DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction

Nikola K. Kasabov, *Senior Member, IEEE,* and Qun Song

***Abstract*—This paper introduces a new type of fuzzy inference systems, denoted as dynamic evolving neural-fuzzy inference system (DENFIS), for adaptive online and offline learning, and their application for dynamic time series prediction. DENFIS evolve through incremental, hybrid (supervised/unsupervised), learning, and accommodate new input data, including new features, new classes, etc., through local element tuning. New fuzzy rules are created and updated during the operation of the system. At each time moment, the output of DENFIS is calculated through a fuzzy inference system based on m-most activated fuzzy rules which are dynamically chosen from a fuzzy rule set. Two approaches are proposed: 1) dynamic creation of a first-order Takagi–Sugeno-type fuzzy rule set for a DENFIS online model; and 2) creation of a first-order Takagi–Sugeno-type fuzzy rule set, or an expanded high-order one, for a DENFIS offline model. A set of fuzzy rules can be inserted into DENFIS before or during its learning process. Fuzzy rules can also be extracted during or after the learning process. An evolving clustering method (ECM), which is employed in both online and offline DENFIS models, is also introduced. It is demonstrated that DENFIS can effectively learn complex temporal sequences in an adaptive way and outperform some well-known, existing models.**

***Index Terms*—Dynamic evolving neural-fuzzy inference system (DENFIS), hybrid systems, online adaptive learning, online clustering, time series prediction.**

I. Introduction

THE complexity and dynamics of real-world problems, especially in engineering and manufacturing, require sophisticated methods and tools for building online, adaptive intelligent systems (ISs). Such systems should be able to grow as they operate, to update their knowledge and refine the model through interaction with the environment [2], [40], [41]. This is especially crucial when solving artificial intelligence (AI) problems such as adaptive speech and image recognition, multimodal information processing, adaptive prediction, adaptive online control, and intelligent agents on the world-wide web [7], [67].

Seven major requirements of the present ISs (that are addressed in the ECOS framework presented in [39] and [43]) are discussed in [37], [39], [40], and [43]. They are concerned with fast learning, online incremental adaptive learning, open structure organization, memorising information, active interaction, knowledge acquisition and self-improvement, and spatial and temporal learning.

Online learning is concerned with learning data as the system operates (usually in real time) and the data might exist only for a short time. Several investigations [21], [22], [32], [62]–[64] proved that the most popular neural network models and algorithms that include multilayer perceptrons (MLPs) trained with the back propagation (BP) algorithm, radial basis function (RBF) networks, and self-organizing maps (SOMs) are not suitable for adaptive, online learning. At the same time, several models that have elements of adaptive, online learning or structure and knowledge adaptation, have been developed that include connectionist models [1]–[4], [10]–[12], [16], [19], [21], [23], [25], [26], [30], [31], [33], [34], [45]–[49], [54], [57], [61], [64], [65], fuzzy logic models [69], [6], [29], [35], [51], [68], models based on genetic algorithms [18], [24], hybrid models [27], [35]–[38], [41], [42], [44], [51], [55], [68], evolving fuzzy-neural networks [37], [39], [40], [44], and evolving SOMs [17].

The evolving connectionist systems (ECOSs) framework [39] assumes that a system evolves its structure and functionality from a continuous input data stream in an adaptive, life-long, modular way. The system creates connectionist-based modules and connects them, if that is required according to the input data distribution and the system's performance at a certain time moment. ECOSs employ local learning (see, for example, [8], [56]).

The evolving fuzzy neural network (EFuNN) model was introduced in [40] as one way for creating connectionist modules in an ECOS architecture. In [37] and [44], the EFuNN model is further developed mainly in respect of dynamic parameter self-optimization. EFuNNs are fuzzy logic systems that have five-layer structures (Fig. 1). Nodes and connections are created/connected as data examples are presented. An optional short-term memory layer can be used through a feedback connection from the rule (also called case) node layer. The layer of feedback connections could be used if temporal relationships of input data are to be memorised structurally.

