An Adaptive Input Data Space Parting Solution to the Synthesis of Neuro-Fuzzy Models

Sy Dzung Nguyen and Kieu Nhi Ngo

Abstract: This study presents an approach for approximation an unknown function from a numerical data set based on the synthesis of a neuro-fuzzy model. An adaptive input data space parting method, which is used for building hyperbox-shaped clusters in the input data space, is proposed. Each data cluster is implemented here as a fuzzy set using a membership function MF with a hyperbox core that is constructed from a min vertex and a max vertex. The focus of interest in proposed approach is to increase degree of fit between characteristics of the given numerical data set and the established fuzzy sets used to approximate it. A new cutting procedure, named *NCP*, is proposed. The *NCP* is an adaptive cutting procedure using a pure function Ψ and a penalty function τ for direction the input data space parting process. New algorithms named *CSHL*, *HLM1* and *HLM2* are presented. The first new algorithm, *CSHL*, built based on the cutting procedure *NCP*, is used to create hyperbox-shaped data clusters. The second and the third algorithm are used to establish adaptive neuro-fuzzy inference systems. A series of numerical experiments are performed to assess the efficiency of the proposed approach.

Keywords: Fuzzy set, hyperbox-shaped cluster, hyperplane-shaped cluster, min-max hyperbox, neuro-fuzzy inference systems.

1. INTRODUCTION

Fuzzy systems are successfully applied to many different application areas. They are very suitable for complex systems when it is difficult or impossible to describe the system mathematically. A neuro-fuzzy paradigm for the solution of function approximation problems is one of the useful applications of fuzzy systems.

In general, given the set T_Σ consists of the input-output data pairs $(\overline{x}_i, y_i)$, $\overline{x}_i = [x_{i1}\ x_{i2} \ldots x_{in}]$, $i = 1 \ldots P$, which expresses the values of an unknown function f at points $\overline{x}_i$, $(y_i = f(\overline{x}_i))$. System modeling is defined by extrapolation the function f based on these known data. It is well known that there are some existing algorithms to solve this problem based on the fuzzy inference system called T-S model which was proposed by Takagi and Sugeno [7]. The *k*th rule of this fuzzy model is as following:

$R^{(k)}$: If x_{i1} is $B_1^{(k)}$ and … and x_{in} is $B_n^{(k)}$

then

$$y_{ki} = \sum_{j=1}^{n} a_j^{(k)} x_{ij} + a_0^{(k)}, \tag{1}$$

where $\overline{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$ is the *i*th input data vector of T_Σ; $B^{(k)}$ is the input fuzzy set; $[a_0^{(k)}, a_1^{(k)}, \ldots, a_n^{(k)}]^T$ is the weight vector; y_{ki} is output data; $j = 1 \ldots n$, n is the dimension of input set; k=1…M, M is the number of IF-THEN rules.

Based on the T-S model, the input data set is partitioned to build data clusters. Each data cluster is implemented here can be considered as a crisp frame on which different types of MFs (and firing strengths) can be adapted. Some serious drawbacks often affect the clustering algorithms adopted in this context, according to the particular data spaces where they are applied. To overcome such problems, M. Panella et al. [1] proposed a more refined procedure, so that each hyperplane-shaped cluster could be determined in correspondence to the consequent part of the T-S rule. In this approach pure status of the clusters in the input space is established by using the *ARC cutting* procedure of [2]. Although the suggested solution of [1] has the advantage, the *ARC cutting* procedure sometimes manifests shortcoming when it chooses dimension to cut. This respect will be more clearly described in the example presented in Fig. 2 (Sec. 4).

In this paper we present a new cutting procedure

Manuscript received August 17, 2007; accepted June 17, 2008. Recommended by Editor Young-Hoon Joo.

Sy Dzung Nguyen is with the Ho Chi Minh University of Industry, HUI, 12 Nguyen Van Bao, Ho Chi Minh city, Vietnam (e-mail: nsidung@yahoo.com).

Kieu Nhi Ngo is with the Ho Chi Minh University of Technique, HCMUT, Ho Chi Minh city, Vietnam (e-mail: ngokieunhi@yahoo.com).

built based on the pure function ψ and the penalty function τ to direct better the input data space parting process. These functions denote distributive status of the labeled patterns according as the coordinate axis chosen to cut. In other words, they express degree of pure distributive state of the labeled patterns depending on each cutting solution. This new cutting procedure is used for building the three new algorithms named *CSHL*, *HLM1* and *HLM2*, which are used for the synthesis of adaptive neuron-fuzzy inference systems.

The study is organized in the following manner. First, Section 2 briefly introduces the *hyperplane clustering* algorithm of [1], which will be used in this article. Section 3 gives some basic definitions relating to this researching. The new algorithms, *CSHL*, *HLM1* and *HLM2*, are presented in Sections 4, 5, and 6. In Section 7, we report on comprehensive set of experiments. Finally concluding remarks are covered in Section 8.

2. THE HYPERPLANE CLUSTERING ALGORITHM [1]

In this section we briefly redeliver the *Hyperplane Clustering* algorithm of [1] which will be used in this article.

Initialization: Given a value of *M*, the hyperplanes are initialized using the well-known *C-Means* algorithm. It is applied in the input space, where the centroids are randomly initialized. Let $\Gamma^{(k)}$, $k=1...M$, be the clusters found by the *C-Means* algorithm in the input space. The correspondence between such clusters and the hyperplanes is based on the following criterion: If an input pattern $\overline{x}_i, i=1...P$, belongs to the cluster $\Gamma^{(q)}, 1\le q\le M$, then the corresponding input-output pair $(\overline{x}_i, y_i)$ is assigned to the hyperplane A_q.

Step 1: The coefficients of each hyperplane are updated using the pairs assigned to either in the initialization or in the successive step 2. For the *k*th hyperplane, $k=1...M$, a set of linear equations has to be solved

$$y_t=\sum_{j=1}^{n}a_j^{(k)}x_{tj}+a_0^{(k)}, \qquad (2)$$

where index *t* spans all the pairs assigned to the *k*th hyperplane. Suited *least-squares* technique can be used to solve the set of linear equations in (2).

Step 2: Each pair $(\overline{x}_i, y_i)$ of the training set is assigned to the updated hyperplane A_q, with *q* such that:

$$d_i=\left|\frac{y_i-\left(\sum_{j=1}^{n}a_j^{(q)}x_{ij}+a_0^{(q)}\right)}{\sqrt{1+\sum_{j=1}^{n}(a_j^{(q)})^2}}\right| =\min\left|\frac{y_i-\left(\sum_{j=1}^{n}a_j^{(k)}x_{ij}+a_0^{(k)}\right)}{\sqrt{1+\sum_{j=1}^{n}(a_j^{(k)})^2}}\right|,\quad k=1...M.$$

- Stop criterion:

$$\Theta=\left|D-D^{(old)}\right|/D^{(old)},$$

$$D=\frac{1}{P}\sum_{i=1}^{P}d_i$$

where $D^{(old)}$ is the approximation error calculated in the previous iteration.

