Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 491237, 12 pages
doi:10.1155/2012/491237

Research Article

A Hierarchical Procedure for the Synthesis of ANFIS Networks

Massimo Panella

Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, via Eudossiana 18, 00184 Rome, Italy

Correspondence should be addressed to Massimo Panella, massimo.panella@uniroma1.it

Received 25 May 2012; Accepted 28 August 2012

Academic Editor: Katsuhiro Honda

Adaptive neurofuzzy inference systems (ANFIS) represent an efficient technique for the solution of function approximation problems. When numerical samples are available in this regard, the synthesis of ANFIS networks can be carried out exploiting clustering algorithms. Starting from a hyperplane clustering synthesis in the joint input-output space, a computationally efficient optimization of ANFIS networks is proposed in this paper. It is based on a hierarchical constructive procedure, by which the number of rules is progressively increased and the optimal one is automatically determined on the basis of learning theory in order to maximize the generalization capability of the resulting ANFIS network. Extensive computer simulations prove the validity of the proposed algorithm and show a favorable comparison with other well-established techniques.

1. Introduction

In this paper, a new procedure for the structural optimization of adaptive neurofuzzy inference systems (ANFIS) is proposed, whose synthesis is based on the hyperplane clustering in the joint input-output space of the data set. ANFIS networks are being widely used to many applicative tasks, such as rule-based process controls, pattern recognition, and function approximation. They solve a general regression problem by means of a set of M rules of Sugeno first-order type [1]. The kth rule, $k = 1 \dots M$, has the following form:

$$\begin{aligned} &\text{If } x_1 \text{ is } B_1^{(k)} \text{ and } \dots \text{ and } x_n \text{ is } B_n^{(k)} \text{ then} \\ &y^{(k)} = \sum_{j=1}^{n} a_j^{(k)} x_j + a_0^{(k)}, \end{aligned} \tag{1}$$

where $\underline{x} = [x_1 \;\; x_2 \;\; \dots \;\; x_n]^T$ is a column vector (or pattern) in the n-dimensional input space and $y^{(k)}$ is the scalar output associated with the rule. The latter is characterized by the MFs $\mu_{B_j^{(k)}}(x_j)$ of the fuzzy input variables $B_j^{(k)}$, $j = 1 \dots n$, and by the coefficients $a_j^{(k)}$, $j = 0 \dots n$, of the crisp output.

Several alternatives are possible for the fuzzification of crisp inputs, the composition of input MFs, and the way rule outputs are combined [2]. Usually, the structure of the fuzzy inference system is the following one:

$$\tilde{y} = \frac{\sum_{k=1}^{M} \mu_{\underline{B}^{(k)}}(\underline{x}) y^{(k)}}{\sum_{k=1}^{M} \mu_{\underline{B}^{(k)}}(\underline{x})}, \tag{2}$$

where $\underline{B}^{(k)}$ is the overall fuzzy input variable, $\mu_{\underline{B}^{(k)}}(\underline{x})$ is the corresponding MF, and $\tilde{y}$ the output estimated for an input $\underline{x}$.

The numerical parameters of rules are obtained through a learning process, by using a training set of P input-output pairs $(\underline{x}_i, y_i)$, $i = 1 \dots P$. The crucial problem during the ANFIS learning is to obtain a good generalization capability. This issue has been deeply investigated in the literature; usually, the generalization capability of ANFIS networks is optimized by checking data set for overfitting model validation [3].

The generalization capability is maximized only if the ANFIS network consists of a suitable number of rules. However, the determination of the optimal number is a very critical problem to be solved, since the neural network might be easily overfitted in the case of noisy or ill-conditioned data. In this paper, a new hierarchical constructive procedure for the automatic determination of the ANFIS rules is proposed. It aims to a regularization of the network architecture based on

learning theory and hyperplane clustering-based techniques [4–7]. The underlying idea of such techniques received in the literature many acknowledgments as, for example, in the case of theoretical models [8–14] as well as applications to specific fields [15–26].

The paper is organized as follows. The synthesis technique for determining the parameters of ANFIS rules is described in Section 2. Successively, the structural optimization for the automatic determination of the rules number, which is based on the proposed hierarchical constructive procedure, is illustrated in detail in Section 3. Finally, the numerical results obtained by extensive computer simulations on synthetic benchmarks and a real-world application are summarized in Section 4.

2. Hyperplane Clustering for the Synthesis of ANFIS Rules

When dealing with numerical data, the rules of ANFIS networks are commonly synthesized by using clustering techniques, which reduce the redundancy of data so that significant rules are determined directly from the clusters modeling the training set. Clustering algorithms can use joint input-output data, input data only, or output data only; this choice heavily affects the way ANFIS rules are built [27–34].

When modeling ANFIS networks, the main drawback due to conventional clustering approaches is that induced clusters do not always reflect the real data structure. An innovative approach, dubbed hyperplane clustering synthesis (HCS), was firstly proposed in [4]. By HCS, hyperplane-shaped clusters are determined in correspondence to the consequent part of each Sugeno rule. Namely, the ANFIS architecture can be considered as a smoothed piecewise linear model, where (2) can be rewritten as

$$\tilde{y} = f(\underline{x}), \quad f : \Re^n \longrightarrow \Re. \tag{3}$$

The regression model is identified by the set of M hyperplanes, each associated with a cluster. Therefore, the prototype of the kth cluster in the joint input-output space will be represented by the set of coefficients $a_j^{(k)}$, $j = 0 \dots n$, evaluated by means of standard least-squares techniques.

It is worth summarizing in the following the core steps for clustering in the "hyperplane space," which is fundamentally an alternating optimization technique aiming to identify the cluster prototypes. Let $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_M\}$ be a set of M clusters (i.e., hyperplanes) and let every pattern of the training set be assigned to one of these clusters; this is obtained according to a suitable criterion, as well established in the following of the paper. Then, the hyperplane clustering with M prototypes is based on the following iterative steps.

Step 1. The coefficients of each hyperplane are evaluated by using the following procedure. For the kth hyperplane Γ_k, $k = 1 \dots M$, a set of linear equations has to be solved; the generic equation is

$$y_t = \sum_{j=1}^{n} a_j^{(k)} x_{tj} + a_0^{(k)}, \tag{4}$$

where index "t" spans only the pairs of the training set assigned to the kth cluster. Suited least-squares techniques can be used to solve the set of linear equations in (4).

Step 2. The assignment of patterns to clusters is updated. Each pair $(\underline{x}_i, y_i)$, $i = 1 \dots P$, of the training set is now assigned to the cluster Γ_q, with q such that

$$\begin{aligned} d_i &= \left| \frac{y_i - \left(\sum_{j=1}^{n} a_j^{(q)} x_{ij} + a_0^{(q)} \right)}{\sqrt{1 + \sum_{j=1}^{n} \left(a_j^{(q)} \right)^2}} \right| \\ &= \min_{k=1 \dots M} \left| \frac{y_i - \left(\sum_{j=1}^{n} a_j^{(k)} x_{ij} + a_0^{(k)} \right)}{\sqrt{1 + \sum_{j=1}^{n} \left(a_j^{(k)} \right)^2}} \right|. \end{aligned} \tag{5}$$

The previous choice states that a pattern is assigned the hyperplane having the minimum orthogonal distance from it. This assures the best results since other choices, as for example, the distance along the output axis of y_i only, produces ambiguous results especially in the case of ill-conditioned data yielding quite vertical hyperplanes.

Step 3. For every cluster Γ_k, the *local* approximation error is evaluated:

$$D_k = \frac{1}{P_k} \sum_{t} d_t, \tag{6}$$

where index "t" spans only the P_k pairs of the training set assigned lately (i.e., in Step 2) to the kth cluster.