The input layer represents input variables. The second layer of nodes (fuzzy input neurons, or fuzzy inputs) represents fuzzy quantization of each input variable space. For example, two fuzzy input neurons can be used to represent "small" and "large" fuzzy values. Different membership functions (MFs) can be attached to these neurons (triangular, Gaussian, etc.). The number and the type of MF can be dynamically modified. The task of the fuzzy input nodes is to transfer the input values into membership degrees to which they belong to the MF.

Manuscript received June 13, 2001; revised August 6, 2001. This work was supported by a research program funded by the New Zealand Foundation for Research Science and Technology under Contract UOOX0016.

The authors are with the Department of Information Science, University of Otago, Dunedin 09015, New Zealand (e-mail: nkasabov@otago.ac.nz; qsong@infoscience.otago.ac.nz).

Publisher Item Identifier S 1063-6706(02)02965-X.

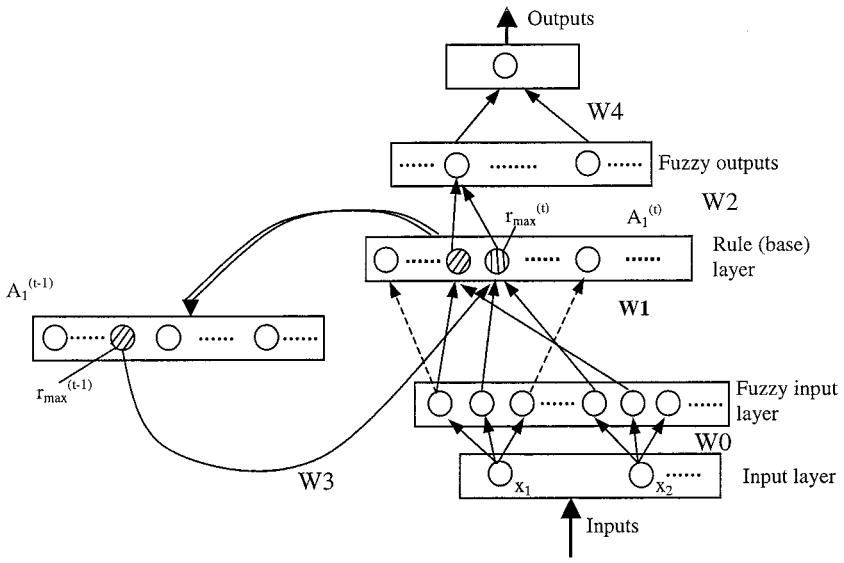

Fig. 1. The structure of EFuNN.

The third layer contains rule (case) nodes that evolve through supervised and/or unsupervised learning. The rule nodes represent prototypes (exemplars, clusters) of input–output data associations that can be graphically represented as associations of hyper-spheres from the fuzzy input and the fuzzy output spaces. Each rule node r is defined by two vectors of connection weights—$W1(r)$ and $W2(r)$, the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the problem space. A linear activation function is used for the neurons of this layer.

The fourth layer of neurons represents fuzzy quantization of the output variables, similar to the input fuzzy neuron representation. Here, a weighted sum input function and a saturated linear activation function is used for the neurons to calculate the membership degrees to which the output vector associated with the presented input vector belongs to each of the output MFs. The fifth layer represents the real values of the output variables. Here a linear activation function is used to calculate the defuzzified values for the output variables (similar to FuNN [42]).