If $\Theta<\theta$ then the clustering algorithm is stopped, else: go back to Step 1. In this paper, we use the default value $\theta=0.01$.

3. BASIC DEFINITIONS

3.1. Hyperplane-shaped cluster and label of a pattern

Given the initial data pattern set T_Σ. If we use the *Hyperplane Clustering* algorithm of [1] for T_Σ with *M* fuzzy rules, then we have *M* labeled hyperplane-shaped clusters. If the pattern $\overline{x}_i=[x_{i1}x_{i2}...x_{in}]$ belongs to the hyperplane-shaped cluster labeled *k*, then we say that the label of the pattern $\overline{x}_i$ is *k*, or the pattern $\overline{x}_i$ belongs to the class label *k*.

The *Hyperplane Clustering* algorithm creates a hybrid hyperbox *hHB* covering all the labeled patterns belonging to the initial training pattern set T_Σ.

3.2. Min-max hyperbox and min-max vertexes

The min-max clustering technique concerns covering the patterns of the data pattern set using min-max hyperboxes (*HB*).

Let T_t be the set of the patterns covered by the *t*th min-max hyperbox HB_t. The HB_t has two vertexes, the max vertex $\overline{\omega}_t$ and the min vertex $\overline{v}_t$ defined as following:

$$\overline{\omega}_t=[\omega_{t1}\omega_{t2}...\omega_{tn}],\quad \overline{v}_t=[v_{t1}v_{t2}...v_{tn}],$$

where

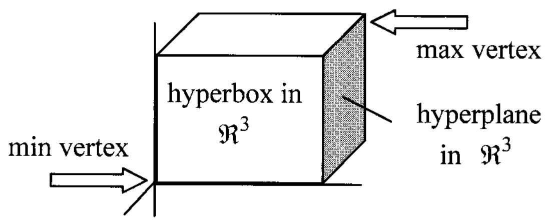


Fig. 1. The min-max hyperbox in $\Re^3$.

$$\omega_{tj} = max(x_{ij} \mid \bar{x}_i \in T_t), \quad j = 1...n$$
$$v_{tj} = min(x_{ij} \mid \bar{x}_i \in T_t), \quad j = 1...n.$$

Each min-max hyperbox (or briefly, hyperbox) can be identified by boundary hyperplanes parallel with the correlative coordinate axes of the data space, such that each hyperplane across one of the two max-min vertexes, either the max vertex or the min vertex (Fig. 1).

3.3. Hyperbox-shaped cluster

All of patterns of the data pattern set T_t covered by a hyperbox can be considered as a data cluster called the hyperbox-shaped cluster (or briefly, the hyperbox cluster).

3.4. Pure hyperbox and hybrid hyperbox

Consider the *t*th hyperbox HB_t covers a set T_t of the labeled patterns:

- If T_t consists of the patterns associated with the class label *m* only, then the HB_t will be said to be the pure hyperbox labeling *m*, and signed $pHB_t^{(m)}$.

- If $T_t \neq \emptyset$ and the HB_t is not the pure hyperbox, i.e., T_t consists of the patterns associated with more than one class label, then the HB_t will be said to be the hybrid hyperbox, and signed hHB_t.

3.5. The overlap condition $^{(*)}$

Given the hyperbox HB_h. Consider an arbitrary hyperbox $HB_k, h \neq k$. We say that the HB_h does not overlap with the HB_k if and only if:

$$\bar{\omega}_h < \bar{v}_k \text{ or } \bar{v}_h > \bar{\omega}_k.$$

Let L_p be the set of all the pure hyperboxes which are created from the initial data pattern set T_Σ. Let L_h be the set of all the hybrid hyperboxes which are created from the initial data pattern set T_Σ, i.e.,

$$T_{\Sigma p} \cup T_{\Sigma h} = T_\Sigma,$$

where

$T_{\Sigma p}$ is the set of all patterns belonging to L_p,

$T_{\Sigma h}$ is the set of all patterns belonging to L_h.

We say that the HB_h satisfies *the overlap condition* if and only if the HB_h does not overlap with any hyper-box *HB* in L_p and in L_h (neither pure nor hybrid).

3.6. Fusion hyperbox $^{(**)}$

Given two pure hyperboxes $pHB_h^{(m)}$ and $pHB_k^{(m)}$, $h \neq k$, labeling *m*. A hyperbox labeling *m*, signed $pHB_f^{(m)}$, will be the fusion hyperbox between the pair of these hyperboxes $pHB_h^{(m)}$ and $pHB_k^{(m)}$ if and only if the three following conditions are concurrently satisfied:

1) $T_f = T_k \cup T_h$
2) $\bar{\omega}_f = \max(\bar{\omega}_k, \bar{\omega}_h)$ and $\bar{v}_f = \min(\bar{v}_k, \bar{v}_h)$
3) $pHB_f^{(m)}$ satisfies *the overlap condition,*

where T_h, T_k, and T_f are the sets of patterns belonging to $pHB_h^{(m)}$, $pHB_k^{(m)}$ and $pHB_f^{(m)}$, respectively.

3.7. The membership function

The membership function MF for each hyperbox fuzzy set describes the degree to which a pattern fits within the hyperbox. In addition, it is typical to have fuzzy membership values range between 0 and 1.

4. PURE FUNCTION, PENALTY FUNCTION AND ALGORITHM CSHL

In this section we present a new algorithm named *CSHL* used to cut hybrid hyperboxes in order to build pure hyperboxes.

4.1. Pure function

Given the hybrid hyperbox *hHB* in $\Re^n$ covering patterns $(\bar{x}_i, y_i)$, $\bar{x}_i = [x_{i1}x_{i2}...x_{in}]$.

Consider the two class labels *nh_1* and *nh_2* which are the most frequent ones with respect to the whole number of the patterns covered by this *hHB*. Let S_1 and n_1 be set of the patterns labeling *nh_1* and the number of these patterns in S_1, respectively. Let S_2 and n_2 ($n_1 \geq n_2$) be set of the patterns labeling *nh_2* and the number of these patterns in S_2, respectively. Let C_1 and C_2 be the centroids of S_1 and S_2. We use the notation d_i to express their distance along the *i*th coordinate axis, i.e.,

$$d_i = |C_{1i} - C_{2i}|, i = 1...n. \tag{3a}$$

Let α_i, $i = 1...n$ be the average value between the *i*th axis coordinates of C_1 and C_2, i.e.,

$$\alpha_i = \left(C_{1i} + C_{2i}\right)/2 \qquad i = 1...n. \tag{3b}$$

Let C_i be the point on *i*th coordinate axis at coordinate α_i. We use the notation $MC_i, i = 1...n$, to represent the hyperplane perpendicular to the *i*th coordinate axis at C_i. Each MC_i cuts the hybrid hyperbox *hHB* in two, signed HB_1 and HB_2. So we have *n* cutting ways to choose.