Step 4. The convergence is based on the quantity:

$$\Theta = \frac{\left| D - D^{(\text{old})} \right|}{D^{(\text{old})}}, \tag{7}$$

where D is the *global* approximation error over the whole data set in the current iteration, which is defined by

$$D = \frac{1}{P} \sum_{i=1}^{P} d_i, \tag{8}$$

and $D^{(\text{old})}$ is the global approximation error calculated in the previous iteration. If Θ is less than a predetermined threshold θ, then the clustering algorithm is stopped. Otherwise, the iteration goes back to Step 1 by using the current updated association of patterns to clusters. The default value $\theta = 0.01$ will be used in the following.

The clustering algorithm described so far yields the linear consequents of Sugeno rules only. In order to achieve the complete structure of the ANFIS network, it is mandatory to determine the firing strengths corresponding to the antecedents of rules. To this end, once the iterations of hyperplane clustering have stopped, each pattern of the training set can be labeled with an integer q, $1 \le q \le M$, representing the hyperplane it has been assigned during Step 2 of the last iteration. By using the labeled training set, a classification

problem can be solved in the input space only; at the end of the training process, the input space will be tiled by a classification model able to assign a fuzzy label $L(\underline{x})$ to any pattern $\underline{x}$ of the input space

$$L(\underline{x}) = \left[\mu_{\underline{B}^{(1)}}(\underline{x}) \quad \mu_{\underline{B}^{(2)}}(\underline{x}) \ \ldots \ \mu_{\underline{B}^{(M)}}(\underline{x})\right], \tag{9}$$

where the kth element of $L(\underline{x})$ represents the fuzzy membership of the pattern to the kth class and hence, it can be assumed as the firing strength $\mu_{\underline{B}^{(k)}}(\underline{x})$ of the kth rule associated with the hyperplane corresponding to that class.

As a reference classification model, we will adopt in the following well-known Simpson's Min-Max model [35], since it has demonstrated the best performance in this regard. The combination of the hyperplane clustering followed by the fuzzy classification in the input space defines the HCS procedure allowing the determination of an ANFIS network for a given number of rules. HCS will be the basic algorithm adopted for the hierarchical optimization successively proposed.

It is worth mentioning that the hyperplane clustering inside HCS is a particular instance of the well-known fuzzy c-regression models (FCRM) algorithm [27]; in this case, a set of $c = M$ linear models (hyperplanes) is fit on the training data and each pattern is assigned during clustering in a crisp manner (i.e, using hard labels) to a unique hyperplane, the one at the minimum distance (5). Although FCRM can use nonlinear models as well, they are basically adopted for regression in order to fit the training data only, even if they are not representing a function (i.e, when several output values y correspond to a same input $\underline{x}$). Conversely, the hyperplane clustering of HCS is successively completed by the classification in the input space, under the assumption that the resulting ANFIS model can be used to approximate/estimate an unknown function represented by its samples in the training set.

In this regard, when the output $\tilde{y}$ must be estimated for any input $\underline{x}$ during the normal ANFIS operation (i.e., testing), the classifier is used to determine the fuzzy label (9) using only the input value $\underline{x}$; then, the output $\tilde{y}$ is calculated using (2) by means of the firing strengths contained in the fuzzy label $L(\underline{x})$ and the linear consequents in (1), early determined by the hyperplane clustering. In this phase, it is not necessary, in general, to assign the test sample $\underline{x}$ a specific hyperplane cluster. However, a "hard approximation" using a single ANFIS rule can be obtained by a defuzzification of the fuzzy label $L(\underline{x})$, that is,

$$\begin{aligned} \tilde{y}^{(\text{hard})} &= \sum_{j=1}^{n} a_j^{(q)} x_j + a_0^{(q)} \\ q &= \max_{k=1\ldots M} \left\{\mu_{\underline{B}^{(k)}}(\underline{x})\right\}. \end{aligned} \tag{10}$$

3. Hierarchical Optimization of the ANFIS Structure

When training an ANFIS network, the main problems are the local convergence of estimation algorithms and the correct determination of the number M of rules. As evidenced successively, the former problem mainly depends on a good (usually random) initialization of numerical parameters associated with each rule. The latter one is a well-known problem, which is directly related to the generalization capability of the neurofuzzy network and it can also be referred to as "structural optimization" problem. In fact, the ANFIS performance could be inadequate if the training set is either underfitted or overfitted by a lacking or an excessive number of rules, respectively.

3.1. The Plain Optimization Approach. A plain solution to these problems could be based on the use of the HCS algorithm with different values of M and with different initializations for every value of M. The HCS algorithm should be initialized before Step 1 of its first iteration by setting an association of each pattern with one of the M hyperplane clusters. The simplest way is to associate patterns to clusters randomly. In this paper, a different approach will be followed, by clustering data in the input space only, in order to obtain a preliminary partition of the training set in clusters. Consequently, the random initialization is necessary for the chosen clustering algorithm in the input space as, for instance, the fuzzy c-means [36, 37]. Once the set of different ANFIS networks is generated, the best network can be chosen by relying on the supervised nature of the learning procedure, that is, by using a cost function measuring the overall generalization capability of the network in terms of its complexity and its approximation error. Basic concepts of learning theory can be adopted in this regard [2, 3]. Namely, the ANFIS network achieving the best generalization capability is the one that, under the same performance on the training set, is characterized by the lowest number of rules.

As a measure of the network performance on the training set, the mean squared error (MSE) is adopted:

$$E = \frac{1}{P}\sum_{i=1}^{P}(y_i - \tilde{y}_i)^2, \tag{11}$$

where $\tilde{y}_i$ is the output generated by the ANFIS network in correspondence to the ith input pattern of the training set. The optimal network is selected by using the following cost function depending upon the number of ANFIS rules:

$$F(M) = (1-\lambda)\frac{E(M) - E_{\min}}{E_{\max} - E_{\min}} + \lambda\frac{M}{P}, \tag{12}$$

where $E_{\min}$ and $E_{\max}$ are the extreme values of the performance E that are encountered during the performance investigation of the different ANFIS networks; λ is a weight $0 \le \lambda \le 1$. This weight is not critical, since the results are slightly affected by its variation in a large interval centered in 0.5. Evidently, for a given value $M = \overline{M}$, the ANFIS network showing the best value $F(\overline{M})$ in (12) will be the one whose initialization and the successive HCS iterations yield the best performance $E(\overline{M})$ on the training set.

The plain minimization of (12) can be obtained by the constructive technique denoted as optimized HCS (OHCS). In this procedure, the number of rules is progressively

increased from 1 to $M_{\max}$, where $M_{\max}$ is a fraction of the training set cardinality P and it represents the maximum complexity that is allowed to the network. For each value of M, different initializations are considered and several ANFIS networks are generated through HCS. In fact, as mentioned, the latter is based on a random initialization of the hyperplanes and hence, different initializations might produce different networks for the same value of M. This is mainly due to the hyperplane clustering, which can easily get trapped at local minima of (8), as for any other alternating optimization procedure.

If different T initializations are carried out for each value of M, the OHCS procedure will generate $TM_{\max}$ networks and the optimal one will be selected according to (12). This optimization approach still suffers from serious drawbacks, which basically depend on the number of different initializations performed for each value of M. The lower is the number of initializations, the lower is the probability to obtain a satisfactory ANFIS network after the HCS algorithm. Conversely, the higher is the number of initializations, the higher is the computational cost of the optimization procedure.