Each rule node, e.g., r_j, represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space, the $W1(r_j)$ connection weights representing the coordinates of the center of the sphere in the fuzzy input space, and the $W2(r_j)$—the coordinates in the fuzzy output space. The radius of the input hyper-sphere of a rule node r_j is defined as $R_j = 1 - S_j$, where S_j is the sensitivity threshold parameter defining the minimum activation of the rule node r_j to a new input vector x from a new example (x, y) in order to be considered for association with this rule node. The pair of fuzzy input–output data vectors (x_f, y_f) will be allocated to the rule node r_j if x_f falls into the r_j input receptive field (hyper-sphere), and y_f falls in the r_j output reactive field hyper-sphere. This is ensured through two conditions, that a local normalized fuzzy difference between x_f and $W1(r_j)$ is smaller than the radius R_j, and the local normalized fuzzy difference between y_f and $W2(r_j)$ is smaller than the error threshold E_j. The latter represents the error tolerance of the system.

Definition: A local normalized fuzzy difference (distance) between two fuzzy membership vectors d_{1f} and d_{2f} that represent the membership degrees to which two real-value data vectors d_1 and d_2 belong to predefined MFs, is calculated as

$$D(d_{1f}, d_{2f}) = ||d_{1f} - d_{2f}||/||d_{1f} + d_{2f}|| \quad (1)$$

where: $||x - y||$ denotes the sum of all the absolute values of a vector that is obtained after vector subtraction (or summation in case of $||x+y||$) of two vectors x and y; '/' denotes division. For example, if $d_{1f} = (0, 0, 1, 0, 0, 0)$ and $d_{2f} = (0, 1, 0, 0, 0, 0)$, than $D(d_1, d_2) = (1 + 1)/2 = 1$ which is the maximum value for the local normalized fuzzy difference. In EFuNNs, the local normalized fuzzy distance is used to measure the distance between a new input data vector and a rule node in the local vicinity of the rule node.

Through the process of associating (learning) of new data points to a rule node r_j, the centers of this node hyper-spheres adjust in the fuzzy input space depending on the distance between the new input vector and the rule node through a learning rate l_{1j}, and in the fuzzy output space depending on the output error trough the Widrow-Hoff LMS algorithm (delta algorithm). This adjustment can be represented mathematically by the change in the connection weights of the rule node r_j from $W1(r_j^{(1)})$ and $W2(r_j^{(1)})$ to $W1(r_j^{(2)})$ and $W2(r_j^{(2)})$ respectively according to the following vector operations:

$$W1\left(r_j^{(2)}\right) = W1\left(r_j^{(1)}\right) + l_{1j} \cdot D\left(W1\left(r_j^{(1)}\right), x_f\right) \quad (2)$$

$$W2\left(r_j^{(1)}\right) = W2\left(r_j^{(1)}\right) + l_{2j} \cdot (A2 - y_f) \cdot A1\left(r_j^{(1)}\right) \quad (3)$$

where $A2$ is the activation vector of the fuzzy output neurons in the EFuNN structure when x is presented, and $A1(r_j^{(1)}) = 1 - D(W1(r_j^{(1)}), x_f)$ is the activation of the rule node $r_j^{(1)}$.

In initial algorithms for different types of learning in EFuNN, structures are presented in [40], that include: online,

offline, active, passive—sleep learning, etc. More sophisticated algorithms are included in [44] where different applications of EFuNN are also developed, such as adaptive speech recognition, gene expression data analysis and profiling, adaptive control.

Here, we propose a model called dynamic evolving neural fuzzy inference system (DENFIS). DENFIS is similar to EFuNN in some degree, and it inherits and develops EFuNNs dynamic features which make DENFIS suitable for online adaptive systems. The DENFIS model is developed with the idea that, depending on the position of the input vector in the input space, a fuzzy inference system for calculating the output is formed dynamically bases on m fuzzy rules that had been created during the past learning process.

