of the association vector, x_i represents the input value of the ith dimension for a specific input state x, m_{ij} represents the center of the receptive field functions, σ_{ij} represents the variance of the receptive field functions, and N_D represents the number of the receptive field functions for each input state. The function $P(\alpha)$ computes a scalar output y by projecting the association vector onto a vector of adjustable receptive field functions. Each element of the receptive field functions is inferred to produce a partial fuzzy output by applying the value of its corresponding association vector as input matching degree. The partial fuzzy output is defuzzified into a scalar output y by the centroid of area (COA) approach. Then the actual output y is derived as follows:

$$y = \frac{\sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\sum_{j=1}^{N_L} \alpha_j} \tag{5}$$

The jth element of the TSK-type output vectors is described as follows:

$$a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i \tag{6}$$

where a_{oj} and a_{ij} denote the scalar value, N_D denotes the number of the input dimensions, N_L denotes the number of hypercube cells, and x_i denotes the ith input dimension. Based on the above structure, a novel hybrid learning algorithm will be proposed to determine the proper network structure and its adjustable parameters.

3. A novel hybrid learning algorithm for P-FCMAC network

A novel hybrid learning algorithm, which consists of an input space partition scheme and a parameter learning scheme, is developed for constructing the P-FCMAC network. First, the input space partition scheme is used to determine proper input space partitioning and to find the mean and the width of each receptive field function. The

input space partition is based on the self-clustering algorithm (SCA) to appropriately determine the various distributions of the input training data. Second, the parameter learning scheme is based on modified genetic algorithm (MGA). The MGA is used to adjust the receptive field functions and the TSK-type output vector. According to the requirements of the system, these parameters will be given proper values to represent the memory information. For the initial system, the values of the tuning parameters a_{0j} and a_{ij} of the element of the TSK-type output vector are generated randomly, and the m and σ of receptive field functions are generated by the proposed SCA.

3.1. An input space partition scheme

The receptive field functions can map input patterns. Hence, the discriminative ability of these new features is determined by the centers of the receptive field functions. To achieve good classification, centers are best selected based on their ability to provide large class separation.

An input space partition scheme, called self-clustering algorithm (SCA), is proposed to implement scatter partitioning of the input space. Without any optimization, the SCA is a fast, one-pass algorithm for a dynamic estimation of the number of hypercube cells in a set of data, and for finding the current centers of hypercube cells in the input data space. It is a distance-based connectionist clustering algorithm. In any hypercube cell, the maximum distance between an example point and the hypercube cell center is less than a threshold value, which has been set as a clustering parameter and which would affect the number of hypercube cells to be estimated.

In the clustering process, the data examples come from a data stream, and the process starts with an empty set of hypercube cells. When a new hypercube cell is created, the hypercube cell center, C, is defined, and its hypercube cell distance and hypercube cell width, D_c and W_d, is initially set to zero. When more samples are presented one after another, some created hypercube cells will be updated by changing the positions of their centers and increasing the hypercube cell distances and hypercube cell width. Which hypercube cell will be updated and how much it will be changed depends on the position of the current example in the input space. A hypercube cell will not be updated any more when its hypercube cell distance, D_c, reaches the value that is equal to the threshold value D_{thr}. The detail process is described in (Lin, Chen, & Lee, 2004).

3.2. The parameter learning scheme

In the parameter learning scheme, there are four parameters need to be tuned, i.e. m_{ij}, σ_{ij}, a_{0j}, and a_{ij}. In this paper, we propose the MGA to tune the free parameters. The MGA is unlike traditional genetic algorithm which an initial population is generated randomly within a fixed range. It uses the sequential-search based efficient generation (SSEG) method to generate an initial population and to decide on efficient mutation points. The population size will be reduced to a minimum size and the best chromosome will be chosen to perform the mutation operator in every generation. Like traditional genetic algorithm, the proposed MGA consists of three major operators: reproduction, crossover, and mutation. Before the details of these three operators are explained, coding, initialization are discussed. The coding step, we divide a chromosome into two parts. The first part of the chromosome gives the parameters of the nonlinear input mapping α_R, and the second part of the chromosome gives the consequent parameters of the TSK-type output A_R (i.e., the coefficients of the linear combination). The initialization step assigns the population values before the evolution process begins. The whole learning process is described step by step below.