Let n_1^{1j} and n_2^{1j} be the number of the patterns labeling *nh_1* and *nh_2* covered by the HB_1, respectively; let n_1^{2j} and n_2^{2j} also be the number of the patterns labeling *nh_1* and *nh_2* covered by the HB_2, respectively, when the cutting operation is executed on coordinate axis *j*, *j*=1...*n*.

Let ψ^{1j}, ψ^{2j} be functions defined as following:

$$\psi^{1j} = \left|\frac{n_1^{1j}}{n_1} - \frac{n_2^{1j}}{n_2}\right|, \quad \psi^{2j} = \left|\frac{n_1^{2j}}{n_1} - \frac{n_2^{2j}}{n_2}\right|$$
$$\Rightarrow 0 \le \psi^{1j} = \psi^{2j} \le 1.$$

Let ψ^j be the pure function. It is defined as following

$$\psi^j = \left|\frac{n_1^{1j}}{n_1} - \frac{n_2^{1j}}{n_2}\right|, \tag{4}$$

where due to $\psi^{1j} = \psi^{2j}$ so we can freely choose, either the HB_1 or the HB_2, to establish the pure function (4).

We outline that each value of the pure function ψ^j expresses a distributive status of the patterns labeling *nh_1* and *nh_2* in the HB_1 and the HB_2. For example, consider some values of the function ψ^j as following:

- If $\psi^j = 0$: we can infer that if cutting operation is executed on coordinate axis *j*, then the ratios of the number of the patterns labeling *nh_1* and *nh_2* covered by the HB_1 to the number of the patterns labeling *nh_1* and *nh_2* covered by the initial hybrid hyperbox *hHB*, respectively, are the same , i.e.,

$$\frac{n_1^{1j}}{n_1} = \frac{n_2^{1j}}{n_2} = k_0.$$

In addition, if $k_0 = 50\%$ then we can see that the number of the patterns labeling *nh_1* (or *nh_2*) in the HB_1 and the number of the patterns labeling *nh_1* (or *nh_2*) in the HB_2 are equal.

Clearly, we should avoid these distributive statuses because in these cases, the convergent speed of data space parting process is slow. This respect will be denoted by the penalty function τ in the next section.

- If $\psi^j = 1$: we can infer that if cutting operation is executed on coordinate axis *j*, then the ratio of the number of patterns labeling *nh_1* covered by the HB_1 to the number of patterns labeling *nh_1* covered by the initial hybrid hyperbox *hHB* is 100% (or 0%), and the ratio of the number of patterns labeling *nh_2* covered by the HB_1 to the number of patterns labeling *nh_2* covered by the initial hybrid hyperbox *hHB*, conversely, is 0% (or 100%). It is the same as the distributive status in HB_2. Clearly, we want to get this distributive status to increase the convergent speed of the data space parting process building pure hyperboxes. This respect will also be denoted by the penalty function τ in the next section.

In general, the pure function ψ^j shows the pure degree of distributive statuses of patterns labeling *nh_1* and *nh_2* in the HB_1 and the HB_2 according as the coordinate axis chosen to cut. The higher value of the function ψ^j, the more purebred degree of distribution of patterns labeling *nh_1* and *nh_2* in HB_1 and HB_2, and conversely.

4.2. Penalty function

Penalty function τ_j is defined as following:

$$\tau_j = \begin{cases} 0 & if \quad \psi^j \le \varepsilon_1 \\ (\psi^j + \Delta) & if \quad \psi^j \ge \varepsilon_2 \\ 1 & if \quad \varepsilon_1 < \psi^j < \varepsilon_2, \end{cases} \tag{5a}$$

where

$$[\varepsilon_1, \varepsilon_2, \Delta]^T \tag{5b}$$

is the parameter vector used for direction for input data space parting process, which is rewarded if $\psi^j \ge \varepsilon_2$, or penalized if $\psi^j \le \varepsilon_1$, or not influenced if $\varepsilon_1 < \psi^j < \varepsilon_2$. Here, $j = 1...n$, n is the dimension of input data space pattern set T_Σ. In the following experiments, we will use $\varepsilon_1 = 0.05$; $\varepsilon_2 = 0.95$.

So, if we use the cutting hyperplanes MC_j, *j*=1...*n*, to cut the *hHB* in two, the HB_1 and the HB_2 , we will have *n* different values of the penalty function $\tau_j, j = 1...n$, i.e., have *n* solutions to choose.

4.3. Algorithm *CSHL*

According to the above detail analysis, we can see that the direction for the data space parting process can be based on the pure function ψ^j and the penalty function τ_j. The pure function ψ^j denotes distributive status of the patterns labeling *nh_1* and *nh_2* in the HB_1 and the HB_2 according as the axis chosen to cut. The penalty function τ_j expresses preference to the parting solution having high value of the pure function ψ^j. These are the firm bases to build the new cutting procedure, which will be used for the *CSHL*, because the main aim of this procedure is to establish the pure clusters covering all the labeled patterns from the initial data set T_Σ.

4.3.1 A new cutting procedure - *NCP*

The difference between the new cutting procedure, named *NCP*, proposed in this paper and the *ARC cutting procedure* of [2], which was used for [1], is the cutting criterion. For the *ARC*, the edge of *hHB* to be cut is the one parallel with the coordinate axis *k* such that the distance between C_1 and C_2 along this coordinate axis (3a) is maximum:

$$d_k = \max(d_i), \quad i = 1...n. \tag{6}$$

For the *NCP*, the edge to be cut is the one parallel with the coordinate axis *k* such that the two following priorities are satisfied:

- The first: the value of the pure function ψ^k is large. Quantity of this priority is expressed by the large value of the penalty function τ_k (5).

- The second: the edge to be cut is the one parallel with the coordinate axis *k* such that distance between C_1 and C_2 along this coordinate axis, d_k, is large.

Consequently, the *NCP* cuts on the coordinate axis *k* such that:

$$\tau_k d_k = \max(\tau_i d_i), \quad i = 1...n. \tag{7}$$

The following example more clearly presents influence of distributive status on alternative and decision of the *ARC* and the *NCP*.

Example (Fig. 2) The hybrid hyperbox *hHB* covers three class labels as following:

$$\circ \; n_1 = 20; \; \bullet \; n_2 = 20; \; * \; n_3 = 12.$$

Consider the two class labels *nh_1* ($\circ$) and *nh_2* ($\bullet$) which are the most frequent ones with respect to the whole number of the patterns covered by this *hHB*. Suppose that d_1, d_2 are given like that shown in Fig. 2. Given $\varepsilon_1 = 0.05; \varepsilon_2 = 0.95; \Delta = 0.335$ (5b).

If MC_1 is used then the HB_1 is the left part and the HB_2 is the right part. If MC_2 is used then HB_1 is one below, and HB_2 is the one above.