This paper is organized as follows. Section II gives a description of an evolving clustering method (ECM) and its extension—evolving clustering method with constrained minimization (ECMc), both of which are used in the DENFIS model for partitioning the input space. A comparison between ECM, ECMc and some other clustering methods, such as EFuNN, fuzzy C-means [5], K-means [52], and subtractive clustering method [14], is also shown in this section on the same data set (Gas-furnace [9]). Section III introduces the DENFIS online model, and in Section IV, DENFIS online model is applied to Mackay–Glass (MG) time series [13], [15] prediction; the results are compared with the results obtained with the use of resource-allocation network (RAN) [60], EFuNN [40], and evolving self-organizing maps (ESOM) [17]. In Section V, two DENFIS offline models are introduced, and in Section VI, DENFIS offline models are applied to MG time series and Gas-furnace time series prediction. The results are compared with the results obtained with the use of adaptive neural-fuzzy inference system (ANFIS) [35], and the multilayer perceptrons trained with the back propagation algorithm (MLP-BP). Conclusions and directions for further research are presented in Section VII.

The comparative analysis clearly indicates the advantages of DENFIS when used for both offline, and especially online, learning applications. In addition to this, the ECMs, ECM and ECMc, can perform well as online, or offline, self-organized generic clustering models.

II. Evolving Clustering Method: ECM

Here, evolving, online, maximum distance-based clustering method, called ECM, is proposed to implement a scatter partitioning of the input space for the purpose of creating fuzzy inference rules. This method has two modes: the first one is usually applied to online learning systems, and the second one is more suitable for offline learning systems. The online ECM is used in the DENFIS online model. The offline ECM with constrained minimization (ECMc) is an extension of the online mode. It takes the result from the online mode as initial values. An optimization is then applied, that makes a predefined objective function based on a distance measure to reach a minimum value subject to given constraints.

A. Online ECM

Without any optimization, the online ECM is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in a set of data, and for finding their current centers in the input data space. It is a distance-based connectionist clustering method. With this method, cluster centers are represented by evolved nodes (rule nodes in the EFuNN terminology). In any cluster, the maximum distance, *MaxDist*, between an example point and the cluster center, is less than a threshold value, *Dthr*, that has been set as a clustering parameter and would affect the number of clusters to be estimated.

In this paper, the distance, between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, denotes a *general Euclidean distance* defined as follows:

$$\|\boldsymbol{x}-\boldsymbol{y}\| = \left(\sum_{i=1}^{q}|x_i-y_i|^2\right)^{\frac{1}{2}} \Big/ q^{\frac{1}{2}} \tag{4}$$

where $\boldsymbol{x},\boldsymbol{y} \in \boldsymbol{R}^q$.

In the clustering process, the data examples come from a data stream and this process starts with an empty set of clusters. When a new cluster is created, the cluster center, $\mathbf{Cc}$, is defined and its cluster radius, Ru, is initially set to zero. With more examples presented one after another, some created clusters will be updated through changing their centers' positions and increasing their cluster radiuses. Which cluster will be updated and how much it will be changed, depends on the position of the current example in the input space. A cluster will not be updated any more when its cluster radius, Ru, reaches the value that is equal to a threshold value, *Dthr*.

Fig. 2 shows a brief ECM clustering process in a 2-D space. The ECM algorithm is described as follows.

- Step 0: Create the first cluster C_1^0 by simply taking the position of the first example from the input stream as the first cluster center $\mathbf{Cc}_1^0$, and setting a value 0 for its cluster radius Ru_1 [Fig. 2(a)].
- Step 1: If all examples of the data stream have been processed, the algorithm is finished. Else, the current input example, $\boldsymbol{x}_i$, is taken and the distances between this example and all n already created cluster centers $\mathbf{Cc}_j, D_{ij} = \|\boldsymbol{x}_i - \mathbf{Cc}_j\|, j = 1,2,\ldots,n$, are calculated.
- Step 2: If there is any distance value, $D_{ij} = \|\boldsymbol{x}_i - \mathbf{Cc}_j\|$, equal to, or less than, at least one of the radii, $\mathrm{Ru}_j, j = 1,2,\ldots,n$, it means that the current example $\boldsymbol{x}_i$ belongs to a cluster C_m with the minimum distance

$$D_{im} = \|\boldsymbol{x}_i - \mathbf{Cc}_m\| = \min(\|\boldsymbol{x}_i - \mathbf{Cc}_j\|)$$
$$\text{subject to the constraint } D_{ij} \le \mathrm{Ru}_j, \quad j = 1,2,\ldots,n.$$