3.2.1. Coding step

The first step in MGA is to code a P-FCMAC model into a chromosome. Fig. 2 shows an example of a P-FCMAC model coded into a chromosome, where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with ith dimension and jth hypercube cell (i.e., $i = 1 \ldots n, j = 1 \ldots R$, R represents the number of hypercube cell), a_{oj} and a_{ij} denote the scalar value of the TSK-type output vectors, respectively.

3.2.2. Initialization step

Before the P-FCMAC model is designed, individuals forming an initial population should be generated. Unlike traditional genetic algorithm, an initial population is generated randomly within a fixed range. In MGA, the initial population is generated using the SSEG method to ensure that chromosomes with good genes can be generated. The detailed steps of the initialization method are described as follows:

- **Step 0:** The first chromosome that represents a P-FCMAC model will be generated initially. The following formulations show how to generate the first chromosome:

$$\text{Deviation} : \text{Chr}_j[p] = \text{random}[\sigma_{\min}, \sigma_{\max}] \quad \text{where } p = 2, 4, 6, \ldots, 2 * n - 1 \tag{7}$$

$$\text{Mean} : \text{Chr}_j[p] = \text{random}[m_{\min}, m_{\max}] \quad \text{where } p = 1, 3, 5, \ldots, 2 * n - 1 \tag{8}$$

$$\text{Weight} : \text{Chr}_j[p] = \text{random}[a_{\min}, a_{\max}] \quad \text{where } p = 2 * n + 1, \ldots, 2 * n + (1 + n) \tag{9}$$

where Chr_j means chromosome, $p$represents the pth gene in a Chr_j; j represents jth hypercube cell and $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$ and $[a_{\min}, a_{\max}]$ represent the rang that we predefined to generate the chromosomes.

- **Step 1:** To generate the other chromosomes, we propose the SSEG method to generate the new chromosomes. The search algorithm of SSEG is similar to the local search procedure in (Merz & Freisleben, 2000). In SSEG, every gene in the previous chromosomes is selected using a sequential search and the gene's value is updated to evalu-

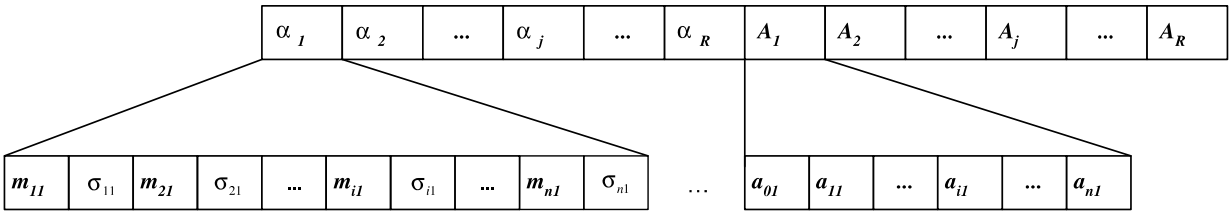

Fig. 2. Coding a P-FCMAC model into a chromosome in MGA method.

ate the performance based on the fitness value. The details of the SSEG method are as follows:

(a) Sequentially search for a gene in the previous chromosome.
(b) Update the chosen gene in (a) according to the following formula:

$$\mathrm{Chr}_j[p] = \begin{cases} \mathrm{Chr}_j[p] + \Delta(\text{fitness\_value}, \delta_{\max} - \mathrm{Chr}_j[p]), & \text{if } \eta > 0.5 \\ \mathrm{Chr}_j[p] - \Delta(\text{fitness\_value}, -\mathrm{Chr}_j[p - \delta_{\min}), & \text{if } \eta < 0.5 \\ \text{where } p = 2,4,6,\ldots,2*n \end{cases} \tag{10}$$

$$\mathrm{Chr}_j[p] = \begin{cases} \mathrm{Chr}_j[p] + \Delta(\text{fitness\_value}, m_{\max} - \mathrm{Chr}_j[p]), & \text{if } \eta > 0.5 \\ \mathrm{Chr}_j[p] - \Delta(\text{fitness\_value}, -\mathrm{Chr}_j[p - m_{\min}), & \text{if } \eta < 0.5 \\ \text{where } p = 1,3,5,\ldots,2*n-1 \end{cases} \tag{11}$$

$$\mathrm{Chr}_j[p] = \begin{cases} \mathrm{Chr}_j[p] + \Delta(\text{fitness\_value}, a_{\max} - \mathrm{Chr}_j[p]), & \text{if } \eta > 0.5 \\ \mathrm{Chr}_j[p] - \Delta(\text{fitness\_value}, -\mathrm{Chr}_j[p - a_{\min}), & \text{if } \eta < 0.5 \\ \text{where } p = 2^*n+1,\ldots,2^*n+(1+n) \end{cases} \tag{12}$$

where

$$\Delta(\text{fitness\_value}, v) = v^* \lambda^* (1/\text{fitness\_value})^{\lambda} \tag{13}$$

where η, $\lambda \in [0,1]$ are the random values; fitness_value is the fitness computed using Eq. (14); p represents the pth gene in a chromosome; and j represents jth hypercube cell, respectively. The function $\Delta(\text{fitness_value}, v)$returns a value, such that $\Delta(\text{fitness\_value}, v)$ comes close to 0 as fitness_value increases. This property causes the mutation operator to search the space uniformly during the initial stage (when fitness_value is small) and locally during the later stages, thus increasing the probability of generating children closer to its successor than a random choice and reducing the number of generations.

If the new gene that is generated from (b) can improve the fitness value, then replace the old gene with the new gene in the chromosome. If not, recover the old gene in the chromosome. After this, go to (a) until every gene is selected. The pseudo code for the SSEG method is listed in Fig. 3. The $\mathrm{Chr}_{k,j}$ represents the kth chromosome and jth hypercube cell in a neural fuzzy system. And Nf denote the size of the population, fitness($\mathrm{Chr}_{k,j\_\mathrm{new}}$) is a fitness function by Eq. (14) using the kth new chromosome.

```
Begin
    Let p=0,i=0;
  Repeat
      k=k+1;
      Repeat
        j=j+1;
        Repeat
          p=p+1;
          Perform Chr_{k,j_new}=inttialize(Chr_{k,j_old}[p]);by(11)to(14);
            Evaluate fitness(Chr_{k,j_new}) and fitness(Chr_{k,j_old}) by(15);
          If fitness(Chr_{k,j_new}) >fitness(Chr_{k,j_old}) Then
              Chr_{k,j_old} = Chr_{k,j_new}; else Chr_{k,j_new} = Chr_{k,j_old};
        Until p=2*n+(1+n);
      Until j=R;
    Until k=Nf;
End
```

Fig. 3. The pseudo code for the SSEG method.

• **Step 2:** If no genes are selected to improve the fitness value in step 1, than the new chromosome will be generated according to step 0. After the new chromosome is generated, the initialization method returns to step 1 until the total number of chromosomes is generated.

In this paper, the fitness value is designed according the follow formulation:

$$\text{Fitness Value} = 1 - (E(y,\bar{y})/N) \tag{14}$$

$$\text{where } E(y,\bar{y}) = \sum_{i=1}^{N} |y_i - \bar{y}_i| \tag{15}$$

where y_i represents the true value of the ith output, $\bar{y}_i$ represents the predicted value, $E(\bar{y},\bar{y})$ is a error function and N represents a numbers of the training data of each generation.