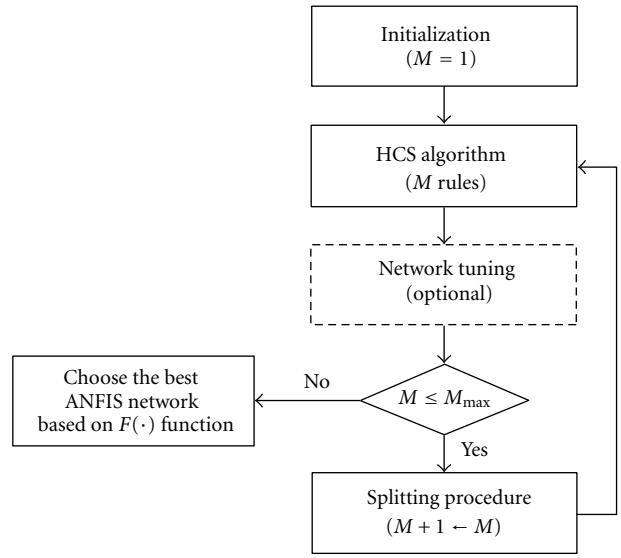

Figure 1: Flow chart of the HHCS algorithm. The tuning of ANFIS networks is an optional step, as discussed in Section 3.5.

3.2. The Hierarchical HCS Procedure. In order to overcome the previous problems, in this paper, the use of a splitting hierarchical optimization is proposed, which will be referred to in the following as hierarchical HCS (HHCS). It is based on the constructive procedure shown in Figure 1, where M is increased progressively and only one run of the HCS algorithm is needed for every value of M. In fact, HHCS starts with only one hyperplane (i.e., $M = 1$), which is initialized without any ambiguity since every pattern of the training set is assigned to the sole cluster. Then an iteration starts, involving the succession of the following procedures.

(i) Considering the current association of patterns to clusters, the HCS algorithm is executed on the ANFIS network with M hyperplanes.

(ii) An optional fine tuning of the final ANFIS parameters obtained by the HCS algorithm can be performed by using suited techniques, as outlined in Section 3.5.

(iii) If the maximum number of rules $M_{\max}$ is reached, then the iteration is stopped and the best ANFIS network is chosen among those obtained during the iterations, once again according to (12). Conversely, if $M < M_{\max}$, a *splitting procedure* is carried out.

(iv) By means of the splitting procedure, which is illustrated successively in Section 3.3, the cluster associated with the hyperplane showing "in some sense" the worst local approximation of data is selected to be split into two new hyperplanes, with the aim to better approximate the patterns belonging to the split cluster (as determined by the last execution of Step 2 within the HCS algorithm.) After the split is performed, the old cluster is removed from the ANFIS network and the new ones are inserted, increasing evidently by 1 the value of M. So, the iteration starts back with the HCS algorithm applied to an ANFIS network composed by $M+1$ rules, where each pattern of the training set is currently assigned to exactly one cluster.

It is important to outline that patterns belonging to the split cluster are assigned exclusively to one of the new generated clusters, while the association of patterns to the other clusters is kept unaltered. This evidences the hierarchical nature of the proposed HHCS procedure. Furthermore, an unambiguous initialization of HCS is assured for any value of M and hence, the necessity of random initializations and to optimize different HCS solutions for a given value of M is eliminated. Consequently, the computational cost of HHCS is heavily reduced with respect to the plain OHCS approach.

The algorithm will stop when M reaches the maximum complexity $M_{\max}$. This is not a critical value to be set; in fact, the same solutions obtained when M spans up to a given value of $M_{\max}$ are still found when a larger $M_{\max}$ is adopted. So, the quality of the optimal solution can be traded off with the computational cost of the whole optimization routine given that the larger $M_{\max}$, the more solutions are explored spending more time. By the way, we have determined as a value generally acceptable for $M_{\max}$ the 40% of the training set cardinality P (i.e., $M_{\max} = 0.4P$).

3.3. Splitting Procedure of Clusters. The key role in the HHCS algorithm is played by the splitting procedure, which is intended to determine optimally the cluster (or hyperplane) to be split and, successively, to initialize the hyperplanes associated with its offspring. The underlying idea is to prevent the initialization of new hyperplanes in underpopulated zones of the training set, since this is the typical situation where the HCS algorithm will converge to a poor local minimum of (8). Several heuristics are possible in this regard; the one illustrated in the following steps is based on the supervised nature of function approximation problems

and on principal component analysis (PCA). In fact, the cluster is split orthogonally with respect to the direction of maximum variance of local data.

(1) Taking into account the local approximation errors (6), which are determined in the last execution of Step 3 within the HCS algorithm, the cluster to be split, denoted as Γ_s, $1 \le s \le M$, is the one exhibiting the largest local approximation error

$$D_s = \max_{k=1\ldots M} D_k. \tag{13}$$

(2) Let $I(s)$ be the set of indexes of the P_s patterns associated to Γ_s during the last execution of Step 2 within the HCS algorithm. The mean vector $\underline{\mu}^{(s)}$ of such patterns is assumed in the following to be

$$\underline{\mu}^{(s)} = \frac{1}{P_s} \sum_{t \in I(s)} \underline{x}_t, \tag{14}$$

and the covariance matrix $\mathbf{C}^{(s)}$ to be

$$\mathbf{C}^{(s)} = \frac{1}{P_s} \sum_{t \in I(s)} \left(\underline{x}_t - \underline{\mu}^{(s)}\right)\left(\underline{x}_t - \underline{\mu}^{(s)}\right)^T. \tag{15}$$

Using these notations, the principal components (or directions) $\underline{q}_j^{(s)}$, $j = 1 \ldots n$, of Γ_s are computed. More precisely, the column vector $\underline{q}_j^{(s)}$ is the jth eigenvector of $\mathbf{C}^{(s)}$ and the eigenvectors are sorted in descending order, according to the value of the corresponding eigenvalue.

(3) Let $\underline{q}_1^{(s)}$ be the principal component corresponding to the highest eigenvalue. Then, the projections of patterns of Γ_s along the principal direction is calculated

$$\xi_{t1}^{(s)} = \left(\underline{x}_t - \underline{\mu}^{(s)}\right)^T \underline{q}_1^{(s)}, \quad t \in I(s). \tag{16}$$

Finally, the cluster Γ_s is removed from the ANFIS network together with the corresponding rule; it will be replaced by the new generated clusters $\Gamma_s^{(+)}$ and $\Gamma_s^{(-)}$. Taking into account the superscript notation of these clusters, each pattern of Γ_s will be assigned to either $\Gamma_s^{(+)}$ or $\Gamma_s^{(-)}$, according to the sign of the projection $\xi_{t1}^{(s)}$ in (16).

3.4. A Toy Example to Clarify the HHCS Procedure. The details of the proposed hierarchical optimization can be further clarified with the following example concerning the approximation of a simple piecewise linear function consisting of 4 different linear models:

$$f(x) = \begin{cases} 0, & -15 \le x < 0 \\ 2x, & 0 \le x < 5 \\ -4x + 30, & 5 \le x < 10 \\ -10, & 10 \le x < 20. \end{cases} \tag{17}$$

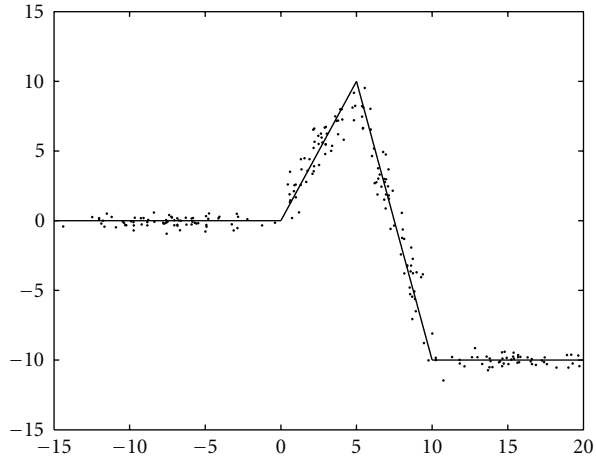

Figure 2: Training set associated with the toy function (17), represented by the continuous line.