  In this case, neither a new cluster is created, nor any existing cluster is updated (the cases of $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$ in Fig. 2); the algorithm returns to Step 1. Else—go to the next step.
- Step 3: Find cluster C_a (with center $\mathbf{Cc}_a$ and cluster radius Ru_a) from all n existing cluster centers through calculating the values $S_{ij} = D_{ij} + \mathrm{Ru}_j, j = 1,2,\ldots,n$, and then choosing the cluster center $\mathbf{Cc}_a$ with the minimum value S_{ia}:

$$S_{ia} = D_{ia} + \mathrm{Ru}_a = \min(S_{ij}), \quad j = 1,2,\ldots,n.$$

- Step 4: If S_{ia} is greater than $2 \times Dthr$, the example $\boldsymbol{x}_i$ does not belong to any existing clusters. A new cluster is

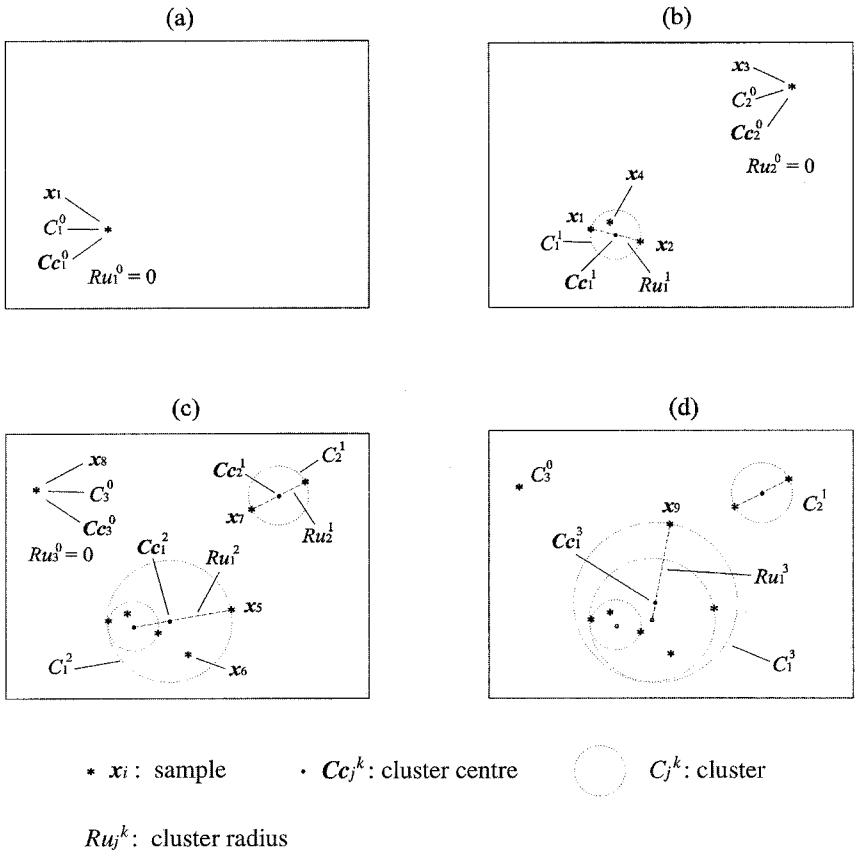

Fig. 2. A brief clustering process using ECM with samples $\boldsymbol{x}_1$ to $\boldsymbol{x}_9$ in a 2-D space. (a) The example $\boldsymbol{x}_1$ causes the ECM to create a new cluster C_1^0. (b) $\boldsymbol{x}_2$: update cluster $C_1^0 \rightarrow C_1^1$, $\boldsymbol{x}_3$: create a new cluster C_2^0, $\boldsymbol{x}_4$: do nothing. (c) $\boldsymbol{x}_5$: update cluster $C_1^1 \rightarrow C_1^2$, $\boldsymbol{x}_6$: do nothing, $\boldsymbol{x}_7$: update cluster $C_2^0 \rightarrow C_2^1$, $\boldsymbol{x}_8$: create a new cluster C_3^0. (d) $\boldsymbol{x}_9$: update cluster $C_1^2 \rightarrow C_1^3$.