Fig 2(a): $d_1 = 3 < d_2 = 11$

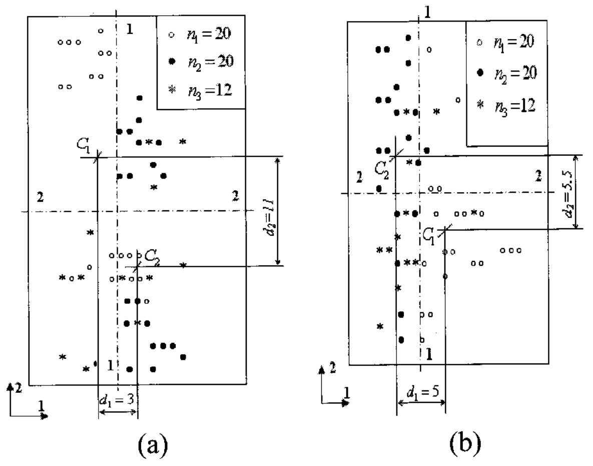


Fig. 2. Cutting solutions: *ARC* [1,2] and the new cutting procedure- *NCP*.

$$\psi^1 = \left|\frac{n_1^{11}}{n_1} - \frac{n_2^{11}}{n_2}\right| = \left|\frac{14}{20} - \frac{0}{20}\right| = 0.7$$

$$\psi^2 = \left|\frac{n_1^{12}}{n_1} - \frac{n_2^{12}}{n_2}\right| = \left|\frac{10}{20} - \frac{10}{20}\right| = 0$$

$$\Rightarrow \tau_1 = 0.7; \tau_2 = 0$$

$$\Rightarrow \tau_1 d_1 = 0.7 \times 3 = 2.1 > \tau_2 d_2 = 0 \times 11 = 0$$

Fig 2(b): $d_1 = 5 < d_2 = 5.5$

$$\psi^1 = \left|\frac{n_1^{11}}{n_1} - \frac{n_2^{11}}{n_2}\right| = \left|\frac{0}{20} - \frac{19}{20}\right| = 0.95$$

$$\psi^2 = \left|\frac{n_1^{12}}{n_1} - \frac{n_2^{12}}{n_2}\right| = \left|\frac{16}{20} - \frac{5}{20}\right| = 0.55$$

$$\Rightarrow \begin{cases} \tau_1 = (\psi^1 + \Delta) = (0.95 + 0.335) = 1.285 \\ \tau_2 = 1 \end{cases}$$

$$\Rightarrow \begin{cases} \tau_1 d_1 = 1.285 \times 5 = 6.425 \\ \tau_2 d_2 = 1 \times 5.5 = 5.5 \end{cases}$$

$$\Rightarrow \tau_1 d_1 > \tau_2 d_2$$

Consequently, based on (6) and (7) we can infer that in both of these cases, the *ARC cutting procedure* of [2] chooses coordinate axis 2 (line 2-2) to cut; conversely, the *NCP* chooses coordinate axis 1 (line 1-1) to cut, in spite of $d_1 < d_2$. Consider distributive status of patterns shown in Fig. 2. We can see that cutting operation in co-ordinate axis 1 is better because it creates the HB_1 and the HB_2 having the pure degree of distribution to be higher.

4.3.2 Algorithm *CSHL*

Let L_p be set of the whole of pure hyperboxes created from the initial data pattern set T_Σ. Let L_h be set of the whole of hybrid hyperboxes created from the initial data pattern set T_Σ. Let *box_number* be the number of hybrid hyperboxes belonging to the L_h.

The *Hyperplane Clustering algorithm* creates the only *hHB* covering all the labeled patterns belonging to the set, T_Σ. So the algorithm *CSHL* begins with *box_number*=1 (at this time, $L_p = \varnothing$; $L_h = \{hHB\}$).

- Delete HB_1 and HB_2

(In this algorithm, delete a hyperbox *HB* means set up the state of $HB = \varnothing$).

Step 1:

- If $box\_number = 0$ then go to Step 4.
- If $box\_number > 0$ then arrange hybrid hyperboxes *hHB* of the L_h in order of the increasing the number of patterns of each *hHB*;

Identify hHB_{box_number}; Go to Step 2.

Step 2: Cut hHB_{box_number} in two, HB_1 and HB_2:

- Find out the coordinate axis k satisfying (7). Use (3b) to calculate α_k.
- Cutting operation on coordinate axis k based on the following principle: consider all the input patterns $\bar{x}_i = [x_{i1}x_{i2}...x_{in}]$ belonging to hHB_{box_number}:
  - If $x_{ik} \le \alpha_k$ then $\bar{x}_i \in HB_1$;
  - Else $x_{ik} > \alpha_k$ then $\bar{x}_i \in HB_2$.

Step 3: Classify HB_1 and HB_2:

- If one of the two HB_1 and HB_2 is pure then:

  Store this pure hyperbox in L_p;
  Store this hybrid hyperbox in L_h;
  Delete the hHB_{box_number}, HB_1, and HB_2;
  Retain *box_number* and return to Step 1.
- If both HB_1 and HB_2 are pure: store both ones in L_p. Delete the hHB_{box_number}, HB_1, and HB_2; $box\_number := box\_number - 1$;
  Return to step 1.
- If both HB_1 and HB_2 are hybrid: store both ones in L_h. Delete the hHB_{box_number}, HB_1, and HB_2; $box\_number := box\_number + 1$;
  Return to Step 1.

Step 4: Test the *overlap condition* (*) to establish *fusion hyperboxes* (**) (Sec. 3). Stop.

5. ALGORITHM HLM1

HLM1 is an algorithm used to build neuro-fuzzy network to approximate an unknown function $y = f(\bar{x})$ from a numerical pattern set T_Σ, in which the algorithm *CSHL* (Sec. 4) is used to generate a set of input pure hyperboxes. The Simpson's method [9] is also used to create the MFs (8). Besides, the structure of the ANFIS network of [1] is also used.

5.1. Structure of the neuron-fuzzy network

Structure of the neuron-fuzzy network used for the algorithm *HLM1* is the same as structure of the ANFIS network of [1] (Fig. 3(a)). However, here net's output $\hat{y}_i$, $i = 1...P$ is calculated by the Centre method.

- To establish the fuzzy sets, we adopt the Simpson's MF [9]:

$$\mu_{pHB_r^{(k)}}(\bar{x}_i) = \frac{1}{n}\sum_{j=1}^{n}[1 - f(x_{ij} - \omega_{rj}, \gamma) - f(v_{rj} - x_{ij}, \gamma)],$$
$$f(x,\gamma) = \begin{cases} 1, & if \quad x\gamma > 1 \\ x\gamma, & if \quad 0 \le x\gamma \le 1 \\ 0, & if \quad x\gamma < 0, \end{cases} \tag{8}$$

where

$r = 1...R_k$; R_k is the number of pure hyperboxes labeling k, and $pHB_r^{(k)}$ denotes the rth pure hyperbox among R_k pure hyperboxes labeling k;

$\bar{\omega}_r = [\omega_{r1}\omega_{r2}...\omega_{rn}]$ and $\bar{v}_r = [v_{r1}v_{r2}...v_{rn}]$ are max and min vertexes of $pHB_r^{(k)}$.

