Applied Soft Computing 9 (2009) 775–785

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A novel parametric fuzzy CMAC network and its applications

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

[a] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung County 411, Taiwan, ROC
[b] Department of Computer Science and Information Engineering, Nankai University of Technology, Nantou County 542, Taiwan, ROC

ARTICLE INFO

Article history:
Received 26 October 2005
Received in revised form 19 October 2007
Accepted 12 June 2008
Available online 21 October 2008

Keywords:
TSK-type fuzzy model
Cerebellar model articulation controller (CMAC)
Self-clustering
Backpropagation
Chaotic
Approximation

ABSTRACT

This paper shows fundamentals and applications of the novel parametric fuzzy cerebellar model articulation controller (P-FCMAC) network. It resembles a neural structure that derived from the Albus CMAC algorithm and Takagi–Sugeno–Kang parametric fuzzy inference systems. The Gaussian basis function is used to model the hypercube structure and the linear parametric equation of the network input variance is used to model the TSK-type output. A self-constructing learning algorithm, which consists of the self-clustering method (SCM) and the backpropagation algorithm, is proposed. The proposed the SCM 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 of things such as the number of clusters present in a data set. The backpropagation algorithm is used to tune the adjustable parameters. Illustrative examples were conducted to show the performance and applicability of the proposed model.

1. Introduction

In 1975, the cerebellar model articulation controller (CMAC) developed by Albus [1,2], is an artificial neural network inspired by the cerebellum. 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, and easier hardware implementation, etc. Because of the simple structure and fast learning speed of the CMAC network, it has been successfully applied in many fields, such as identification [3], control [4], pattern recognition [5], image processing [6], and equalization [7]. However, there are several limitations for the Albus' CMAC network. Using this technique, 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. The CMAC network can be viewed as a basis function network that uses plateau basis functions. In addition to the difference on the type of basis functions, another major difference of the CMAC network from the radial or Gaussian basis function networks and wavelet networks is that the CMAC network its basis functions each restricted to a local area, i.e., a hypercube. To compute the output of the network for given input data point, only those basis functions assigned to the hypercubes covering the input data point are needed. Then 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 Reay [8], and fuzzy membership functions by Lin [9], etc. In this paper, we use mathematical equations to describe the CMAC network with Gaussian basis functions as receptive field functions.

Secondly, the CMAC network needs an extraordinarily big memory space to solve high-dimensional problems for implementation [3]. Thus, 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) [10], linear vector quantization (LVQ) [11], fuzzy Kohonen partitioning (FKP), and pseudo-FKP [12], to perform cluster analysis. However, such clustering techniques require *prior* knowledge of things such as the number of clusters present in a data set. To solve the above problem, online-based cluster techniques were proposed [13,14]. But there are still problems with these methods; that is, the clustering methods 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-constructing method (SCM) to partition the input space of the CMAC network.

A neural fuzzy network consists of a set of fuzzy IF–THEN rules that describe the input–output mapping relationship of the

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

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.06.004

network. The antecedents of fuzzy rules partition the input space into a number of linguistic term sets while the consequent constituent can be chosen as a fuzzy membership function (Mamdani-type fuzzy model) [15], a singleton value, or a function of a linear combination of input variables (i.e., TSK-type fuzzy model) [16–18]. No matter which type of neural fuzzy networks is chosen, different consequent constituents result in different types of fuzzy models. Many researchers [17,18] 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. In this paper, we use a TSK-type fuzzy model to represent the weight vector of the CMAC network.

This work presents a new network structure, mainly derived from the CMAC algorithm and Takagi–Sugeno–Kang (TSK) parametric fuzzy inference systems [19,20]. The so-called parametric fuzzy CMAC (P-FCMAC) network resembles the original CMAC proposed by Albus, in the sense that it is a local network, i.e., for a given input vector, only a few of the networks nodes (or hypercube cells) will be active and will effectively contribute to the corresponding network output. In addition, the internal mapping structure is built in such a way that it implements, for each CMAC memory locations, one linear parametric equation of the network input variance.

The rest of this paper is organized as follows. The fundamentals of the P-FCMAC network are detailed in Section 2. We also present the difference between the conventional CMAC network and the proposed P-FCMAC network. Section 3 presents the learning algorithm for the P-FCMAC network. The various applications of the proposed network are shown in Section 4. Section 5 shows the conclusion of the computer simulation for the P-FCMAC network.

2. The parametric fuzzy CMAC network

In this section, we will review the architecture of the CMAC network and describe the proposed P-FCMAC network, individually. The active principle from network input to network output and the difference for these two networks are in detail here.

2.1. The CMAC network

The CMAC network [1] is a local network implies for a given input vector, only a few of the networks nodes (or hypercube cells) will be active and will effectively contribute to the corresponding network output. The architecture of the CMAC network is shown in Fig. 1. 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. In the case of the CMAC network, it 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. For the systems with multiple output dimensions, this CMAC network can be extended directly. The function $S(x)$ maps each point x in the input space onto an association vector $\alpha = S(x) \in A$ that has N_L nonzero elements ($N_L < N_A$). For a conventional CMAC network, the association vector contains only binary elements, either zero or one. The function $P(\alpha)$ computes a scalar output y by projecting the association vector onto a vector w of adjustable weights so that the scalar output y can be obtained by evaluating the *inner product* of the two vectors α and w. Then the actual output y is derived as follows:

$$y = P(\alpha) = \alpha^T w = \sum_{j=1}^{N_L} \alpha_j w_j \tag{3}$$

where α_j represents the jth element of the association vector, w_j represents the jth element of the weight vector, and N_L denotes the number of neurons.

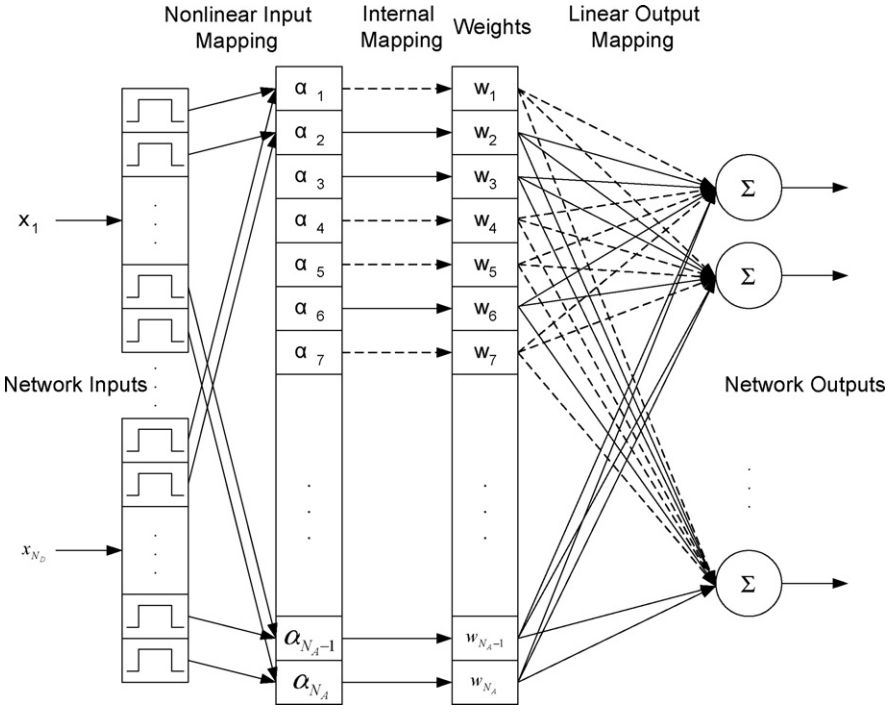


Fig. 1. The architecture of the CMAC network.