The training set derived from this function is shown in Figure 2: 50 points per hyperplane, for a total of 200 points, are randomly sampled using (17) and a small perturbation by white Gaussian noise is also added in order to have a realistic data set.

The HHCS procedure starts with one rule and the hyperplane clustering is uniquely initialized in this case, since all patterns are initially assigned to the unique hyperplane. Its final determination, after the convergence of clustering, is represented in Figure 3. Assuming no network tuning is performed in this example, the splitting procedure illustrated in Section 3.3 is applied to the sole hyperplane that will be removed from the ANFIS network. More precisely, the patterns belonging to the hyperplane (i.e., all the training set in the case $M = 1$) are separated according to the sign of projection (16) along their principal direction. The resulting initialization of hyperplane clustering for $M = 2$ is shown in Figure 4, where the patterns associated with the new generated clusters are represented by crosses and circles, respectively. They are used for the first estimation of the hyperplanes, which are also evidenced in Figure 4.

The hyperplane clustering gets trapped in a local convergence in this case. In fact, the final determination of hyperplanes illustrated in Figure 5 is obtained after a few iterations and they are really similar to the initial estimate in Figure 4. In spite of this, the HHCS procedure continues and the hyperplane with negative slope will be split because it marks the largest local approximation error (13). After the splitting procedure, the initialization of hyperplane clustering for $M = 3$ is obtained, as illustrated in Figure 6. The patterns belonging to the split cluster are separated into two subsets similarly to the previous case. Obviously, the patterns associated with the other cluster are not involved in the splitting procedure.

Also in this case, there is a fast but local convergence to the configuration evidenced in Figure 7, which is similar to the former initialization. A hyperplane is then selected for splitting in order to obtain the initialization of HCS

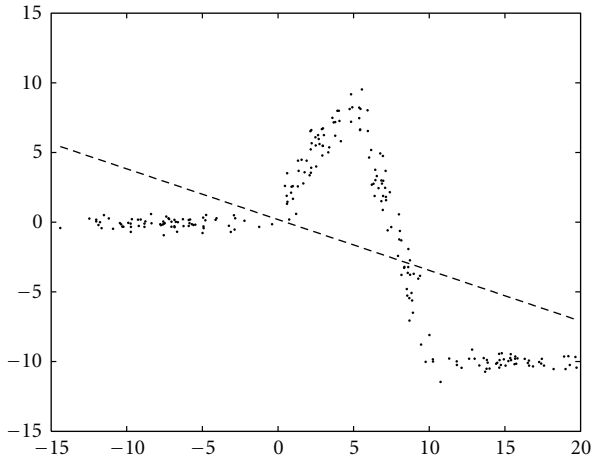

Figure 3: Final hyperplane (dashed line) estimated by the hyperplane clustering in the case $M = 1$.

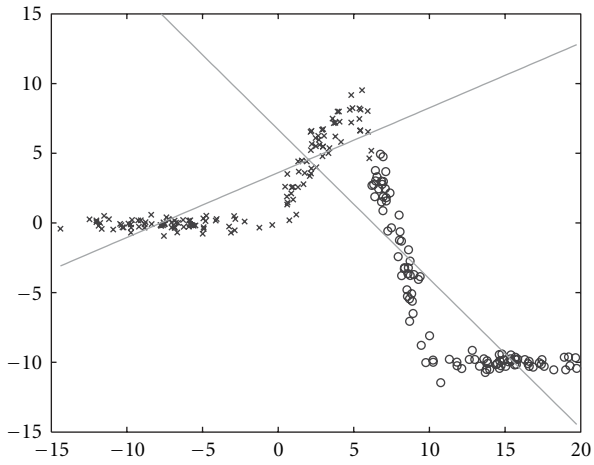

Figure 4: Initialization of hyperplane clustering for $M = 2$. Patterns belonging to the new generated clusters are represented by the symbols "x" and "o", respectively; the first estimate of hyperplanes based on these subsets is shown by the continuous gray lines.

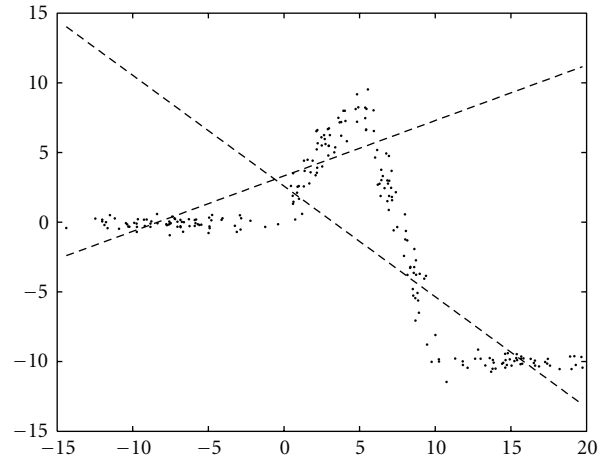

Figure 5: Final hyperplanes (dashed lines) estimated by the hyperplane clustering in the case $M = 2$.

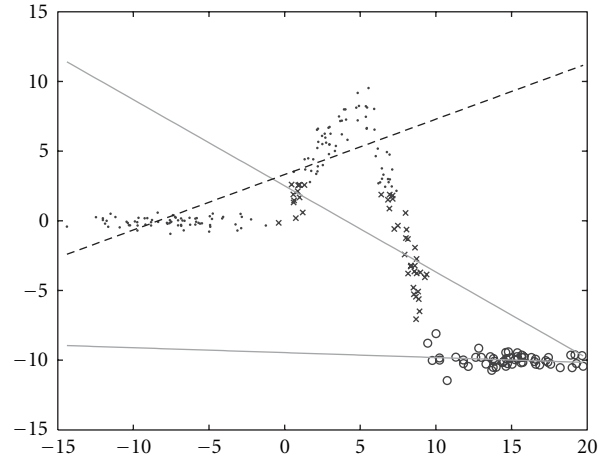

Figure 6: Initialization of hyperplane clustering for $M = 3$. Patterns belonging to the new generated clusters are represented by the symbols "x" and "o", respectively; the first estimate of hyperplanes based on these subsets is shown by the continuous gray lines. Patterns represented by simple dots belong to the cluster associated with the hyperplane that has not been split (dashed line).

illustrated in Figure 8 for the next HHCS iteration with $M = 4$. The convergence of hyperplane clustering is slower and accurate in this case and it yields the final configuration shown in Figure 9 where the linear components of (17) are well modeled now.

Finally, the global approximation performed by the ANFIS network obtained by the HHCS procedure with 4 rules is illustrated in Figure 10. The plotted function is obtained using a test set of 35000 patterns uniformly sampled in the input space, $x_i = -15 + 0.001(i - 1)$, $i = 1 \dots 35000$, $-15 \leq x_i < 20$, and determining the corresponding outputs $\tilde{y}_i$ using (2).