3.2.3. Reproduction step

Reproduction is a process in which individual strings are copied according to their fitness value. In this study, we use

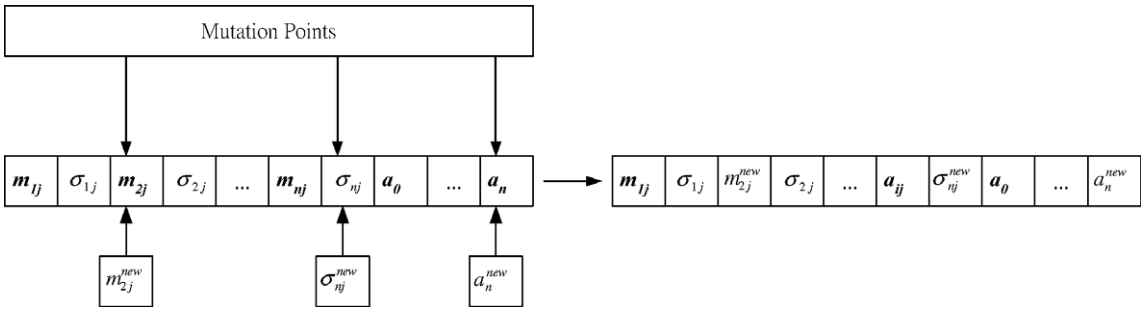

Fig. 4. Efficient mutation operation in three mutation points with jth hypercube cell.

the roulette-wheel selection method (Cordon, Herrera, Hoffmann, & Magdalena, 2001) – a simulated roulette is spun–for this reproduction process. The best performing individuals in the top half of the population (Juang, Lin, & Lin, 2000) advances to the next generation. The other half is generated to perform crossover and mutation operations on individuals in the top half of the parent generation.

3.2.4. Crossover step

Reproduction directs the search toward the best existing individuals but does not create any new individuals. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurred for a selected pair with a crossover rate that was set to 0.5 in this study. The first step is to select the individuals from the population for the crossover. Tournament selection (Cordon et al., 2001) is used to select the top-half of the best performing individuals (Juang et al., 2000). The individuals are crossed and separated using a two-point crossover that is the new individuals are created by exchanging the site's values between the selected sites of parents' individual. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

3.2.5. Mutation step

Although reproduction and crossover will produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation is an operator that randomly alters the allele of a gene. In this paper we propose using efficient mutation, which is unlike the traditional mutation, to mutate the chromosomes. In MGA, we perform efficient mutation using the best fitness value chromosome of every generation. And we use SSEG to decide on the mutation points. When the mutation points are selected, we use Eqs. (10)–(13) to update the genes. The efficient mutation of an individual is shown in Fig. 4.

The aforementioned steps are done repeatedly and stopped when the predetermined condition is achieved.

4. Illustrative examples

In this section, we compare the performance of the P-FCMAC network with other some existing models on classification applications. The first example is the face detection problem. The second example is the Wisconsin breast cancer diagnostic data. The initial parameters are given in Table 1 before training.

4.1. Example 1: the face detection problem

Face detection from images is a key problem in human computer interaction studies and pattern recognition research. In this paper, we use P-FCMAC to solve the face detection in color image problems. Color image of pre-processing contain three major modules: (1) lighting compensation; (2) color segment; (3) skin tone detection. Lighting compensation uses reference white technology to adjust the original hue of color image. We chose the

Table 1
The initial parameters before training

Parameters	Value
Population size	4
Crossover rate	0.5
Coding type	Real number
$[\sigma_{\min}, \sigma_{\max}]$	[0,1]
$[m_{\min}, m_{\max}]$	[0,1]
$[a_{\min}, a_{\max}]$	[−20,20]