γ is a parameter used to adjust the slope of MFs. In this algorithm, we use the default value $\gamma = 0.5$.

- Several *pHB* can be associated with the same class label k. In this case, the overall input *MF* $\mu_{\bar{B}_i^{(k)}}(\bar{x}_i)$, i.e., the firing strength of the kth rule, $k = 1...M$, can be determined based on the method proposed in [1]:

$$\mu_{\bar{B}_i^{(k)}}(\bar{x}_i) = \max\left\{\mu_{pHB_1^{(k)}}(\bar{x}_i), ..., \mu_{pHB_{R_k}^{(k)}}(\bar{x}_i)\right\} \tag{9}$$
$$k = 1...M, i = 1...P.$$

- The output of the neuro-fuzzy network:

$$\hat{y}_i = \frac{\sum_{k=1}^{M}\mu_{\bar{B}_i^{(k)}}(\bar{x}_i).y_{ki}(\bar{x}_i)}{\sum_{k=1}^{M}\mu_{\bar{B}_i^{(k)}}(\bar{x}_i)}, \quad (i = 1...P) \tag{10}$$

$$y_{ki}(\bar{x}_i) = \sum_{j=1}^{n} a_j^{(k)} x_{ij} + a_0^{(k)}. \tag{11}$$

5.2. Algorithm *HLM1*

Let M_{min} and M_{max} be the min number and max number of fuzzy rules used for training.

$M = M_{\min} - 1$

Step 1: Establish a set of hyperplane-shaped clusters and a labeled pattern set from the initial training pattern set T_{Σ} : $M:=M+1$;

Call the *Hyperplane Clustering* algorithm (Sec. 2).

Step 2: Build set L_p of pure hyperboxes *pHB* covering all the patterns belonging to the initial training pattern set T_{Σ} :

Call the *algorithm CSHL* (Sec. 4).

Step 3: Establish the neuro-fuzzy network:

- Calculate values of MFs based on (8) , (9);
- Calculate $\hat{y}_i$ based on the use of (10), (11);
- Calculate the Mean Squared Error:

$$E = \frac{1}{P}\sum_{i=1}^{P}(y_i - \hat{y}_i)^2 \tag{12}$$

Step 4: Test the Stop Criterion:

- If $M < M_{\max}$, then return to Step 1.
- If $M = M_{\max}$, then go to Step 5.

Step 5: Choose the optimal neuro-fuzzy network based on priorities : $E \le [E]$ and M small. Stop.

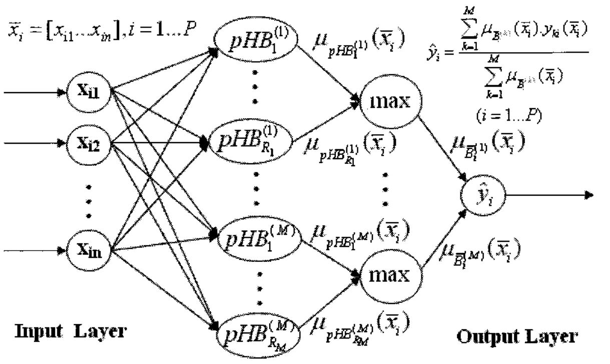
6. ALGORITHM HLM2

Algorithm *HLM2* is used to build neuron-fuzzy networks having the optimal input fuzzy set. In this manner, the *CSHL* is used to establish set L_p of input pure hyperboxes consisting of the whole of the patterns belonging to the initial training pattern set T_{Σ}. Based on L_p and the structure of the net shown in Fig. 3(b), subsequently the neuron-fuzzy network is built and trained to establish the optimal weight set W_{op} used to create the optimal input fuzzy set of network. By this way, consequently, an adaptive neuro-fuzzy inference system with higher degree of the fit between particularity of the initial numerical data set T_{Σ} and the created fuzzy set is established.

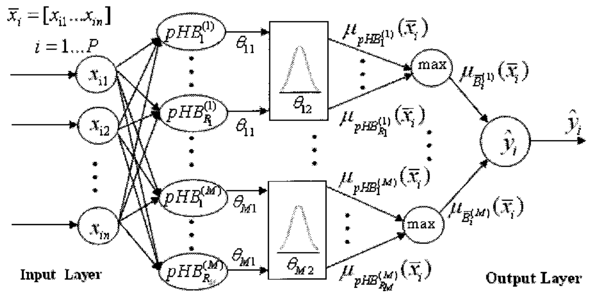
6.1. Structure of the neuron-fuzzy network

Structure of the neuron-fuzzy network used for this algorithm is shown in Fig. 3(b) The input layer and the output layer of this net is the same as the input layer and the output layer of the net used for the algorithm HLM 1 shown in Fig. 3(a).

The difference between the two nets is hidden layer. MFs of the algorithm HLM 2 uses the function Gauss, in which centre location and width of each Gauss's graph depend on the two parameters θ_{i1}, θ_{i2}, $i = 1...M$. So if we use M fuzzy rules, we will have


$\bar{x}_i = [x_{i1}...x_{in}], i = 1...P$
x_{i1}
x_{i2}
x_{in}
$pHB_1^{(1)}$
$pHB_{R_1}^{(1)}$
$pHB_1^{(M)}$
$pHB_{R_M}^{(M)}$
$\mu_{pHB_1^{(1)}}(\bar{x}_i)$
$\mu_{pHB_{R_1}^{(1)}}(\bar{x}_i)$
$\mu_{pHB_1^{(M)}}(\bar{x}_i)$
$\mu_{pHB_{R_M}^{(M)}}(\bar{x}_i)$
max
max
$\hat{y}_i = \frac{\sum_{k=1}^{M}\mu_{B^{(k)}}(\bar{x}_i)y_k(\bar{x}_i)}{\sum_{k=1}^{M}\mu_{B^{(k)}}(\bar{x}_i)}$
$(i = 1...P)$
$\mu_{B^{(1)}}(\bar{x}_i)$
$\mu_{B^{(M)}}(\bar{x}_i)$
$\hat{y}_i$
Input Layer
Output Layer

(a) HLM 1.


$\bar{x}_i = [x_{i1}...x_{in}]$
$i = 1...P$
x_{i1}
x_{i2}
x_{in}
$pHB_1^{(1)}$
$pHB_{R_1}^{(1)}$
$pHB_1^{(M)}$
$pHB_{R_M}^{(M)}$
θ_{11}
θ_{12}
θ_{M1}
θ_{M2}
$\mu_{pHB_1^{(1)}}(\bar{x}_i)$
$\mu_{pHB_{R_1}^{(1)}}(\bar{x}_i)$
$\mu_{pHB_1^{(M)}}(\bar{x}_i)$
$\mu_{pHB_{R_M}^{(M)}}(\bar{x}_i)$
max
max
$\mu_{B^{(1)}}(\bar{x}_i)$
$\mu_{B^{(M)}}(\bar{x}_i)$
$\hat{y}_i$
Input Layer
Output Layer

(b) HLM 2.