3.5. Computational Remarks. When dealing with the hyperplane-based synthesis of ANFIS networks, a number of considerations can be taken mostly pertaining to the inference of rules' antecedents from the consequents associated with the hyperplanes and to the fine tuning of ANFIS parameters after the HCS procedure. For example, when the nonlinear functions to be approximated are quite complicated, the number of HBs generated by the Min-Max classifier is often larger than the number of hyperplanes determined by the hyperplane clustering. Thus, a same hyperplane induces several HBs in the input space and can approximate patterns belonging to well-separated regions of the input space, each associated with a different HB.

Furthermore, a tuning procedure of ANFIS parameters can be applied after every HCS procedure or, alternatively, to the optimal ANFIS network only. A simple tuning step is considered in this paper, it is applied after every execution

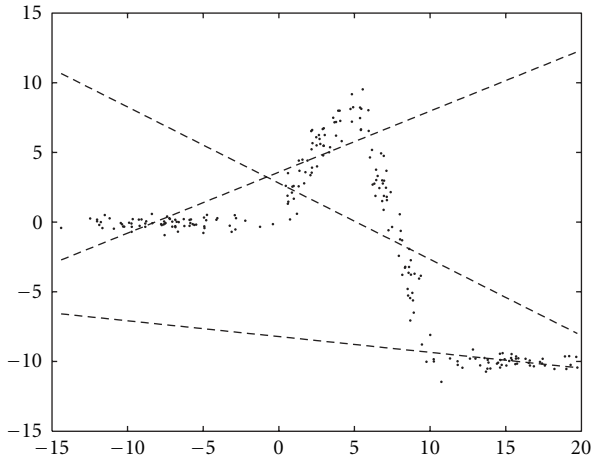

Figure 7: Final hyperplanes (dashed lines) estimated by the hyperplane clustering in the case $M = 3$.

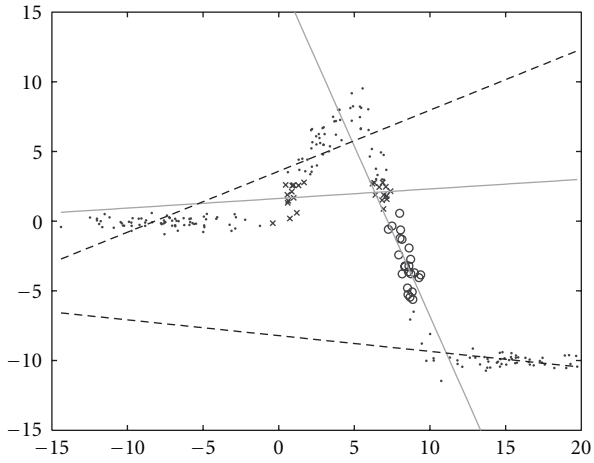

Figure 8: Initialization of hyperplane clustering for $M = 4$. Patterns belonging to the new generated clusters are represented by the symbols "x" and "o", respectively; the first estimate of hyperplanes based on these subsets is shown by the continuous gray lines. Patterns represented by simple dots belong to the clusters associated with the hyperplanes that have not been split (dashed lines).

of the HCS both in OHCS and HHCS procedures. The coefficients $a_j^{(k)}$, $j = 0 \dots n, k = 1 \dots M$, calculated during the hyperplane clustering are updated by a fuzzy least-squares solution of a set of $(n + 1)M$ linear equations, which are obtained by using the training set and by setting to zero the derivatives of the MSE (11) with respect to every unknown $a_j^{(k)}$:

$$\left\langle y_i \gamma_{ij}^{(k)} \right\rangle_i = \sum_{h=1}^{M} \sum_{r=0}^{n} a_r^{(h)} \left\langle \gamma_{ir}^{(h)} \gamma_{ij}^{(k)} \right\rangle_i \qquad (18)$$

$$j = 0 \dots n, k = 1 \dots M,$$

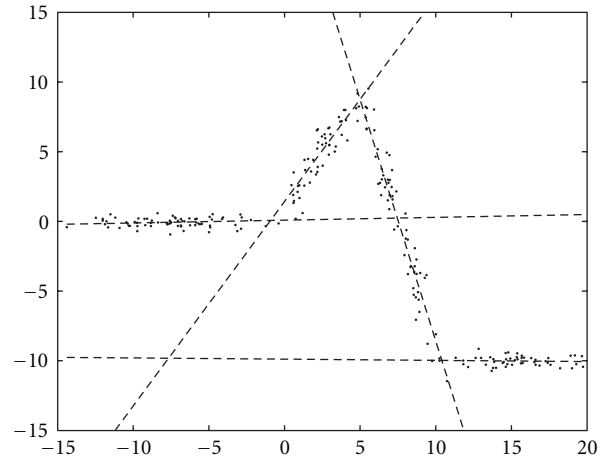

Figure 9: Final hyperplanes (dashed lines) estimated by the hyperplane clustering in the case $M = 4$.

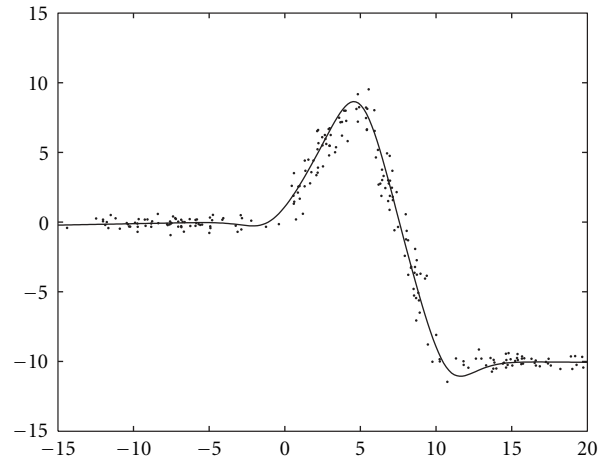

Figure 10: Approximation of the toy function (17) using the ANFIS network obtained by the HHCS procedure with 4 rules.

where $\langle \cdot \rangle_i$ denotes the average over subscript i and

$$\gamma_{ij}^{(k)} = \begin{cases} \dfrac{\mu_{\underline{B}^{(k)}}(\underline{x}_i)}{\sum_{h=1}^{M} \mu_{\underline{B}^{(h)}}(\underline{x}_i)} \, x_{ij}, & 1 \le j \le n \\ \dfrac{\mu_{\underline{B}^{(k)}}(\underline{x}_i)}{\sum_{h=1}^{M} \mu_{\underline{B}^{(h)}}(\underline{x}_i)}, & j = 0 \end{cases} \qquad (19)$$

$$k = 1 \dots M, \; i = 1 \dots P.$$

What is more important to remark is that HHCS allows the structural optimization of ANFIS networks and this is obtained by controlling the computational cost. Moreover, the increasing complexity of networks during training makes HHCS structurally constrained, similarly to other deterministic annealing or entropy constrained approaches [38–40]. In other words, overfitting due to the presence of outliers in the training set can be prevented when M is low, ensuring robustness with respect to noise in the function to be approximated.

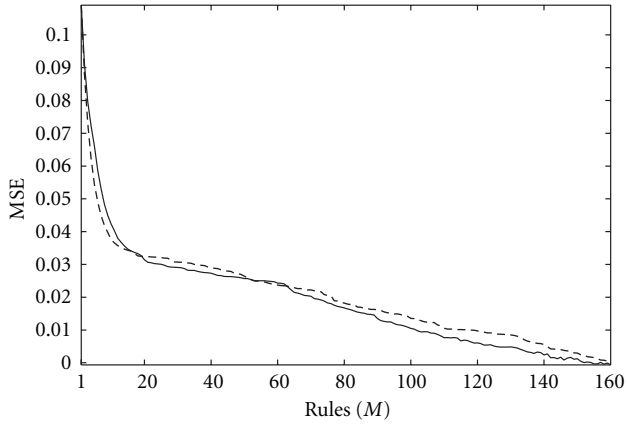

Figure 11: Prediction accuracy of the HCS algorithm: single execution in HHCS (continuous line); best execution in OHCS (dashed line).