Let us take a two-dimensional (2D) input vector, or the so-called two-dimensional CMAC (2D CMAC), as an example. The structure of a 2D CMAC is shown in Fig. 2. The input vector is defined by two input variables, x_1 and x_2, with quantized into three discrete regions, call *blocks*. It is noted that the width of blocks affects the generalization capability of the CMAC network. For the first way of quantization, the variable x_1 is divided into blocks A, B, and C and the variable x_2 is divided into blocks a, b, and c. The areas Aa, Ab, Ac, Ba, Bb, Bc, Ca, Cb, and Cc formed by quantized regions are called *hypercubes*. By shifting each block a small interval, different hypercubes can be obtained. In Fig. 2, there are 27 hypercubes used to distinguish 49 different states in the 2D CMAC. For example, let the hypercubes Bb, Ee, and Hh be addressed by the state $(x_1, x_2) = (3, 3)$, only these three hypercubes are one, and the others are zero. About above mention, the trained data is stored into these regions.

2.2. The architecture of the P-FCMAC network

In this section, we propose a new parametric fuzzy CMAC network. The architecture of the P-FCMAC network is illustrated in Fig. 3, 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 non-linear function $y = f(x)$ by using two primary mappings, $S(x)$ and $P(\alpha)$. 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 $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{N_A})$, where $0 \le \alpha \le 1$ 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:

$$\mu(x) = \mathrm{e}^{-(x-m/\sigma)^2} \tag{4}$$

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:

$$\alpha_j = \prod_{i=1}^{N_D} e^{-(x_i - m_{ij}/\sigma_{ij})^2} \tag{5}$$

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:

$$y = \frac{\sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\sum_{j=1}^{N_L} \alpha_j} \tag{6}$$

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

$$a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i \tag{7}$$

where a_{0j} 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 learning algorithm will be proposed to determine the proper network structure and its adjustable parameters.

3. The learning algorithm for P-FCMAC network

In this section, a learning algorithm, which consists of an input space partition scheme and a parameter learning scheme, is developed for constructing the P-FCMAC network. The flow diagram of the learning scheme for the P-FCMAC network is

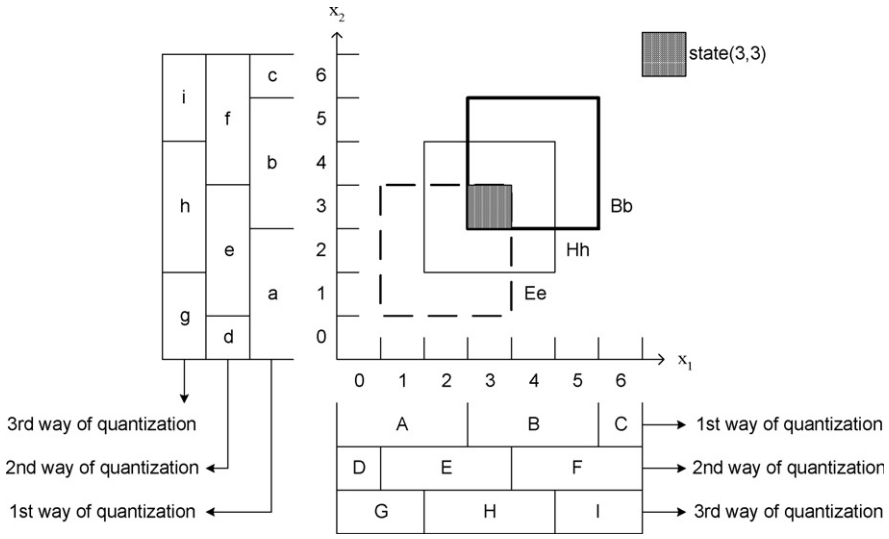


Fig. 2. The structure of a 2D CMAC.

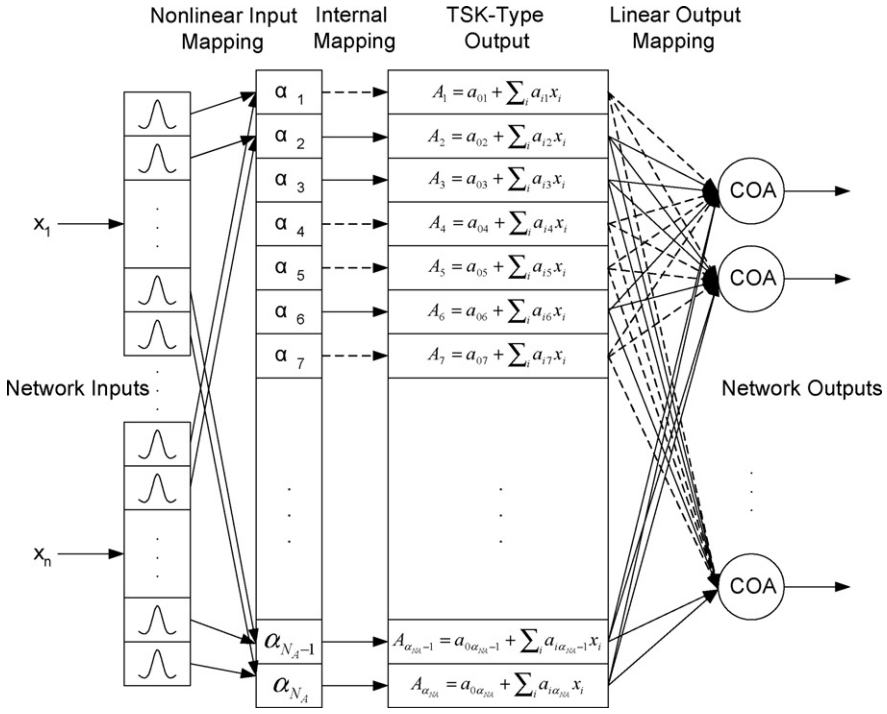


Fig. 3. The architecture of the P-FCMAC network.

shown in Fig. 4. 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 method to appropriately determine the various distributions of the input training data. After the self-clustering method, the number of hypercube cells is determined. That is, we can obtain the initial m and σ of receptive field functions by using SCM. Second, the parameter learning scheme is based on supervised learning algorithms. The gradient descent learning algorithm is used to adjust the free parameters. To minimize a given cost function, the m and σ of the receptive field functions and the parameters a_{0j} and a_{ij} of the TSK-type output vector are adjusted using the backpropagation algorithm. 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 SCM clustering method.

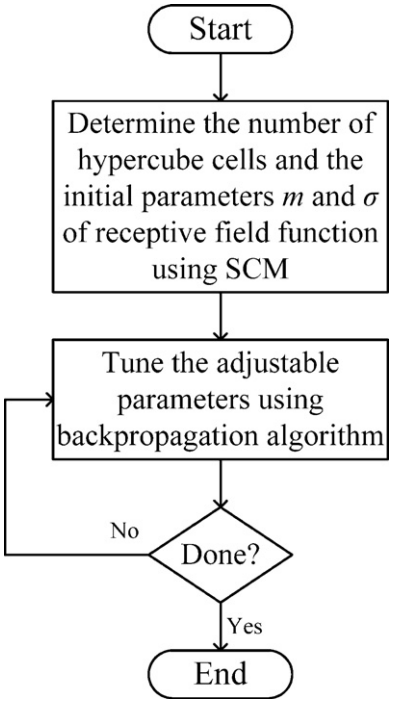


Fig. 4. The flow diagram of the learning scheme for the P-FCMAC network.

3.1. A self-clustering method

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 the self-clustering method, is proposed to implement scatter partitioning of the input space. Without any optimization, the proposed SCM clustering method 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, Dc and Wd, 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, Dc, reaches the value that is equal to the threshold value D_{thr}.