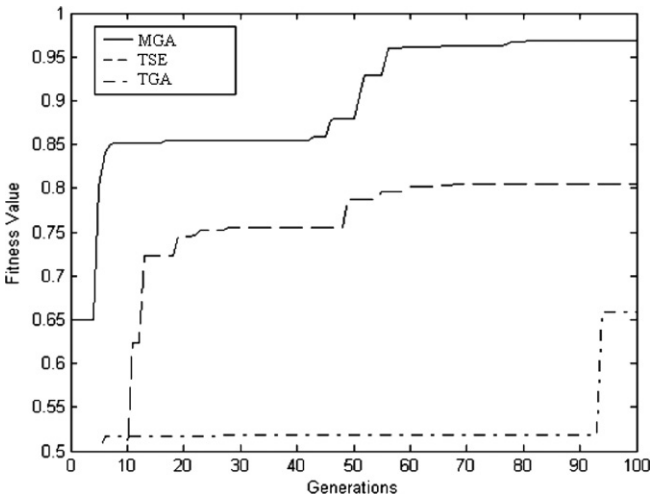

Fig. 5. The training curves of the three methods.

Table 2
Performance comparison of some existing models

	MGA	TSE (Juang et al., 2000)	TGA (Karr, 1991)	Neural networks (Lin & Lee, 1996)
Training data	6000	6000	6000	6000
Accuracy rate (training)	96.9%	80.5%	71.6%	85.35%
Error (training)	186	1169	1701	897
Accuracy rate (testing)	92.3%	71.0%	63.2%	66.12%
Error (testing)	466	1740	2210	2033
Iterations/generations	100	100	100	2500

Fig. 6. Original of California Institute of Technology face database.

YCbCr color space to substitute RGB color space for color segment. Because of the YCbCr color space has good separation between the luminance and chrominance information.

We exploited California Institute of Technology face database, via http://www.vision.caltech.edu/Image_Datasets/faces/, including 450 images, each of size 320×240 pixels, containing 27 different people and a variety of lighting, backgrounds and facial expressions. This experiment used three input dimensions (Y, Cb and Cr), and there were 6000 training data and 6000 test data. Further, through the network generated binary outputs (1/0 for skin/non-skin). The initial threshold value D_{thr} in the SCM is 0.6. After the SCM clustering process, there are four hypercube cells generated. In this example, the performance of the MGA method was compared with the traditional symbiotic evolution (TSE) (Juang et al., 2000) and traditional genetic algorithm (TGA) (Karr, 1991). Fig. 5 shows the training curves for the three methods. We could find the proposed MGA method converged more quickly than some GAs methods (Juang et al., 2000; Karr, 1991). The performance of the MGA method was also compared with neural network (Lin & Lee, 1996). It indicates includes the training and testing accuracy rates, training and testing errors,