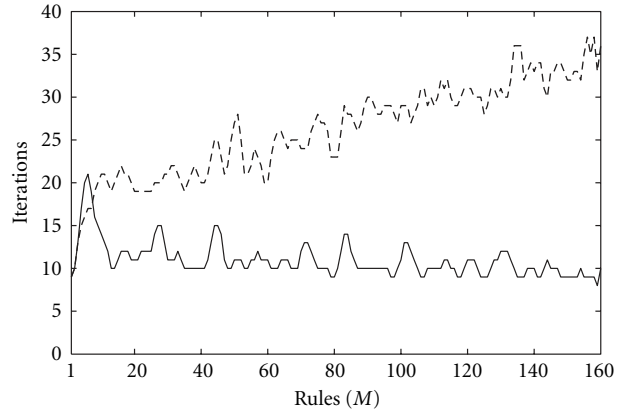

Figure 12: Convergence of the HCS algorithm: single execution in HHCS (continuous line); best execution in OHCS (dashed line).

Although the convergence of HHCS is guaranteed by the inner convergence of HCS algorithm and by the maximum complexity $M_{\max}$ allowed to the network, it should be noticed that HHCS does not ensure the determination of the global minimum of (8) for any value of M. As said, achieving a global optimum is inherently prevented by the HCS algorithm, being this an alternating minimization scheme. HHCS algorithm aims to obtain a suboptimal minimum ensuring an acceptable accuracy of the resulting ANFIS network, although the peculiarity of HHCS is to keep limited the computational cost of the whole structural optimization procedure. In fact, the methods commonly adopted in the literature for searching the global optimum are often characterized by an expensive requirement of computational resources. By the way, in the case of HCS algorithm, one can use some particular modifications that have been proposed in this regard [40–42] or more general approaches for searching the global minimum based, for example, on simulated annealing, genetic algorithms, or tabù search.

A characteristic behavior that usually occurs, revealing all the HHCS potentiality, is illustrated in the example pertaining to the approximation problem $f_1(\underline{x})$ described in Section 4. The final MSE of the data set versus the number of rules is plotted in Figure 11, while the number of HCS iterations needed to reach convergence is plotted in Figure 12. The continuous line is related to the single HCS execution in HHCS that generates just one ANFIS network for each value of M; the dashed line corresponds to the HCS execution yielding the ANFIS network that showed the best performance, among those differently initialized by applying the plain OHCS procedure (10 different initializations have been used for each value of M). It is evident that the performance of ANFIS networks obtained by HHCS is comparable, and even better, with respect to the ones obtained by OHCS. This also means that the faster HCS convergence in HHCS does not implies a poor local convergence. On the contrary, the lower number of HCS iterations in HHCS is due to the change of only one rule in the fuzzy inference system when M is increased.

4. Numerical Results

The validity of the HHCS procedure has been proved by extensive computer simulations. Some illustrative examples are summarized in the following, considering the numerical results obtained by well-known neural and neurofuzzy models applied to specific function approximation problems.

4.1. Synthetic Benchmarks. First of all, the HHCS procedure is compared with the plain OHCS in terms of accuracy, complexity and the computational cost necessary to determine the optimal network. Such quantities can be represented, respectively, by MSE, number of rules, overall number of HCS iterations (i.e., the cumulative sum of iterations in all the HCS executions required by the whole optimization procedure), and normalized time. We consider for the sake of comparison the well-known function approximation problems proposed in [43]:

(i) $f_1(\underline{x}) = C_a/(1 + e^{-C_b x_2}) + \Theta(x_1)$;

(ii) $f_2(\underline{x}) = C_c \sin^2(2\pi\sqrt{((5 - x_1)^2 + (5 - x_2)^2)/10})$;

(iii) $f_3(\underline{x}) = C_d(5 - x_2)^2/(3 \cdot (5 - x_1)^2 + (5 - x_2)^2)$;

with $\underline{x} = [x_1, x_2]$, $x_1, x_2 \in [0, 1]$, $\Theta(x_1)$ is a small perturbation of the output surface of the first example. A uniform spiral distribution was used to produce a training set of 400 examples for each function. As mentioned, the default values also used in the subsequent tests are $\lambda = 0.5$; $M_{\max} = 160$ (i.e., 40% of the training set cardinality); $T = 10$ (i.e., 10 different initializations carried out for every value of M in OHCS). The results shown in Table 1 evidence that HHCS has a computational cost considerably smaller than OHCS, while maintaining a good approximation accuracy and controlling the overall complexity in order to avoid overfitting.

Table 1: Comparison between HHCS and OHCS for different approximation problems.

Method	HHCS				OHCS			
	Rules	MSE	Iterations	Time	Rules	MSE	Iterations	Time
$f_1(\underline{x})$	21	3.06	1753	1.0	18	3.26	4187	3.51
$f_2(\underline{x})$	48	1.86	2420	1.0	81	1.14	4154	2.66
$f_3(\underline{x})$	41	1.07	1819	1.0	37	1.23	4183	3.27

MSE values are scaled by 10^{-2}.

Table 2: Comparison with other neural and neurofuzzy models.

Method	HHCS	OHCS	SOFRG	RBF	SMLP	IMLP
$f_1(\underline{x})$	3.06	3.26	4.90	3.10	2.60	5.20
$f_2(\underline{x})$	1.86	1.14	3.80	9.80	17.30	7.80
$f_3(\underline{x})$	1.07	1.23	1.20	1.90	2.10	2.70

MSE values are scaled by 10^{-2}.

The normalized time spent in OCHS with respect to HHCS is more than expected by considering the number of iterations only. As said, this is due to the slower convergence of the HCS procedure within OHCS since random initializations are used whenever.

By using the same data sets, the HHCS procedure is also compared with respect to other neural and neurofuzzy paradigms, in particular the self-organized fuzzy rule generation (SOFRG) method proposed in [43], where also the results of the well-known radial basis function (RBF) [44–46], standard multilayer perceptron (SMLP) [47], and momentum-improved multilayer perceptron (IMLP) [48] are reported. The MSE obtained by these systems on the said data sets are illustrated in Table 2 and once more, they prove that HHCS compare favorably with respect to the other systems and the related learning algorithm. We outline that, in order to obtain comparable results, we have considered the parameters setup of the benchmarks already reported in the literature, where the complexity of RBF, SMLP, and IMLP neural networks was optimized in advance to 25 neurons by using suited cross-validation experiments. Conversely, HHCS, OHCS, and SOFRG determine automatically the complexity of the neurofuzzy network.

The efficacy of the proposed HHCS approach is also evaluated in terms of generalization capability, which is the crucial characteristic of any neural system. Other well-known methods for ANFIS synthesis are considered [49]: the combination of the resilient propagation (RPROP) and the recursive least-squares error (RLSE) technique [50]; the gradient descent (GD) optimization joined to the RLSE method. The following three-input nonlinear function should be modeled:

$$\text{(iv)}\ f_4(\underline{x}) = 1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5},$$

where 216 points are used in the training set and other 125 points in the test set. The performance results are reported in Table 3 in terms of MSE. The optimal number of rules reported in the literature in the case of RPROP + RLSE and GD + RLSE algorithms is $M = 9$, while the constructive optimization of OHCS and HHCS yields ANFIS networks having fewer rules and showing a lower error on the test set and hence, a better generalization capability especially using HHCS.