The details of the SCM algorithm are described in the following steps:

- *Step 0*: Create the first cluster C_1 by simply taking the position of the first input data as the first cluster mean Cc_{i_1} where i means the ith input variables. And setting a dimension distance value to 0 (see Fig. 5(a)).
- *Step 1*: If all of the training input data have been processed, the algorithm is finished. Otherwise, the current input example, $x_i[k]$, is taken and the distances between this input example and all already created cluster mean Cc_{i_j} are calculated:

$$D_{i\_j}[k] = ||x_i[k] - Cc_{i\_j}|| \tag{8}$$

where $j = 1, 2, \ldots, R$ denotes the jth cluster, $k = 1, 2, 3, \ldots, N$ represents the kth input, and $i = 1, 2, \ldots, n$ represents the ith dimension.

- *Step 2*: If the distance calculation in Eq. (8) is equal to or less than all of the dimension distances CD_{i_j} that represent the ith dimension distance in the jth cluster (set to 0 initially), then the current input example belongs to a cluster with the minimum distance:

$$Dmin_j[k] = \min\left(\sum_{i=1}^{n} \|x_i[k] - Cc_{i\_j}\|\right) \tag{9}$$

$$Dmin_{i\_j}[k] = \|x_i[k] - Cc_{i\_j}\| \tag{10}$$

where j in Eq. (10) represents the jth cluster that is computed using Eq. (9). The use of Eqs. (9) and (10) is to find the minimum sum of all the dimension distances in a cluster with the kth input data. The constraint is described as follows:

$$Dmin_{i\_j}[k] \le CD_{i\_j} \tag{11}$$

If no new clusters are created or no existing clusters are updated (the cases of $(x_1[4], x_2[4])$ and $(x_1[6], x_2[6])$ in Fig. 5(b)), the algorithm returns to Step 1. Otherwise, the algorithm goes to the next step.

- *Step 3*: Find a cluster from all existing cluster centers by calculating $S_{i\_j}[k] = D_{i\_j}[k] + CD_{i\_j}, j = 1, 2, \ldots, R$, and then choosing the cluster center with the minimum value:

$$Smin_{i\_j}[k] = Dmin_{i\_j}[k] + CD_{i\_j} \tag{12}$$

where $j = 1, 2, \ldots, R$.

In Eqs. (10) and (11), the minimum distance from any cluster mean to the examples that belong to this cluster is not greater than the threshold D_{thr}, though the algorithm does not keep any information of passed examples. However, we find that the formulation only considers the distance between the input data and the cluster mean in Eq. (12). But the special situation [6] shows that the distances between the given point $x_i[10]$ and both cluster means Cc_{i_1} and Cc_{i_2} are the same as in Fig. 6. In the aforementioned technique, the cluster C_2, which has small dimension distances CD_{i_2}, will be selected to expand according to Eq. (12). However this causes a problem in that the cluster numbers increase quickly. To avoid this problem, we state a condition, as follows.

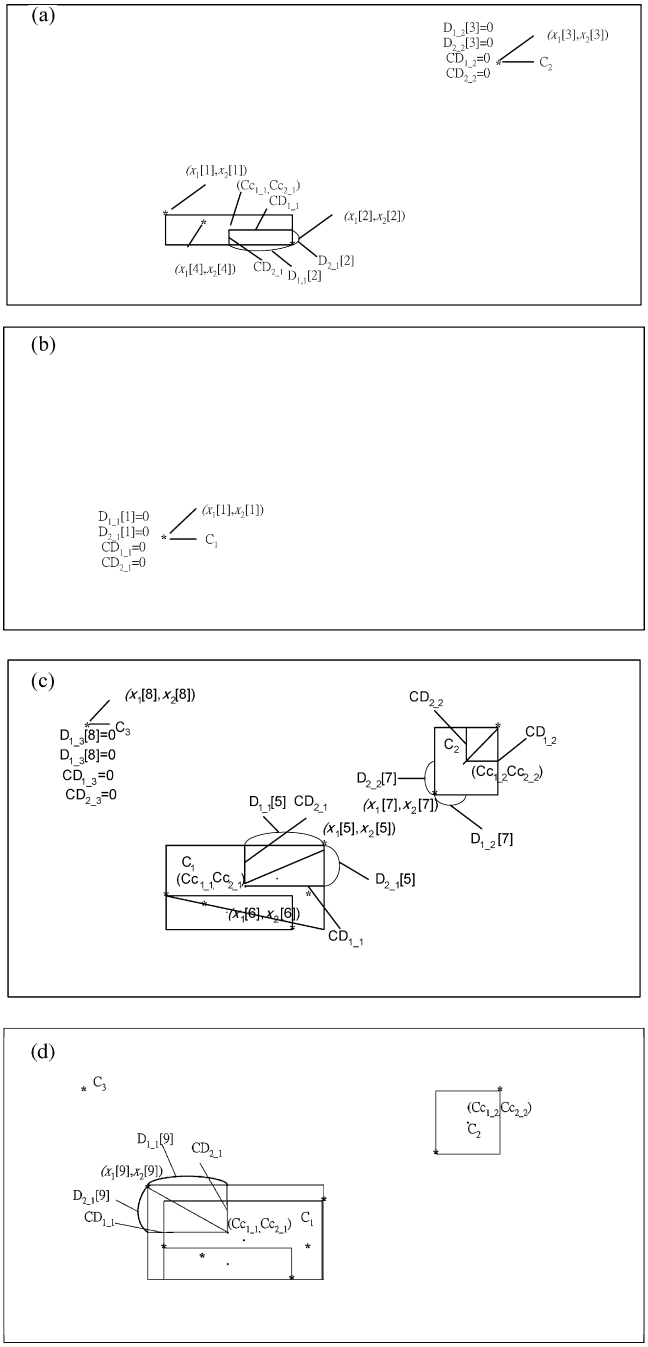


Fig. 5. A brief clustering process using SCM with samples in 2D space.

If there are two $Dmin_j$ [10] compute in Eq. (9) that $Dmin_1[10] = Dmin_2[10]$ and $(CD_{1\_1} + CD_{2\_1} > CD_{1\_2} + CD_{2\_2})$

Then $Dmin_{1\_1}[10] = D_{1\_1}[10]$ (13)

$$D_{min2\_1}[10] = D_{2\_1}[10] \tag{14}$$

where $Dmin_1[10]$ represents the minimum distance between the 10th input data and the mean of the 1st cluster that is calculated by Eq. (11); $Dmin_{2\_1}[10]$ represents dimension distance between the 2nd dimension of the 10th input data and the 2nd dimension mean of the 1st cluster that is calculated by Eq. (12); $D_{2\_1}[10]$ represents dimension distance between the 2nd dimension of the 10th input

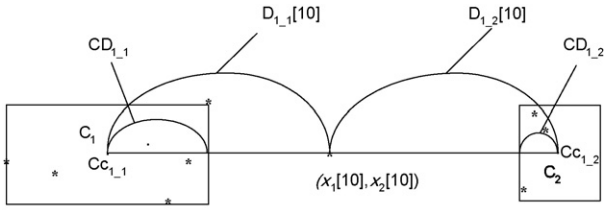


Fig. 6. The special case of SCM.

data and the 2nd dimension mean of the 1st cluster that is calculated by Eq. (10). In Eqs. (13) and (14), we find that when the distances between the input data and both clusters are the same, the formulation will choose the cluster that has the large dimension distance $CD_{1_1}$and CD_{2_1}.