Fig. 3.The structure of neuron-fuzzy network.

2M parameters θ_{ij} as a weight set W of net. The optimal weight set, signed W_{op}, is the optimal value set of W such that:

$$E = \frac{1}{P}\sum_{i=1}^{P}(y_i - \hat{y}_i)^2 \to \min. \tag{13}$$

W_{op} can be established by using algorithms for training of neural networks well known. In this paper, we use the algorithm Conjugate Gradient [10].

The MFs are proposed as following:

$$\mu_{pHB_r^{(k)}}(\bar{x}_i) = e^{\frac{-\sum_{j=1}^{n}\left[x_{ij} - \frac{1}{2}\theta_{k1}(\omega_{rj} + v_{rj})\right]^2}{n(\theta_{k2})^2}}, \tag{14}$$

where

$r = 1...R_k$; R_k is the number of pure hyperboxes labeling k, and $pHB_r^{(k)}$ is the rth pure hyperbox among R_k pure hyperboxes labeling k;

$\bar{\omega}_r = [\omega_{r1}\omega_{r2}...\omega_{rn}]$ and $\bar{v}_r = [v_{r1}v_{r2}...v_{rn}]$ are max and min vertexes of $pHB_r^{(k)}$.

- The firing strength of the kth rule, $k = 1...M$, can be determined based on (9).

- The output of the neuro-fuzzy network is calculated by use of (10) and (11).

6.2. Algorithm *HLM2*

Let M_{min} and M_{max} be the min number and max number of fuzzy rules used for training.

$M = M_{\min} - 1$

Step 1: Establish a set of hyperplane-shaped clusters and a labeled pattern set from T_Σ :

$M := M + 1.$

Call the *Hyperplane Clustering* algorithm. (Sec. 2);

Step 2: Build set L_p of pure hyperboxes *pHB* covering all patterns belonging to the initial training pattern set T_Σ :

Call the *algorithm CSHL.* (Sec. 4).

Step 3: Establish the neuron-fuzzy network with an optimal input fuzzy set which is obtained by training the network shown in Fig. 3(b) to create the optimal weight set W_{op}.

- The MFs are calculated by (14) and (9);
- The output of net $\hat{y}_i$ calculated by (10), (11)
- Calculate the Mean Squared Error:

$$E = \frac{1}{P}\sum_{i=1}^{P}(y_i - \hat{y}_i)^2.$$

Step 4: Test the Stop Criterion:

- If $M < M_{\max}$, then return to Step 1.
- If $M = M_{\max}$, then go to Step 5.

Step 5: Choose the optimal neuro-fuzzy network based on priorities : $E \le [E]$ and M small. Stop.

7. NUMERICAL EXPERIMENTS

7.1. Experiment 1

Let us consider the random pattern set named *tr_set1* which consists of 15 random patterns given by MATLAB with three inputs and a single output.

We use the method proposed in [1] and the two new algorithms, *HLM1* (with parameters of the vector (5b) are $\varepsilon_1 = 0.05$; $\varepsilon_2 = 0.95$; $\Delta = 0.35$) and *HLM2* to train the neuron-fuzzy networks to approximate unknown function $y = f(\bar{x})$ from the numerical data set *tr_set1*. The result in the Table 1 shows the effect of the two new algorithms.

7.2. Experiment 2

In this example we consider the following function

Table 1. Error (MSE) of algorithms.

Rules	MSE		
	[1]	***HLM1***	***HLM2***
M=5	0,02760	0,0213	0,0246
M=6	0,02561	0,0107	$7.1148.10^{-4}$
M=7	$2{,}8576.\ 10^{-6}$	$1{,}0199.10^{-7}$	$7.3906.10^{-8}$
M=8	$7{,}0300.\ 10^{-6}$	$1{,}7844.10^{-7}$	$8{,}6461.10^{-9}$
M=9	$5{,}6341.\ 10^{-6}$	$4{,}2997.10^{-7}$	$1{,}1859.\ 10^{-7}$

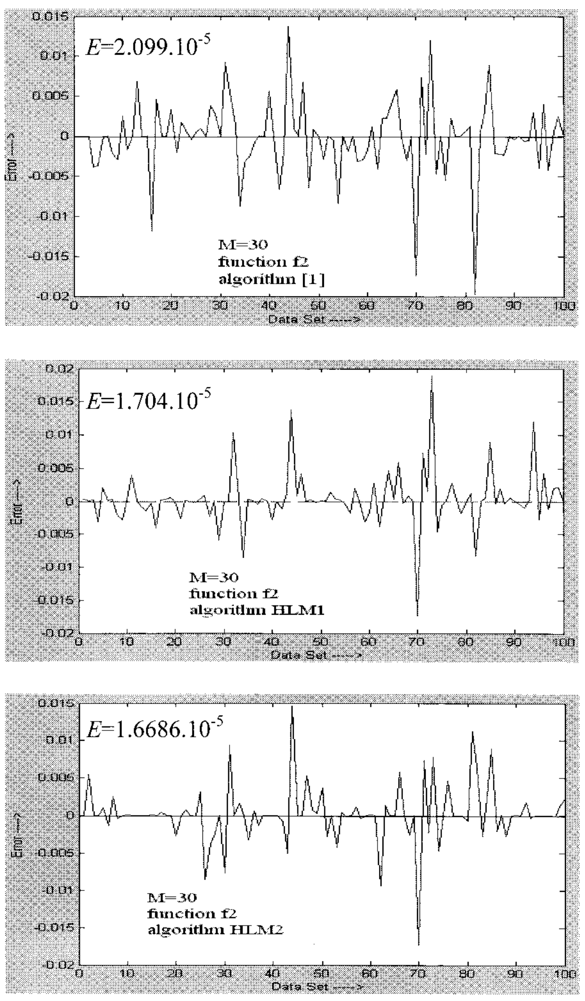


Fig. 4. Error between each output of the data set and of net, $error_i = y_i - \hat{y}_i$, $i = 1...100$. MSE (E) of [1], *HLM1* and *HLM2* with 30 rules.

of [1]:

$$y_2 = (5 - x_2)^2 / [3(5 - x_1)^2 + (5 - x_2)^2],$$

where x_1, x_2 are random values created by MATLAB, $0 \le x_1, x_2 \le 10$. We take 100 input-output patterns as training set named *tr_set2.1* and another 80 input-output patterns as test set named *tr_set2.2*

The *tr_set2.1*

We use *tr_set2.1* as training set.

The parameter vector (5b) of *HLM1* is

$[\varepsilon_1, \varepsilon_2, \Delta]^T = [0.05,\ \ 0.95,\ \ 0.35]^T$.