Finally, the comparison with some gradient descent optimization techniques for the learning of fuzzy rule bases, in particular the Sugeno rules (TS-A) and centered-Sugeno rules (C-TS-A), is considered. Both training and test sets consist of 250 patterns, obtained by random sampling of the following two-input nonlinear function [51]:

$$\text{(v)}\ f_5(\underline{x}) = (3e^{2x_1} + 2e^{-4x_2})/170,\ x_1, x_2 \in [-1, 1].$$

The numerical results are shown in Table 4; the optimal number of rules obtained in [51] by the TS-A and C-TS-A approaches is $M = 16$; also in this case HHCS attains a good generalization capability on the test set even if a reduced number of rules is used with respect to OHCS, TS-A, and C-TS-A methods.

4.2. Time Series Prediction. The synthesis procedure proposed in this paper is also validated by considering the use of the resulting ANFIS network in a typical real-world application. More precisely, we focus on the prediction of biological time series, which are relevant to health monitoring and risk prevention in many daily activities as in clinical applications, telemedicine, road safety, and so on [52].

The general approach to solve a prediction problem is based on the solution of a suitable function approximation problem. In fact, a prediction problem can be solved by synthesizing the function that links the current sample to be predicted to a suitable set of past ones [53]. Let $S(t)$ be the sample of the time series to be predicted at time t and let the past samples be known up to time $t-1$, the estimated value $\widetilde{S}(t)$ can be obtained as

$$\begin{aligned} \widetilde{S}(t) &= f(\underline{x}), \quad f : \Re^n \to \Re, \\ \underline{x} &= [S(t-1)\ \ S(t-2)\ \ \ldots\ \ S(t-n)]. \end{aligned} \tag{20}$$

Hence, the implementation of a predictor coincides with the estimation of an approximation model as in (3), by using any data driven function approximation technique. Neurofuzzy networks are useful to solve such problems, because of

Table 3: Comparison with RPROP + RLSE and GD + RLSE methods.

Method	HHCS	OHCS	RPROP + RLSE	GD + RLSE
Rules	3	4	9	9
MSE (training set)	$4.75 \cdot 10^{-4}$	$1.39 \cdot 10^{-4}$	$6.67 \cdot 10^{-6}$	$8.60 \cdot 10^{-4}$
MSE (test set)	$7.92 \cdot 10^{-3}$	$1.27 \cdot 10^{-2}$	$4.74 \cdot 10^{-2}$	$6.69 \cdot 10^{-2}$

Table 4: Comparison with TS-A and C-TS-A methods.

Method	HHCS	OHCS	TS-A	C-TS-A
Rules	13	19	16	16
MSE (training set)	12.13	3.32	10.40	7.20
MSE (test set)	5.41	5.73	34.40	37.60

MSE values are scaled by 10^{-6}.

Table 5: Prediction of biological time series (dB).

Predictor	Glucose	Heart rate	Conductivity
LSE (training set)	4.541	27.082	12.879
LSE (test set)	2.845	21.258	12.614
RBF (training set)	22.127	29.435	14.707
RBF (test set)	21.817	23.582	11.787
MoG (training set)	23.996	28.133	16.269
MoG (test set)	22.195	22.329	12.136
HHCS (training set)	26.745	27.992	17.032
HHCS (test set)	23.576	24.855	12.863

the intrinsic nonlinearity and complexity of the underlying functions to be estimated [54]. The ANFIS network obtained by the HHCS procedure is compared with several data driven modeling techniques useful for approximation: a linear model determined by the well-known least-squares approximation (LSE); the RBF neural network; the Mixture of Gaussian (MoG) neural network, which is particularly suited to the solution of multivalued and nonconvex function approximation problems [40, 55].

The glucose time series is sampled every 20 minutes; the time series of heart rate is sampled every 5 seconds; the skin conductivity is sampled every one second. On the basis of a preliminary analysis of such time series, performed by experts in the context of clinical trials, three past samples are sufficient to predict the future value, that is, $n = 3$ is used in (20). Each predictor is trained on the first 500 samples of every time series (training set); the related performance is evaluated as the signal-to-noise ratio (SNR) in predicting the successive 150 samples (test set). The SNR is defined as the ratio, measured in decibels (dB), between the mean squared error of prediction and the variance of the time series to be predicted.

The performances obtained on both training and test set of the considered time series are illustrated in Table 5. Except for the linear model, RBF and MoG neural networks have been optimized by cross-validation and constructive procedures similarly to HHCS, in order to find the optimal number of parameters that maximizes the generalization capability. Looking at the performance on the test set, the ANFIS networks generated by the HHCS procedure obtain a higher SNR with respect to the other models for all the considered time series.

5. Conclusion

This paper focuses on the training of ANFIS networks with the aim to find effective solutions for improving the accuracy in actual regression problems. The novel HHCS procedure is proposed in the paper within the framework of hyperplane clustering synthesis of ANFIS networks; it involves a constructive approach that progressively increases the number of rules. This procedure ensures a structural robustness of the resulting network with respect to the overfitting phenomenon, while improving the convergence and reducing the computational cost in correspondence to any given number of rules. Thus, it is characterized by a high robustness and it is well suited to play the core role in more complex modeling systems.

The numerical results illustrated in the paper encourage to a further development of the HHCS approach, since the resulting ANFIS networks perform better than several neurofuzzy models and the related learning algorithms applied to well-known benchmarks and real-world problems reported in the literature.

References

[1] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[2] J. S. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy And Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, Upper Saddle River, NJ, USA, 1997.

[3] S. Haykin, *Neural Networks, A Comprehensive Foundation*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2nd edition, 1999.

[4] M. Panella, A. Rizzi, F. M. Frattale Mascioli, and G. Martinelli, "ANFIS synthesis by hyperplane clustering," in *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol. 1, pp. 340–345, Vancouver, Canada, July 2001.

[5] M. Panella and A. S. Gallo, "An input-output clustering approach to the synthesis of ANFIS networks," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 1, pp. 69–81, 2005.

[6] Z. He and A. Cichocki, "An efficient K-hyperplane clustering algorithm and its application to sparse component analysis," in *Proceedings of the 4th International Symposium on Neural Networks: Part II, Advances in Neural Networks*, D. Liu, S. Fei, Z.-G. Hou, H. Zhang, and C. Sun, Eds., vol. 4492 of *Lecture Notes in Computer Science*, pp. 1032–1041, Springer, Nanjing, China, 2007.

[7] A. Sharma, R. Podolsky, J. Zhao, and R. A. Mcindoe, "A modified hyperplane clustering algorithm allows for efficient and

accurate clustering of extremely large datasets," *Bioinformatics*, vol. 25, no. 9, pp. 1152–1157, 2009.

[8] J. Echanobe, I. del Campo, and G. Bosque, "An adaptive neuro-fuzzy system for efficient implementations," *Information Sciences*, vol. 178, no. 9, pp. 2150–2162, 2008.

[9] S. D. Nguyen and K. N. Ngo, "An adaptive input data space parting solution to the synthesis of neuro-fuzzy models," *International Journal of Control, Automation and Systems*, vol. 6, no. 6, pp. 928–938, 2008.

[10] S. H. Hosseini and A. H. Etemadi, "Adaptive neuro-fuzzy inference system based automatic generation control," *Electric Power Systems Research*, vol. 78, no. 7, pp. 1230–1239, 2008.