- *Step 4*: If $Smin_{i\_j}[k]$ in Eq. (12) is greater than D_{thr}, the input example $x_i[k]$does not belong to any existing cluster. A new cluster is created in the same way as described in Step 0 (the cases of (x_1[3], x_2[3]) and (x_1[8], x_2[8]) in Fig. 5(c)), and the algorithm returns to Step 1.
- *Step 5*: If $Smin_{i\_j}[k]$ is not greater than D_{thr}, the cluster is updated by moving its mean, Cc_{i_j}, and increasing the value of its dimension distances. The new mean is moved to the point on the line connecting the input data, and the distance from the new mean to the point is equal to its dimension distance (the cases of (x_1[5], x_2[5]) and (x_1[9], x_2[9]) in Fig. 5(d)). The details for updating the equations are as follows:

$$\begin{aligned} &\text{If } CD_{i\_j} < \frac{x_i[k] - Cc_{i\_j} + CD_{i\_j}}{2} \\ &\text{Then } CD_{i\_j} = \frac{x_i[k] - Cc_{i\_j} + CD_{i\_j}}{2} \end{aligned} \tag{15}$$

$$Cc_{i\_j} = x_i[k] - CD_{i\_j} \quad \text{if } Cc_{i\_j} >= x_i[k] \tag{16}$$

$$Cc_{i\_j} = x_i[k] + CD_{i\_j} \quad \text{if } Cc_{i\_j} < x_i[k] \tag{17}$$

where $k = 1, 2, 3, \ldots, N$ represents the kth input, j represents the jth cluster that has a minimum distance in Eq. (9), x represents the input data, and i represents the ith dimension. After this step is performed, the algorithm returns to step 1.

The threshold parameter D_{thr} is an important parameter in the input space partition scheme. A low threshold value leads to the learning of fine clusters (such that many hypercube cells are generated), whereas a high threshold value leads to the learning of coarse clusters (such that fewer hypercube cells are generated). Therefore, the selection of the threshold value D_{thr} critically affects the simulation results, and the threshold value is determined by practical experimentation or trial-and-error tests.

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}. The total number of tuning parameters for the multi-input single-output P-FCMAC network is $2N_DN_L + 4N_L$, where N_D and N_L denote the number of inputs and hypercube cells, respectively. The parameter learning algorithm of the P-FCMAC network uses the supervised gradient descent method to modify these parameters. When we consider the single output case for clarity, our goal is to minimize the cost function E, defined as follows:

$$E(t) = \frac{1}{2}\left(y^d(t) - y(t)\right)^2 \tag{18}$$

where $y^d(t)$ denotes the desired output at time t and $y(t)$ denotes the actual output at time t. Then their parameter learning algorithm, based on backpropagation, is described in detail as follows.

The TSK-type outputs are updated according to the following equation:

$$a_{0j}(t+1) = a_{0j}(t) + \Delta a_{0j} \tag{19}$$

$$a_{ij}(t+1) = a_{ij}(t) + \Delta a_{ij} \tag{20}$$

where a_{0j} denotes the proper scalar, a_{ij} denotes the proper scalar coefficient of the ith input dimension, and j denotes the jth element of the TSK-type output vector for $j = 1, 2, \ldots, N_L$. The elements of the TSK-type output vectors are updated by the amount

$$\Delta a_{0j} = \eta e \frac{\partial y}{\partial a_{0j}} = \eta e \frac{\alpha_j}{\sum_{j=1}^{N_L} \alpha_j} \tag{21}$$

$$\Delta a_{ij} = \eta e \frac{\partial y}{\partial a_{ij}} = \eta e \frac{x_i \alpha_j}{\sum_{j=1}^{N_L} \alpha_j} \tag{22}$$

where η is the learning rate, between 0 and 1, and e is the error between the desired output and the actual output, $e = y^d - y$.

The receptive field functions are updated according to the following equation:

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij} \tag{23}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij} \tag{24}$$

where i denotes the ith input dimension for $i = 1, 2, \ldots, n$, m_{ij} denotes the mean of the receptive field functions, and σ_{ij} denotes the variance of the receptive field functions. The parameters of the receptive field functions are updated by the amount

$$\begin{aligned} \Delta m_{ij} &= \eta e \frac{\partial y}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial m_{ij}} \\ \Delta m_{ij} &= \eta e \frac{\left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right) \sum_{j=1}^{N_L} \alpha_j - \sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\left(\sum_{j=1}^{N_L} \alpha_j\right)^2} \alpha_j \frac{2(x_i - m_{ij})}{\sigma_{ij}^2} \end{aligned} \tag{25}$$

$$\begin{aligned} \Delta \sigma_{ij} &= \eta e \frac{\partial y}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial \sigma_{ij}} \\ \Delta \sigma_{ij} &= \eta e \frac{\left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right) \sum_{j=1}^{N_L} \alpha_j - \sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\left(\sum_{j=1}^{N_L} \alpha_j\right)^2} \alpha_j \frac{2(x_i - m_{ij})^2}{\sigma_{ij}^3} \end{aligned} \tag{26}$$

where η is the learning rate of the mean and the variance for the receptive field functions.

4. Illustrative examples

In this section, we compare the performance of the P-FCMAC network with other various existing models on three applications.

There are two parameter learning schemes are used during the training process. The first parameter learning scheme (i.e., Scheme 1) represents that only the TSK-type consequent parameters are tuned by gradient descent method while the receptive field functions are fixed. The second parameter learning scheme (i.e., Scheme 2) represents that both the TSK-type consequent parameters and the receptive field functions are tuned by gradient descent method. The first example is the prediction of the chaotic time series [21]. The second example is the approximation of the nonlinear systems [22]. The third example is the identification of the nonlinear systems that are described in [23].

5. Example 1: prediction of the chaotic time series

The Mackey–Glass chaotic time series $x(t)$ in consideration here is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{27}$$

Cowder [21] extracted 1000 input–output data pairs $\{x, y^d\}$ which consist of four past values of $x(t)$, i.e.

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \tag{28}$$

where $\tau = 17$ and $x(0) = 1.2$. There are four inputs to the P-FCMAC network, corresponding to these values of $x(t)$, and one output representing the value $x(t + \Delta t)$, where Δt is a time prediction into the future. The first 500 pairs, from $x(1)$ to $x(500)$, are the training data set, while the remaining 500 pairs, from $x(501)$ to $x(1000)$, are the testing data set used for validating the proposed method.

In this example, the initial threshold value in the SCM is 0.7, and the learning rate is $\eta = 0.01$. After the SCM clustering process, there are four hypercube cells generated. After 500 epochs training for Scheme 1 method (i.e., only the TSK-type consequent parameters are tuned), the final trained rmse (root mean square error) of the prediction output approximates 0.0094. Using Scheme 2 method (i.e., both the TSK-type output parameters and the receptive field functions are tuned), the final trained rmse of the prediction output approximates 0.0048 after 500 epochs. The prediction outputs of the chaotic time series from $x(501) - x(1000)$, when 500 training data from $x(1) - x(500)$ were used for Schemes 1 and 2 methods, are shown in Fig. 7(a) and (c). The solid line represents the output of the time series, and the dotted line represents the output of the P-FCMAC network. Fig. 7(b) and (d) shows the prediction errors from $x(501) - x(1000)$ between the desired output and the P-FCMAC network output using Schemes 1 and 2 methods. The learning curves of Schemes 1 and 2 methods are shown in Fig. 8.

Table 1 shows the comparison results of the prediction performance among various predictors. The previous results were taken from [24,25]. The performance of the very compact fuzzy system obtained by the P-FCMAC network is better than all previous works. In addition, the proposed network takes four hypercube cells and its size only is 48. The comparative table can demonstrate that the proposed P-FCMAC network performs better than other existing models.