- Fig. 4 shows $error_i = y_i - \hat{y}_i$, $i = 1...100$ and the mean squared error MSE (E) of [1], *HLM1* and *HLM2* with 30 rules (M=30). It shows a comparison on performance of these training methods.
- Fig. 5 expresses outputs y_i of the training set *tr_set2.1* and outputs $\hat{y}_i$ of the net in two cases. In

the first case (Fig. 5(a)), [1] is used to train the net, and in the second case (Fig. 5(b)), *HLM2* is used to train. Number of rules, M=20. Degree of the difference between y_i and $\hat{y}_i$ shown in Fig. 5a is higher than it shown in Fig. 5b. It means that accuracy degree of *HLM2* is higher than [1].

- In this experiment, we also update the number of input fuzzy clusters created by using each algorithm: [1], *HLM1* or *HLM2* to compare each other. The results show that complex degree of systems built by *HLM1* and *HLM2* is lower than complex degree of system built by [1].

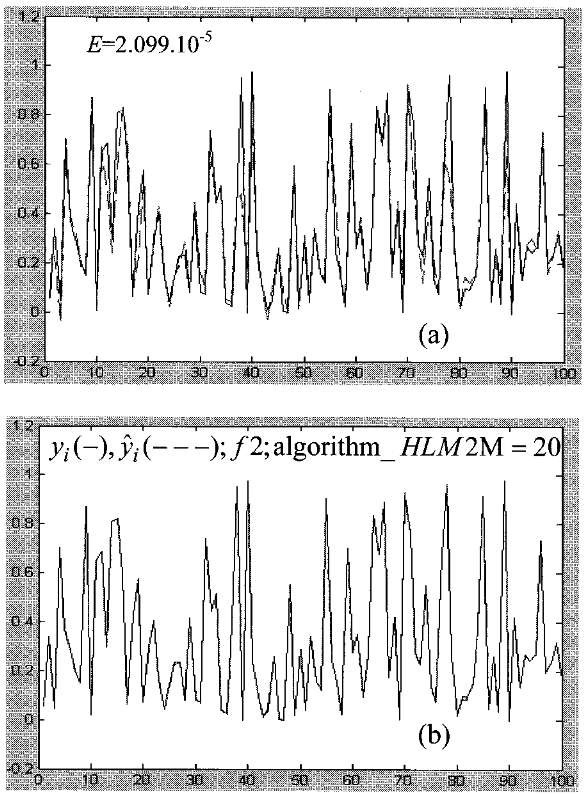


Fig. 5. Degree of the difference between y_i and $\hat{y}_i$ of [1] (Fig. 5a) and *HLM2* (Fig. 5b) with M=20.

Table 2. A comparison on errors (MSE).

The number of rules	Algorithms		
	[1]	***HLM1***	***HLM2***
The training set, *tr_set2.1*			
M=10	2. 10^{-3}	1.7. 10^{-3}	2.7693.10^{-4}
M=20	25.10^{-4}	1.4771.10^{-4}	1.2326.10^{-4}
M=30	2.099.10^{-5}	1.704.10^{-5}	1.6686.10^{-5}
The test set, *tr_set2.2*			
M=10	7.15. 10^{-3}	6.341. 10^{-3}	1.437.10^{-3}
M=20	9.505.10^{-3}	1.112.10^{-3}	8.878.10^{-4}
M=30	4.871.10^{-4}	3.412.10^{-4}	2.866.10^{-4}

The *tr_set2.1* and the *tr_set2.2*

- In this experiment we use *tr_set2.1* as a training set and *tr_set2.2* as a test set. The parameter vector of *HLM1* is used is

$$[\varepsilon_1, \varepsilon_2, \Delta]^T = [0.05, \quad 0.95, \quad 0.35]^T.$$

A comparison on Errors (MSE) among [1], *HLM1* and *HLM2* with different number of rules is shown in Table 2. In the two cases, we can see that if the number of fuzzy rules M are equal, accuracy degree of *HLM1* and of *HLM2* are higher than accuracy degree of [1].

7.3. Experiment 3: *Daily Stock price*

In this experiment we use the daily data of a stock market from [3] ("*Daily Data of Stock A*") as training pattern set. This data consists of ten input and one output $([x_1, x_1, \ldots, x_{10}], y)$.

The parameter vector of *HLM1* is used is

$$[\varepsilon_1, \varepsilon_2, \Delta]^T = [0.05, \quad 0.95, \quad 0.5]^T.$$

Here, we use three methods: [1], *HLM1* and *HLM2* to train the neuro-fuzzy networks to approximate this pattern set. The error MSE (E) of three methods are shown in Table 3. Fig. 6 shows $error_i = y_i - \hat{y}_i$, $i = 1 \ldots 100$ and the mean squared error MSE (E) of [1], *HLM1* and *HLM2* with 12 rules. It shows a comparison on performance of these training methods.

We also update the number of input fuzzy clusters created by each algorithm: [1], *HLM1* or *HLM2* to compare each other. The results show that complex degree of systems built by the new algorithms is lower than complex degree of system built by [1].

These results shows the effect of the two new algorithms, *HLM1* and *HLM2*.

7.4. Experiment 4:

In this experiment we use function

$$y = (1 + x_1^{-2} + x_1^{-1,5})^2, \quad x_1, x_2 \in [1,5]$$

of [4] to establish a training set of 50 patterns which is the same as the training set used in [4]. The parameter vector (5b) of *HLM1* is chosen for this example is:

$$[\varepsilon_1, \varepsilon_2, \Delta]^T = [0.05, \quad 0.95, \quad 0.5]^T;$$

We use algorithms proposed in [1,4-6] and the two new algorithms to approximate this training data set.

Table 3. MSE (E) of the three methods.

Rules	Algorithms		
	[1]	***HLM1***	***HLM2***
M=10	13,4148	0.5618	0,0675
M=12	1,74460	0,2070	0,0352
M=14	3,5028. 10^{-5}	2,1013. 10^{-6}	1,2101.10^{-6}