[11] I. D. Silva and R. Flauzino, "Efficient parametric adjustment of fuzzy inference system using error backpropagation method," in *Proceedings of the 19th International Conference on Artificial Neural Networks: Part I (ICANN '09)*, vol. 5768 of *Lecture Notes in Computer Science*, Springer, 2009.

[12] M. Panella and G. Martinelli, "Neurofuzzy networks with nonlinear quantum learning," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 3, pp. 698–710, 2009.

[13] M. Alizadeh, M. Lewis, M. H. F. Zarandi, and F. Jolai, "Determining significant parameters in the design of ANFIS," in *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '11)*, March 2011.

[14] M. Panella and G. Martinelli, "Neural networks with quantum architecture and quantum learning," *International Journal of Circuit Theory and Applications*, vol. 39, no. 1, pp. 61–77, 2011.

[15] H. Hui and J. H. Li, "Fingerprint matching using ANFIS," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 217–222, October 2003.

[16] F. Sun, H. Zhang, and H. Wu, "Neuro-fuzzy hybrid position/force control for a space robot with flexible dual-arms," in *Proceedings of the International Symposium on Neural Networks, Advances in Neural Networks (ISNN '04)*, vol. 3174 of *Lecture Notes in Computer Science*, Springer, 2004.

[17] X. K. Zhang, Y. C. Jin, and G. Guo, "ANFIS applied to a ship autopilot design," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 2006, pp. 2233–2236, August 2006.

[18] C. Mazzetti, F. M. Frattale Mascioli, F. Baldini, M. Panella, R. Risica, and R. Bartnikas, "Partial discharge pattern recognition by neuro-fuzzy networks in heat-shrinkable joints and terminations of XLPE insulated distribution cables," *IEEE Transactions on Power Delivery*, vol. 21, no. 3, pp. 1035–1044, 2006.

[19] P. Civicioglu, "Using uncorrupted neighborhoods of the pixels for impulsive noise suppression with ANFIS," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 759–773, 2007.

[20] S. Y. Na, D. Shin, J. Y. Kim, S. J. Baek, and S. H. Min, "Obstacle recognition and collision avoidance of a fish robot based on fuzzy neural networks," *Advances in Soft Computing*, vol. 40, pp. 337–344, 2007.

[21] D. Shin, S. Y. Na, J. Y. Kim, and S. J. Baek, "Fuzzy neural networks for obstacle pattern recognition and collision avoidance of fish robots," *Soft Computing*, vol. 12, no. 7, pp. 715–720, 2008.

[22] E. Joelianto, S. Widiyantoro, and M. Ichsan, "Time series estimation on earthquake events using ANFIS with mapping function," *International Journal of Artificial Intelligence*, vol. 3, no. 9, pp. 37–63, 2009.

[23] M. Kang and H. Kim, "A design of RFTOG model for distributed real-time applications," *Journal of Intelligent Manufacturing*, vol. 20, no. 3, pp. 311–319, 2009.

[24] J. Zhang, J. Cheng, and L. Li, "Forecasting coal and rock dynamic disaster based on adaptive neuro-fuzzy inference system," in *Proceedings of the 2nd International Conference on Computational Collective Intelligence: Technologies and Applications (ICCCI '10)*, vol. 6422 of *Lecture Notes in Computer Science*, Springer, 2010.

[25] M. Hayati, A. Rezaei, M. Seifi, and A. Naderi, "Modeling and simulation of combinational CMOS logic circuits by ANFIS," *Microelectronics Journal*, vol. 41, no. 7, pp. 381–387, 2010.

[26] R. Khatibi, M. A. Ghorbani, M. H. Kashani, and O. Kisi, "Comparison of three artificial intelligence techniques for discharge routing," *Journal of Hydrology*, vol. 403, no. 3-4, pp. 201–212, 2011.

[27] R. J. Hathaway and J. C. Bezdek, "Switching regression models and fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 3, pp. 195–204, 1993.

[28] M. Sugeno and T. Yasukawa, "Fuzzy-logic-based approach to qualitative modeling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7–31, 1993.

[29] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent & Fuzzy Systems*, vol. 2, pp. 267–278, 1994.

[30] R. Babuška and H. B. Verbruggen, "An overview of fuzzy modeling for control," *Control Engineering Practice*, vol. 4, no. 11, pp. 1593–1606, 1996.

[31] E. Kim, M. Park, S. Ji, and M. Park, "A new approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 328–337, 1997.

[32] E. Kim, M. Park, S. Kim, and M. Park, "A transformed input-domain approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 4, pp. 596–604, 1998.

[33] M. R. Emami, I. B. Türksen, and A. A. Goldenberg, "Development of a systematic methodology of fuzzy logic modeling," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 3, pp. 346–361, 1998.

[34] S. Guillaume, "Designing fuzzy inference systems from data: an interpretability-oriented review," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 426–443, 2001.

[35] P. K. Simpson, "Fuzzy min-max neural networks—I: classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776–786, 1992.

[36] J. Bezdek, *Pattern Recognition With Fuzzy Objective Functions Algorithms*, Plenum, New York, NY, USA, 1981.

[37] R. Nock and F. Nielsen, "On weighting clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1223–1235, 2006.

[38] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[39] C. C. Chuang, S. F. Su, and S. S. Chen, "Robust TSK fuzzy modeling for function approximation with outliers," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 6, pp. 810–821, 2001.

[40] M. Panella, A. Rizzi, and G. Martinelli, "Refining accuracy of environmental data prediction by MoG neural networks," *Neurocomputing*, vol. 55, no. 3-4, pp. 521–549, 2003.

[41] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.

[42] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton, "SMEM algorithm for mixture models," *Neural Computation*, vol. 12, no. 9, pp. 2109–2128, 2000.

[43] I. Rojas, H. Pomares, J. Ortega, and A. Prieto, "Self-organized fuzzy system generation from training examples," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 1, pp. 23–36, 2000.

[44] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[45] P. Wasserman, *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York, NY, USA, 1993.

[46] A. G. Borş and I. Pitas, "Median radial basis function neural network," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1351–1364, 1996.

[47] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Data Processing*, D. Rumelhart and J. McClelland, Eds., vol. 1, MIT Press, Cambridge, Mass, USA, 1986.

[48] H. C. Hsin, C. C. Li, M. Sun, and R. J. Sclabassi, "Adaptive training algorithm for back-propagation neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 3, pp. 512–514, 1995.

[49] M. Chen and R. Liou, "An efficient learning method of fuzzy inference systems," in *Procedings of the IEEE International Fuzzy Systems Conference Proceedings (FUZZ-IEEE '99)*, vol. 2, pp. 634–638, Seoul, Korea, 1999.

[50] G. Goodwin and K. Sin, *Adaptive Filtering Prediction and Control*, Prentice-Hall, New York, NY, USA, 1984.

[51] F. Guely and P. Siarry, "Gradient descent method for optimizing various fuzzy rule bases," in *Procedings of the 2nd IEEE International Conference on Fuzzy Systems*, pp. 1241–1246, April 1993.

[52] M. Panella, "Advances in biological time series prediction by neural networks," *Biomedical Signal Processing and Control*, vol. 6, no. 2, pp. 112–120, 2011.

[53] H. Abarbanel, *Analysis of Observed Chaotic Data*, Springer, New York, NY, USA, 1996.

[54] Z. Ghahramani, "Solving inverse problems using an EM approach to density estimation," in *Proceedings of the Connectionist Models Summer School*, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1994.

[55] M. Panella, A. Rizzi, F. M. Frattale Mascioli, and G. Martinelli, "A constructive EM approach to density estimation for learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, vol. 4, pp. 2608–2613, Washington, DC, USA, July 2001.