5.1. Example 2: approximation of a Sugeno's nonlinear function

Let us consider the nonlinear system, presented by Sugeno and Yasukawa in [22], used widely in the literature as a

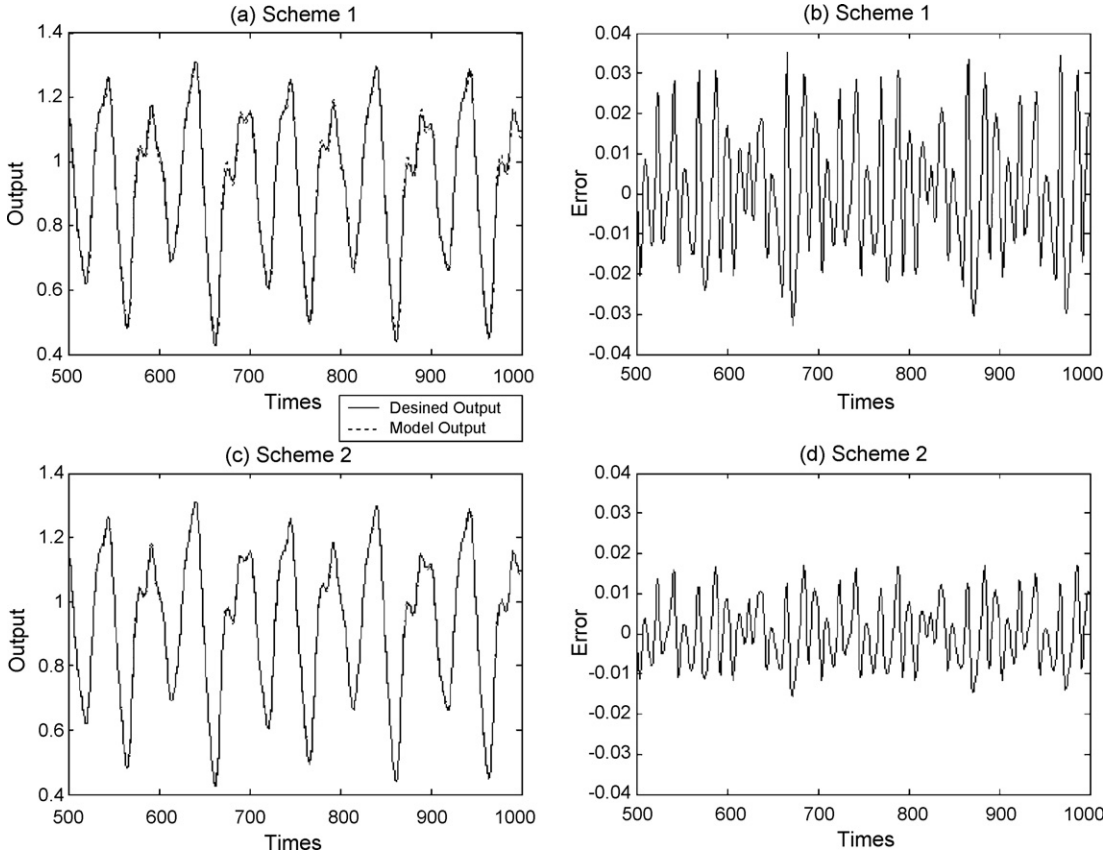


Fig. 7. (a) Simulation results of the time series from $x(501) - x(1000)$ for Scheme 1 method. (b) Prediction errors from $x(501) - x(1000)$ between the desired output and the P-FCMAC network output for Scheme 1 method. (c) Simulation results of the time series from $x(501) - x(1000)$ for Scheme 2 method. (d) Prediction errors from $x(501) - x(1000)$ between the desired output and the P-FCMAC network output for Scheme 2 method.

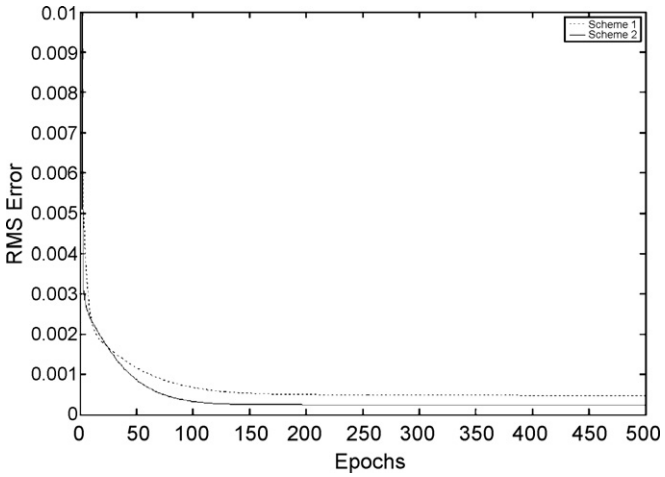


Fig. 8. Learning curves of Schemes 1 and 2 parameter learning methods in Example 1.

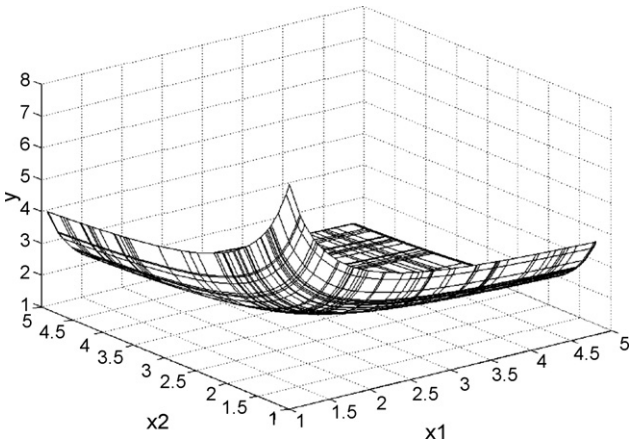


Fig. 9. Desired input–output relation of the nonlinear system.

benchmark

$$z = (1 + x_1^{-2} + x_2^{-1.5})^2, \quad 1 \le x_1, x_2 \le 5 \tag{29}$$

The fuzzy system identification is based on fifty samples reported in [22]. The original samples were four inputs and one output with two dummy inputs. The P-FCMAC network discarded the two dummy inputs. We show a desired input–output graph of this nonlinear function in Fig. 9. Table 2 represents the two-input one-output training data.

In this example, the initial threshold value in the SCM is 2.6, and the learning rate is $\eta = 0.01$. After the SCM clustering process, there are five hypercube cells generated. Using the first and second parameter learning schemes, the final trained rmse of the outputs approximates 0.0051 and 0.0035 after 5000 epochs. The desired output and the P-FCMAC network output (using Schemes 1 and 2 methods) are shown in Fig. 10(a) and (c). Fig. 10(b) and (d) shows the errors between the desired output and the P-FCMAC network output using Schemes 1 and 2 methods. The learning curves of Schemes 1 and 2 methods are shown in Fig. 11.

Table 3 reports results found in the literature plus the error found by the P-FCMAC network. The performance of the very compact fuzzy system obtained by the P-FCMAC network is better than all previous works. In addition, the proposed network takes five hypercube cells and its size only is 40. It can be concluded that the proposed model obtains better results than some existing models.

5.2. Example 3: identification of a nonlinear system

In this example, a nonlinear system with an unknown nonlinear function, which is approximated by the P-FCMAC network as shown in Fig. 12(b), is a model. First, some of training data from the unknown function are collected for an off-line initial learning process of the P-FCMAC network. After off-line learning, the trained P-FCMAC network is applied to the nonlinear system to replace the unknown nonlinear function for on-line test.