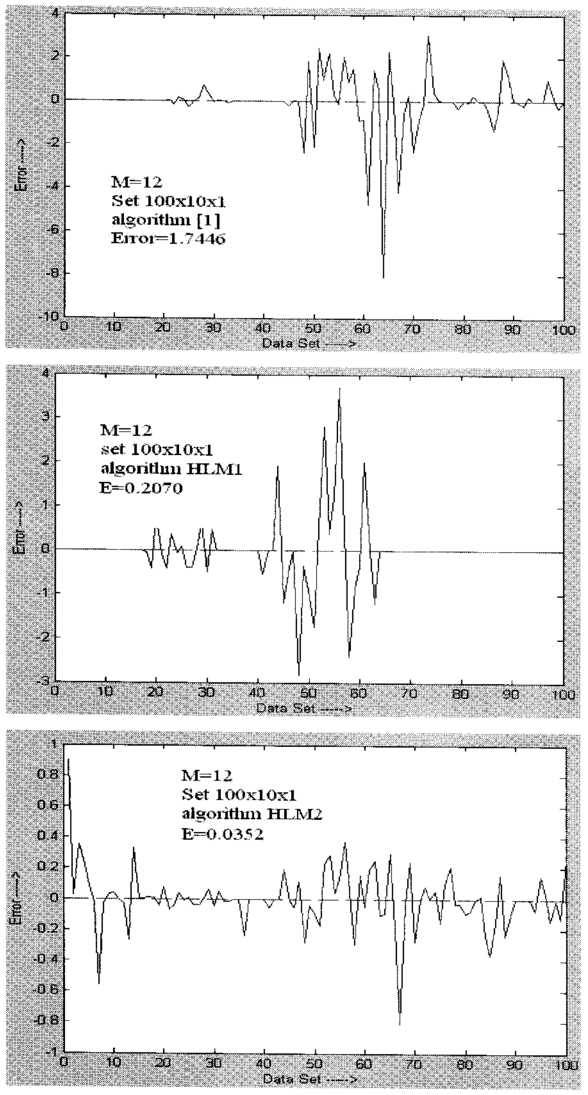


Fig. 6. Error between each output of the data set and of net, $error_i = y_i - \hat{y}_i$, $i = 1...100$. MSE (E) of [1], *HLM1* and *HLM2* with 12 rules.

Table 4. Comparison on errors MSE of algorithms with different number of rules.

Rules	Algorithms					
	[4]	**[5]**	**[6]**	**[1]**	***HLM1***	***HLM2***
M=6	0.0589	0.0599	0.0572	0.0221	0.0182	$1.9587.10^{-4}$
M=8	0.0500	0.0499	0.0499	0.0220	0.0218	$1.8544.10^{-4}$
M=10	0.0148	0.0149	0.0149	0.0188	0,0026	$1.9780.10^{-4}$

Comparison on errors MSE among these algorithms with different number of rules shown in Table 4 denotes the effect of the *HLM1* and the *HLM2*.

8. CONCLUSION

In this paper we describe a neuro-fuzzy system modeling approach to approximate an unknown function $y = f(\bar{x})$ from a numerical data set.

First of all, we present a new input data space parting procedure, named *NCP*, which is based on a pure function ψ and a penalty function τ to direct input data space parting process. The *NCP* is used to build a new algorithm *CSHL*, which is used to create input data clusters as input fuzzy sets. Two new algorithms, *HLM1* and *HLM2*, are proposed. They uses the input fuzzy sets created by *CSHL* to synthesize neuro-fuzzy networks, in which, *HLM1* directly synthesizes neuro-fuzzy network. The other, *HLM2*, uses the optimal weight set W_{op} given by net training process to build a neuro-fuzzy model. By this way, consequently, an adaptive neuro-fuzzy inference system with higher degree of the fit between particularity of the initial numerical data set T_{Σ} and the created fuzzy sets is built. We can see that, the use of the pure function ψ and the penalty function τ increases degree of fit between characteristics of the initial training data set given and the input fuzzy sets established, reduces the number of pure hyperboxes created, i.e., reduces the number of input fuzzy sets. These remarks are inferred by theory and are verified by results of a series of numerical experiments where we use [1], some algorithms well known and the two new algorithms to build neuro-fuzzy models, and then, compare results of them. These experiments show that accurate degree of *HLM1* and *HLM2* is higher than accurate degree of these algorithms. Besides, complex degree of system built by *HLM1* and *HLM2* is lower than complex degree of system built by [1].

REFERENCES

[1] M. Panella and A. S. Gallo, "An input–output clustering approach to the synthesis of ANFIS networks," *IEEE Trans. on fuzzy systems*, vol. 13, no. 1, February 2005.

[2] A. Rizzi, M. Panella, and F. M. F. Mascioli, "Adaptive resolution min-max classifiers," *IEEE Trans. on Neural Networks*, vol. 13, no. 2, pp. 402-414, March 2002.

[3] M. Sugeno and T. Yasukawa, "A fuzzy logic based approach to qualitative modeling," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb. 1993.

[4]] S.-J. Lee and C.-S. Ouyang, "A neuro-fuzzy system modeling with self-constructing rule generation and hybrid SVD-based learning," *IEEE Trans. on Fuzzy Systems*, vol. 11, no. 3, pp. 341-353, June 2003.

[5] Y. Lin, G. A. Cungningham III, and S. V. Coggeshall, "Using fuzzy partitions to create fuzzy system from input-output data and set the initial weights in fuzzy neural network," *IEEE Trans. on Fuzzy systems*, vol. 5, pp. 614-621, Aug. 1997.

[6] C. C. Wong and C. C. Chen, "A hybrid clustering and gradient decent approach for fuzzy modeling," *IEEE Trans. Syst. Man, Cybern. B*, vol. 29, pp. 686-693, Dec. 1999.

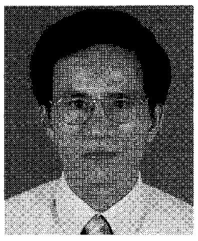
[7] T. Takagi and M. Sugeno, "Fuzzy identification of systems and applications to modeling and control," *IEEE Trans. Syst. Man, Cybern.*, vol. 15, no. 1, pp. 116-132, Jan. 1985.

[8] P. K. Simpson, "Fuzzy min-max neural networks – Part 1: Classification," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 776-786. 1992.

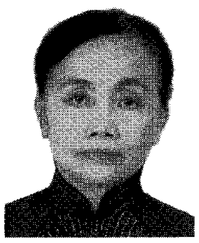
[9] P. K. Simpson, "Fuzzy min-max neural networks – Part 2: Clustering," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 32-45, 1993.

[10] S. D. Nguyen and H. Q. Le, "Adaptive algorithm for training of neural networks based on method conjugate gradient," *Journal of Science & Technology*, no. 58, pp. 68-73, 2006.

[11] J. Mao, J. Zhang, Y. Yue, and H. Ding, "Adaptive-tree-structure-based fuzzy infer-ence system," *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 1, February 2005.



Sy Dzung Nguyen received the M.Eng. degree in Mechanical Engineering from the Ho Chi Minh University of Technique, Vietnam in 2001, where he is currently working toward a Ph.D. degree. From 2001 to 2005, he was a Dean of the Mechanical Faculty of the Vung Tau Community College. He is now a Lecturer with the Mechanical Faculty, Ho Chi Minh University of Industry, HUI, Vietnam. His research interests are in the areas of intelligent control, robotics, and applied mechanics.



Kieu Nhi Ngo received the B.Eng. degree and M.Eng. degree in Dynamics and Machine Durability from the Polytechnic University of Kharkov City, Ucraine, where she received the Ph.D. degree in Mechanics. Currently, she is a Professor and Head of the Applied Mechanical Lab., the Ho Chi Minh University of Technique, Vietnam. Her research interests are in the areas of structural and machine health monitoring system, structural and machine dynamics, computational inverse techniques, and applied mechanics. She has received a number of awards, including the Excellent Teacher Awards in 1998 and in 2006, the International Award named Kovalevska in 2002, and the National Technology-Science Award of Vietnam in 2005.