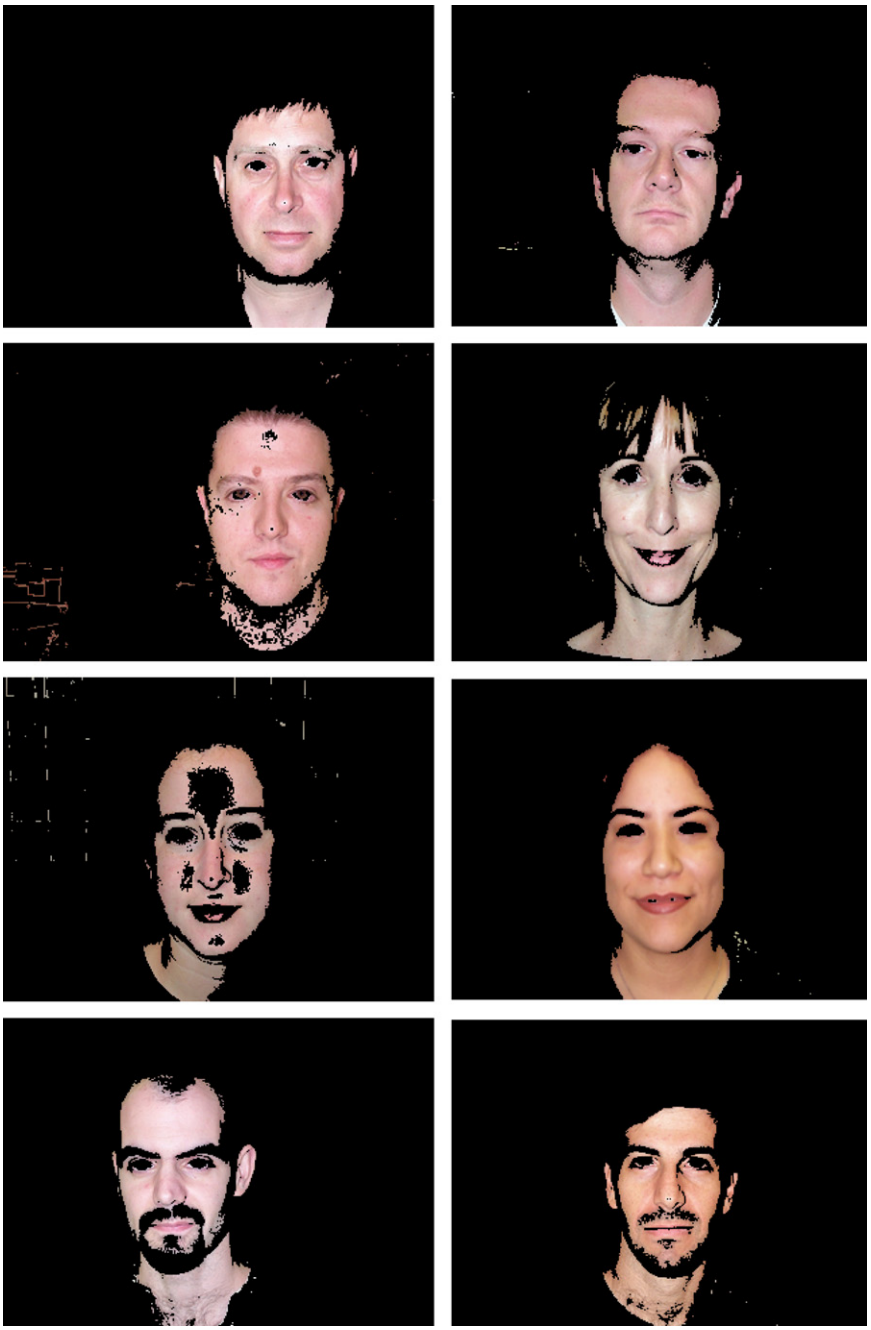

Fig. 7. Test result of three dimension input.

Table 3
Experiment results for independent runs

Experiment #	Iteration	1	2	3	4	5	Average
Accuracy (%) for TGA	1000	93.54	93.54	94.13	89.73	91.49	92.49
Accuracy (%) for MGA	300	97.36	97.36	97.07	97.95	97.07	97.36

Table 4
Experimental results for Wisconsin breast cancer diagnostic data

Models	Average recognition rate (%)
MSC (Lovel & Bradley, 1996)	94.9
NEFCLASS (Nauck & Kruse, 1997)	92.7
NNFS (Setiono & Liu, 1997)	93.94
FEBFC (Lee et al., 2001)	94.67
SANFIS (Wang & George Lee, 2002)	96.07
P-FCMAC	**97.36**

and the iterations. The comparison results are tabulated in Table 2.

The California Institute of Technology face database consists of major of images in involving complex backgrounds employing diverse lighting. Hence, from the comparison data listed in Table 2, we see that the testing accuracy rate in the traditional neural network is about 85.35% (this is from 2500 iterations). The TSE testing accuracy rate was about 71.0%. The TGA testing accuracy rate was about 63.2%, and the MGA was about 92.3%. Our method shows a better testing accuracy rate performance than other methods. This demonstrates that the California Institute of Technology face database is more complex leading to an accuracy rate decrease. However, the MGA method maintained a superior accuracy rate. We utilize color images from the California Institute of Technology face database are shown in Fig. 6. The output results using MGA method are shown in Fig. 7. It demonstrates that our approach accurately determines the facial region.