Consider a nonlinear system in [23] governed by the difference equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \tag{30}$$

We assume that the unknown nonlinear function has the form:

$$g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \tag{31}$$

For off-line learning, twenty-one training data pairs are provided in Table 4 using Eq. (30). The off-line learning configuration of the twenty-one training data points is shown in Fig. 12(a). And the on-line test configuration of the 1000 data points is shown in Fig. 12(b) that using the difference equation is defined as

$$\hat{y}(k+1) = 0.3y(k) + 0.6y(k-1) + \hat{f}[u(k)] \tag{32}$$

where $\hat{f}[u(k)]$ is the approximated function for $g[u(k)]$ by the P-FCMAC network and $\alpha_0 = 0.3$, $\alpha_1 = 0.6$. The error is defined as

Table 1
Comparison results of various existing models on chaotic time series prediction.

Methods	RMSE	Methods	RMSE
P-FCMAC	0.0094 (Scheme 1) 0.0048 (Scheme 2)	6th-order Polynomial	0.04
GEFREX [25]	0.0061	Cascade Correlation NN	0.06
ANFIS [17]	0.007	Min operator	0.09
Kim and Kim [24]	0.026	Wang (product operator)	0.091

Table 2
The two-input and one-output training data for the approximation example.

No.	x_1	x_2	y	No.	x_1	x_2	y
1	1.40	1.80	3.70	26	2.00	2.06	2.52
2	4.28	4.96	1.31	27	2.71	4.13	1.58
3	1.18	4.29	3.35	28	1.78	1.11	4.71
4	1.96	1.90	2.70	29	3.61	2.27	1.87
5	1.85	1.43	3.52	30	2.24	3.74	1.79
6	3.66	1.60	2.46	31	1.81	3.18	2.20
7	3.64	2.14	1.95	32	4.85	4.66	1.30
8	4.51	1.52	2.51	33	3.41	3.88	1.48
9	3.77	1.45	2.70	34	1.38	2.55	3.14
10	4.84	4.32	1.33	35	2.46	2.12	2.22
11	1.05	2.55	4.63	36	2.66	4.42	1.56
12	4.51	1.37	2.80	37	4.44	4.71	1.32
13	1.84	4.43	1.97	38	3.11	1.06	4.08
14	1.67	2.81	2.47	39	4.47	3.66	1.42
15	2.03	1.88	2.66	40	1.35	1.76	3.91
16	3.62	1.95	2.08	41	1.24	1.41	5.05
17	1.67	2.23	2.75	42	2.81	1.35	3.11
18	3.38	3.70	1.51	43	1.92	4.25	1.92
19	2.83	1.77	2.40	44	4.61	2.68	1.63
20	1.48	4.44	2.44	45	3.04	4.97	1.44
21	3.37	2.13	1.99	46	4.82	3.80	1.39
22	2.84	1.24	3.42	47	2.58	1.97	2.29
23	1.19	1.53	4.99	48	4.14	4.76	1.33
24	4.10	1.71	2.27	49	4.35	3.90	1.40
25	1.65	1.38	3.94	50	2.22	1.35	3.39

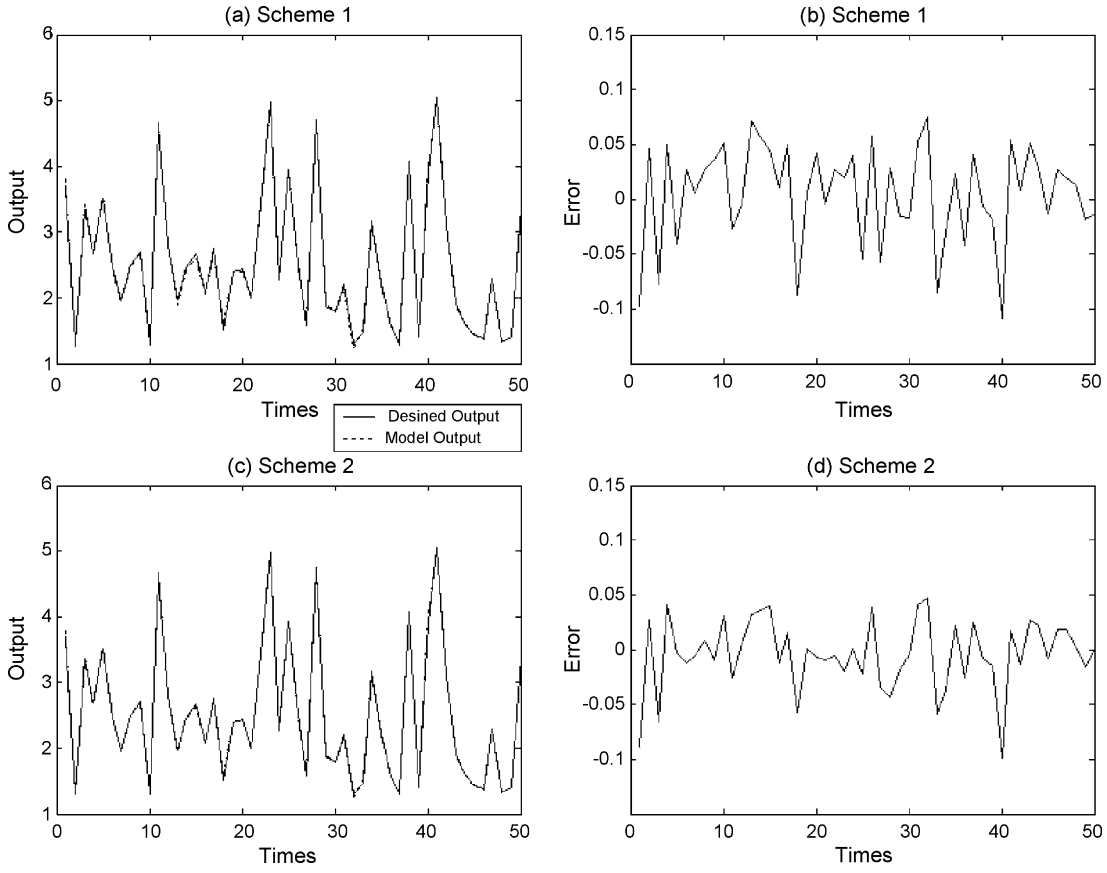


Fig. 10. (a) Simulation results of the approximation problem for Scheme 1 method. (b) Errors between the desired output and the P-FCMAC network output for Scheme 1 method. (c) Simulation results of the approximation problem for Scheme 2 method. (d) Errors between the desired output and the P-FCMAC network output for Scheme 2 method.

follows [28]:

$$\text{error}(t) = \left(y^d(t) - y(t)\right)^2 \tag{33}$$

In this example, the initial threshold value in the SCM is 0.15, and the learning rate is $\eta = 0.01$. After the SCM clustering process, there are eleven hypercube cells generated. Using the first and second parameter learning schemes, the final trained error of the output approximates 0.00057 and 0.00024 after 300 epochs. The numbers of the adjustable parameters of the trained P-FCMAC network are 66.

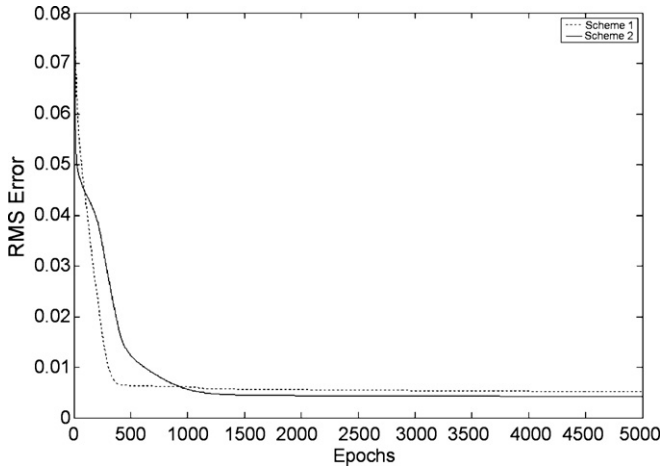


Fig. 11. Learning curves for Schemes 1 and 2 parameter learning methods in Example 2.