4.2. Example 2: the Wisconsin breast cancer diagnostic data

The Wisconsin Breast Cancer Diagnostic data set are available from the University of California, Irvine, via an anonymous ftp://ftp.ics.uci.edu/pub/machine-learning-databases. It contains 699 patterns distributed into two output classes, benign and malignant. Each pattern consists of nine input features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. There are 458 patterns in the benign class and the other 241 patterns are in the malignant class. Since 16 patterns containing missed values, we use 683 patterns to evaluate the performance. To demarcate the output y of the P-FCMAC model using the following rules:

$$\text{Class} = \begin{cases} \text{Benign}, & \text{if } y < 0 \\ \text{Malignant}, & \text{if } 0 \leqslant y \end{cases} \tag{16}$$

A half of 683 patterns were used as the training set which were randomly chosen, and the remaining patterns were used as the testing set. We also repeated the experiment on five different training-testing data sets that are obtained via a random process from the original Wisconsin Breast Cancer Diagnostic data. The initial threshold value D_{thr} in the SCA is 24.2. After the SCA clustering process, there are four hypercube cells generated. Table 3 shows the classification accuracy rate of five different training-testing data for traditional genetic algorithm (TGA) (Karr, 1991) and MGA. The average classification accuracy rate using MGA method is 97.36%. We also compare the performance of our model with some existing methods (Lee, Chen, Chen, & Jou, 2001; Lovel & Bradley, 1996; Nauck & Kruse, 1997; Setiono & Liu, 1997; Wang & George Lee, 2002). The comparison results are tabulated in Table 4. The results show that the proposed model has higher average classification accuracy rate than other methods.

5. Conclusion

In this paper, a new parametric fuzzy CMAC (P-FCMAC) network was proposed for classification applications. The proposed model uses a non-constant differentiable basis function, i.e. Gaussian basis function, to model the hypercube structure and the linear parametric equation of the TSK-type output that can proper to express the various input state. The proposed a novel hybrid algorithm consists of the self-clustering algorithm (SCA) to perform input space partition and the modified genetic algorithm (MGA) to perform parameter learning. The advantages of the proposed P-FCMAC network are summarized as follows: (1) it implements scatter partitioning of the input space dynamically; (2) it can keep a smaller rms error; and (3) it has much lower memory requirement than conventional CMAC network. The two examples given confirm the effectiveness of the proposed model.

References

Albus, J. S. (1975a). A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems Measurement and Control – Transactions of the ASME*(September), 220–227.

Albus, J. S. (1975b). Data storage in the cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems Measurement and Control – Transactions of the ASME*(September), 228–233.

Ang, K. K., Quek, C., & Pasquier, M. (2003). POPFNN-CRI (S): Pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*.

Chen, L., Cooley, D. H., & Zhang, J. (1999). Possibility-based fuzzy neural networks and their application to image processing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 29*(1).

Cordon, O., Herrera, F., Hoffmann, F., & Magdalena, L. (2001). *Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases. Advances in fuzzy systems – Applications and theory* (vol. 19). NJ: World Scientific Publishing.

Glanz, F. H., Miller, W. T., & Graft, L. G. (1991). An overview of the CMAC neural network. *Proceedings of IEEE on Neural Networks and Ocean Engineering*, 301–308.

He, J., Liu, L., & Palm, G. (1995). Speaker identification using hybrid LVQ–SLP networks. In *Proceedings of IEEE international conference on neural networks, vol. 4* (pp. 2052–2055).

Homaifar, A., & McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions of Fuzzy Systems, 3*(9), 129–139.

Hung, M. C., & Yang, D. L. (2001). The efficient FCM clustering technique. In *Proceedings of IEEE international conference on data mining* (pp. 225–232).