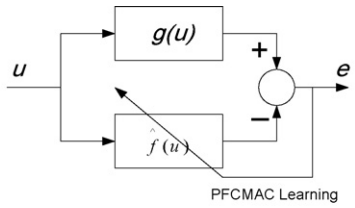
For on-line testing, we assume that the series-parallel model shown in Fig. 12(b) is driven by $u(k) = \sin(2\pi k/250)$. The test results of the P-FCMAC network are shown in Fig. 13(a) and (c) for Schemes 1 and 2 methods. The errors between the desired output and the P-FCMAC network output are shown in Fig. 13(b) and (d)

Table 3
Comparison results of various existing models on the approximation example.

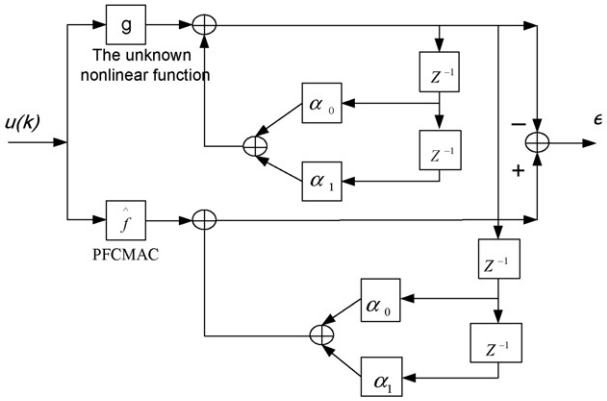
Methods	MSE	Methods	MSE
P-FCMAC	0.0051 (Scheme 1) 0.0035 (Scheme 2)	Sugeno and Yasukawa [22]	0.01
Emami et al. [26]	0.004	Delgado et al. [27]	0.231
Lin et al. [28]	0.005	Genetic algorithm et al. [29]	2.98

Table 4
Training data and approximated data obtained using the P-FCMAC network for 300 epochs.

u	$g(u)$	$\hat{f}(u)$	u	$g(u)$	$\hat{f}(u)$
−1.0000	−0.0000	−0.000357	0.1000	0.5281	0.526161
−0.9000	−0.5281	−0.528020	0.2000	0.6380	0.640683
−0.8000	−0.6380	−0.628281	0.3000	0.4781	0.472271
−0.7000	−0.4781	−0.480252	0.4000	0.3943	0.400248
−0.6000	−0.3943	−0.392093	0.5000	0.4000	0.401699
−0.5000	−0.4000	−0.405011	0.6000	0.3943	0.390031
−0.4000	−0.3943	−0.390852	0.7000	0.4781	0.471195
−0.3000	−0.4781	−0.481167	0.8000	0.6380	0.648287
−0.2000	−0.6380	−0.635818	0.9000	0.5281	0.516553
−0.1000	−0.5281	−0.531204	1.0000	0.0000	0.007459
0.0000	0.0000	0.002119			



(a) Off-line learning by twenty-one training data in Table IV.



(b) On-line testing for real $u(k) = \sin(2\pi k/250)$

Fig. 12. The series-parallel identification model. Off-line learning by twenty-one training data in Table 4. On-line testing for real $u(k) = \sin(2\pi k/250)$.

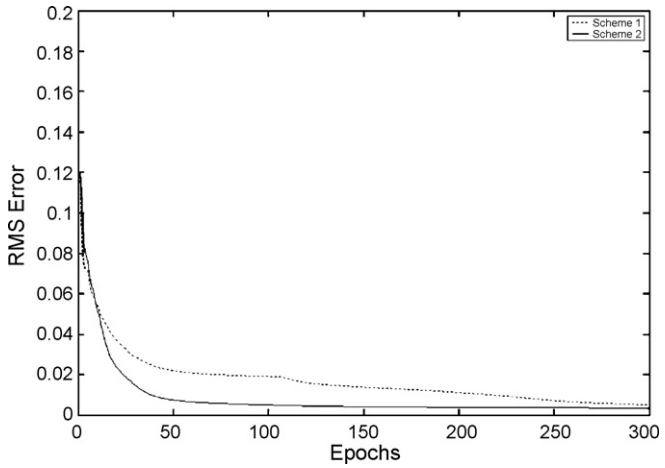


Fig. 14. Learning curves for Schemes 1 and 2 parameter learning methods in Example 3.

for Schemes 1 and 2 methods. The learning curves of Schemes 1 and 2 methods are shown in Fig. 14. Fig. 13 can prove that the P-FCMAC network successfully approximates the unknown nonlinear function.

Table 5 shows the comparison the learning result among various models. The previous results were taken from [30–33]. The performance of the very compact fuzzy system obtained by the P-FCMAC network is better than all previous works.

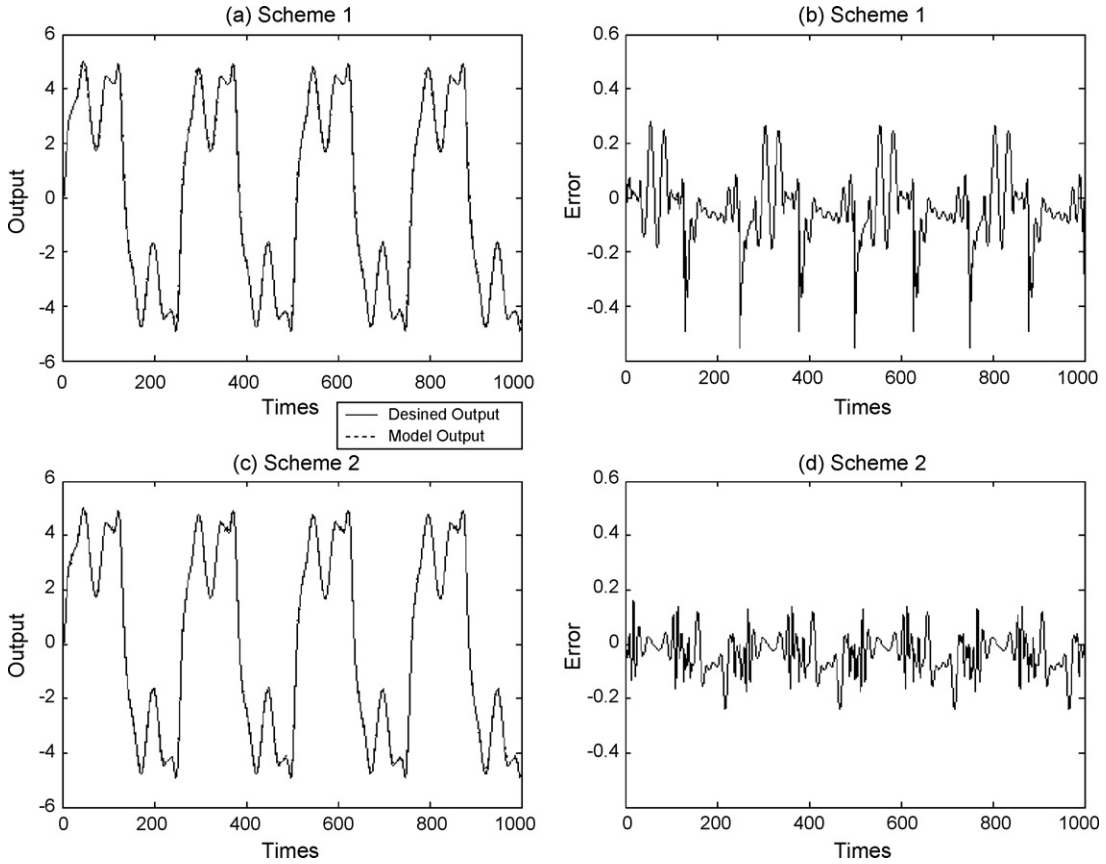


Fig. 13. (a) Outputs of the nonlinear system (solid line) and the identification model using the proposed network (dotted line) for Scheme 1 method. (b) Identification error of the approximated model for Scheme 1 method. (c) Outputs of the nonlinear system (solid line) and the identification model using the proposed network (dotted line) for Scheme 2 method. (d) Identification error of the approximated model for Scheme 2 method.

Table 5
Comparison results of the twenty-one training data for off-line learning.

Methods	Error	Methods	Error
P-FCMAC	0.00057 (Scheme 1) 0.00024 (Scheme 2)	Gradient descent [31]	0.2841
SGA-SSCP [30]	0.00028	MRDGA [32]	0.5221
Symbiotic evolution [33]	0.1997	Genetic algorithm et al. [29]	0.67243

6. Conclusion and future work

In this paper, a new parametric fuzzy CMAC network was proposed for dynamic system identification and prediction. 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. A learning algorithm was presented for constructing and adjusting the parameters. The proposed algorithm consists of the self-clustering method to perform input space partition and the backpropagation algorithm to perform parameter learning. With the proposed self-clustering method, a flexible partitioning of the input space is achieved to find the center and variance of the receptive field functions. 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 three examples given confirm the effectiveness of the proposed model.

Although the P-FCMAC network can perform better than other methods, an advanced topic should be addressed for the proposed P-FCMAC network. In this study, since the backpropagation algorithm is used to minimize the cost function, the results may reach the local minima solution. In future work, we will adopt the genetic algorithms (GA) to solve the local minima problem. GA is a global search technique. Because it simultaneously evaluates many points in the search space, it is more likely to converge toward the global solution.

Acknowledgement

This work was supported by National Science Council, R.O.C. under grant NSC95-2221-E-390-040-MY2.

References

[1] J.S. Albus, A new approach to manipulator control: the cerebellar model articulation controller (CMAC), Trans. ASME J. Dyn. Syst. Meas. Contr. (1975) 220–227.
[2] J.S. Albus, Data storage in the cerebellar model articulation controller (CMAC), Trans. ASME J. Dyn. Syst. Meas. Contr. (1975) 228–233.
[3] Z.J. Lee, Y.P. Wang, S.F. Su, A genetic algorithm based robust learning credit assignment cerebellar model articulation controller, Appl. Soft Comput. 4 (4) (2004) 357–367.
[4] J. Wang, C. Zhang, Y. Jing, Hybrid CMAC-PID controller in heating ventilating and air-conditioning system, in: Proceedings of the International Conference on Mechatronics and Automation, vols. 5–8, 2007, pp. 3706–3711.
[5] H.T. He, Y. Li, The research on flatness pattern recognition based on CMAC neural network, in: Proceedings of the International Conference on Machine Learning and Cybernetics, August 19–22, vol. 5, 2007, pp. 2745–2748.
[6] H.C. Lu, T. Tao, The treatment of image boundary effects in CMAC networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks, July 25–29, vol. 2, 2004, pp. 867–872.
[7] D. Reay, Nonlinear channel equalization using associative memory neural networks, in: Proceedings of the International Workshop on Applied Neural Networks Telecom, Stockholm, Sweden, (1995), pp. 17–24.
[8] D.S. Reay, CMAC and B-spline neural networks applied to switched reluctance motor torque estimation and control, in: Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society (IECON' 03), November 2–6, vol. 1, 2003, pp. 323–328.
[9] C.Y. Lee, C.J. Lin, H.J. Chen, A self-constructing fuzzy CMAC model and its applications, Informat. Sci.: Int. J. 177 (1) (2007) 264–280.
[10] S. Chen, D. Zhangm, Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure, IEEE Trans. Syst. Man Cyber.—Part B 34 (4) (2004) 1907–1916.
[11] C.E. Pedreira, Learning vector quantization with training data selection, IEEE Trans. Pattern Anal. Mach. Intell. 28 (1) (2006) 157–162.
[12] K.K. Ang, C. Quek, M. Pasquier, POPFNN-CRI (S): pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier, IEEE Trans. Syst. Man Cyber.—Part B: Cybernetics 33 (6) (2003) 838–849.
[13] W.L. Tung, C. Quek, GenSoFNN: a generic self-organizing fuzzy neural network, IEEE Trans. Neural Networks 13 (5) (2002) 1075–1086.
[14] N.K. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Systems 10 (2) (2002) 144–154.
[15] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cyber. 22 (6) (1992) 1414–1427.
[16] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cyber. vol. SMC-15 (1985) 116–132.
[17] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cyber. 23 (1993) 665–685.
[18] C.J. Lin, Y.J. Xu, The design of TSK-type fuzzy controllers using a new hybrid learning approach, Int. J. Adaptive Control Signal Process. 20 (1) (2006) 1–25.
[19] M. Sugeno, G.T. Kang, Structure identification of a fuzzy model, Fuzzy Sets Syst. 28 (1) (1988) 15–33.
[20] T. Takagi, M. Segeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 116–132.
[21] R.S. Crowder. Predicting the Mackey-Glass time series with cascade correlation learning. In: Proceedings of 1990 Connectionist Models Summer School, Carnegie Mellon University, 1990, pp. 117–123.
[22] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Trans. Fuzzy Syst. 1 (1) (1993) 7–31.
[23] L.X. Wang, Adaptive Fuzzy Systems and Control, Englewood Cliffs, NJ, Prentice-Hall, 1994.
[24] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 523–535.
[25] M. Russo, Genetic fuzzy learning, IEEE Trans. Evol. Comput. 4 (3) (2000) 259–273.
[26] M.R. Emami, I.B. Türksen, A.A. Goldberg, Development of a systematic methodology of fuzzy logic modeling, IEEE Trans. Fuzzy Syst. 6 (3) (1998) 346–361.
[27] M. Delgado, F. Gómez-Skarmeta, F. Martín, A fuzzy clustering based rapid prototyping for fuzzy rule-based modeling, IEEE Trans. Fuzzy Syst. 5 (2) (1997) 223–233.
[28] Y. Lin, G.A. Cunningham III, S.V Coggeshall, Using fuzzy partitions to create fuzzy systems from input–output data and set the initial weights in a fuzzy neural network, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 614–621.
[29] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proceedings of the 4th Conference on Genetic Algorithms, 1991, pp. 450–457.
[30] W.Y. Wan, Y.H. Li, Evolutionary learning of BMF fuzzy-neural networks using a reduced-form genetic algorithm, IEEE Trans. Syst. Man Cybern.—Part B 33 (6) (2003) 966–976.
[31] C.H. Wang, W.Y. Wang, T.T. Lee, P.S. Tseng, Fuzzy B-spline membership function (BMF) and its applications in fuzzy-neural control, IEEE Trans. Syst. Man Cybern. 25 (5) (1995) 841–851.
[32] W.A. Farag, V.H. Quintana, G. Lambert-Torres, A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems, IEEE Trans. Neural Networks 9 (5) (1998) 756–767.
[33] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Syst. Man Cybern.—Part B 30 (2) (2000) 290–302.