Iiguni, Y. (1996). Hierarchical image coding via cerebellar model arithmetic computers. *IEEE Transactions on Image Processing, 5*(October), 1393–1401.

Jang, J. S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 23*, 665–685.

Jishuang, Q., Chao, W., & Zhengzhi, W. (2003). Structure-context based fuzzy neural network approach for automatic target detection. In *Geoscience and remote sensing symposium, 2003, IGARSS'03, proceedings, 2003 IEEE international, vol. 2* (pp. 767–769).

Jou, C. C. (1992). A fuzzy cerebellar model articulation controller. In *Proceedings of IEEE international conference on fuzzy systems* (pp. 1171–1178).

Juang, C. F. (2002). A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Transactions of Fuzzy Systems, 10*(2), 155–170.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–31.

Juang, C. F., Lin, J. Y., & Lin, C. T. (2000). Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 30*(2), 290–302.

Karr, C. L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm. In *Proceedings of fourth conference on genetic algorithms* (pp. 450–457).

Kasabov, N. K., & Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems, 10*(2), 144–154.

Kolcz, A., & Allinson, N. M. (1994). Application of the CMAC input encoding scheme in the N-tuple approximation network. *IEE Proceedings – Computer and Digital Techniques, 141*, 177–183.

Kusumoputo, B., Irwanto, P., & Jatmiko, W. (2002). Optimization of fuzzy-neural structure through genetic algorithm and its application in artificial odor recognitions. *In Circuits and systems, 2002, APCCAS'02, 2002 Asia-Pacific conference, vol. 2* (pp. 47–51).

Lane, S. H., Handelman, D. A., & Gelfand, J. J. (1992). Theory and development of higher-order CMAC neural networks. *IEEE Control Systems Magazine, 12*, 23–30.

Lee, M. A., & Takagi, H. (1993). Integrating design stages of fuzzy systems using genetic algorithms. In *Proceedings of IEEE international conference on fuzzy systems, vol. 1* (pp. 612–617).

Lee, H. M., Chen, C. M., Chen, J. M., & Jou, Y. L. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 31*, 426–432.

Lin, C.-J., Chen, C.-H., & Lee, C.-Y. (2004). A self-adaptive quantum radial basis function network for classification applications. In *IEEE international joint conference on neural networks* (pp. 3263–3268) [disk].

Lin, C. T., & Lee, C. S. G. (1996). *Neural fuzzy systems: a neural-fuzzy synergism to intelligent systems*. Englewood Cliffs, NJ: Prentice-Hall [with disk].

Lovel, B. C., & Bradley, A. P. (1996). The multiscale classifier. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*, 124–137.

Merz, P., & Freisleben, B. (2000). Fitness landscape analysis and genetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computing, 4*(4), 337–352.

Miller, W. T., Hewes, R. P., Glanz, F. H., & Graft, L. G. (1990). Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. *IEEE Transactions on Robotic and Automation, 6*, 1–6.

Nauck, D., & Kruse, R. (1997). A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems, 89*(3), 277–288.

Reay, D. (1995). Nonlinear channel equalization using associative memory neural networks. In *Proceedings of international workshop on applied neural networks and telecommunications* (pp. 17–24).

Setiono, R., & Liu, H. (1997). Neural-network feature selector. *IEEE Transactions on Neural Networks, 8*, 654–662.

Sugeno, M., & Kang, G. T. (1988). Structure identification of a fuzzy model. *Fuzzy Sets and Systems, 28*(1), 15–33.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, SMC-15*, 116–132.

Tung, W. L., & Quek, C. (2002). GenSoFNN: A generic self-organizing fuzzy neural network. *IEEE Transactions on Neural Networks, 13*(5), 1075–1086.

Wang, J. S., & George Lee, C. S. (2002). Self-adaptive neuro-fuzzy inference systems for classification applications. *IEEE Transactions on Fuzzy Systems, 10*(6), 790–801.

Wang, L. X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 22*(6), 1414–1427.