created in the same way as described in Step 0 (the cases of $\boldsymbol{x}_3$ *and* $\boldsymbol{x}_8$ in Fig. 2), and the algorithm returns to Step 1.

- Step 5: If S_{ia} is not greater than $2 \times Dthr$, the cluster C_a is updated by moving its center, $\mathbf{Cc}_a$, and increasing the value of its radius, Ru_a. The updated radius $\mathrm{Ru}_a^{\mathrm{new}}$ is set to be equal to $S_{ia}/2$ and the new center $\mathbf{Cc}_a^{\mathrm{new}}$ is located at the point on the line connecting the $\boldsymbol{x}_i$ and $\mathbf{Cc}_a$, and the distance from the new center $\mathbf{Cc}_a^{\mathrm{new}}$ to the point $\boldsymbol{x}_i$ is equal to $\mathrm{Ru}_a^{\mathrm{new}}$ (the cases of $\boldsymbol{x}_2, \boldsymbol{x}_5, \boldsymbol{x}_7$ *and* $\boldsymbol{x}_9$ in Fig. 2). The algorithm returns to Step 1.

In this way, the maximum distance from any cluster center to the examples that belong to this cluster is not greater than the threshold value, *Dthr* though the algorithm does not keep any information of passed examples.

B. Constrained Optimization and Offline ECMc

The offline ECM, called ECMc, applies an optimization procedure to the resulted cluster centers after the application of ECM. The ECMc partitions a data set including p vector $\boldsymbol{x}_i, i = 1, 2, \ldots, p$, into n clusters $C_j, j = 1, 2, \ldots, n$, and finds a cluster center in each cluster, to minimize an objective function based on a distance measure subject to given constraints. Taking the *general Euclidean distance* as the measure between an example vector, $\boldsymbol{x}_i$, in cluster j and the corresponding cluster center, $\mathbf{Cc}_j$, the objective function is defined by the following equation:

$$J = \sum_{j=1}^{n} J_j = \sum_{j=1}^{n} \left(\sum_{\boldsymbol{x}_i \in C_j} ||\boldsymbol{x}_i - \mathbf{Cc}_j|| \right) \tag{5}$$

where $J_j = \sum_{\boldsymbol{x}_i \in C_j} ||\boldsymbol{x}_i - \mathbf{Cc}_j||$ is the objective function within cluster j

$$i = 1, 2, \ldots, p; \quad j = 1, 2, \ldots, n$$

and the constraints are defined by the next equation

$$||\boldsymbol{x}_i - \mathbf{Cc}_j|| \leq Dthr, \quad j = 1, 2, \ldots, n. \tag{6}$$

The partitioned clusters are typically defined by a $p \times n$ binary membership matrix $\boldsymbol{U}$, where the element u_{ij} is 1 if the i-th data point $\boldsymbol{x}_i$ belongs to cluster j; and 0 otherwise. Once the cluster centers $\mathbf{Cc}_j$ are fixed, the minimizing u_{ij} for (5) and (6) is derived as follows:

$$\begin{aligned} &\text{if } ||\boldsymbol{x}_i - \mathbf{Cc}_j|| \leq ||\boldsymbol{x}_i - \mathbf{Cc}_k||, \quad \text{for each } j \neq k; \\ &u_{ij} = 1, \quad \text{else } u_{ij} = 0. \end{aligned} \tag{7}$$

For a batch-mode operation, the method determines the cluster centers $\mathbf{Cc}_j$ and the membership matrix $\boldsymbol{U}$ iteratively using the following steps.

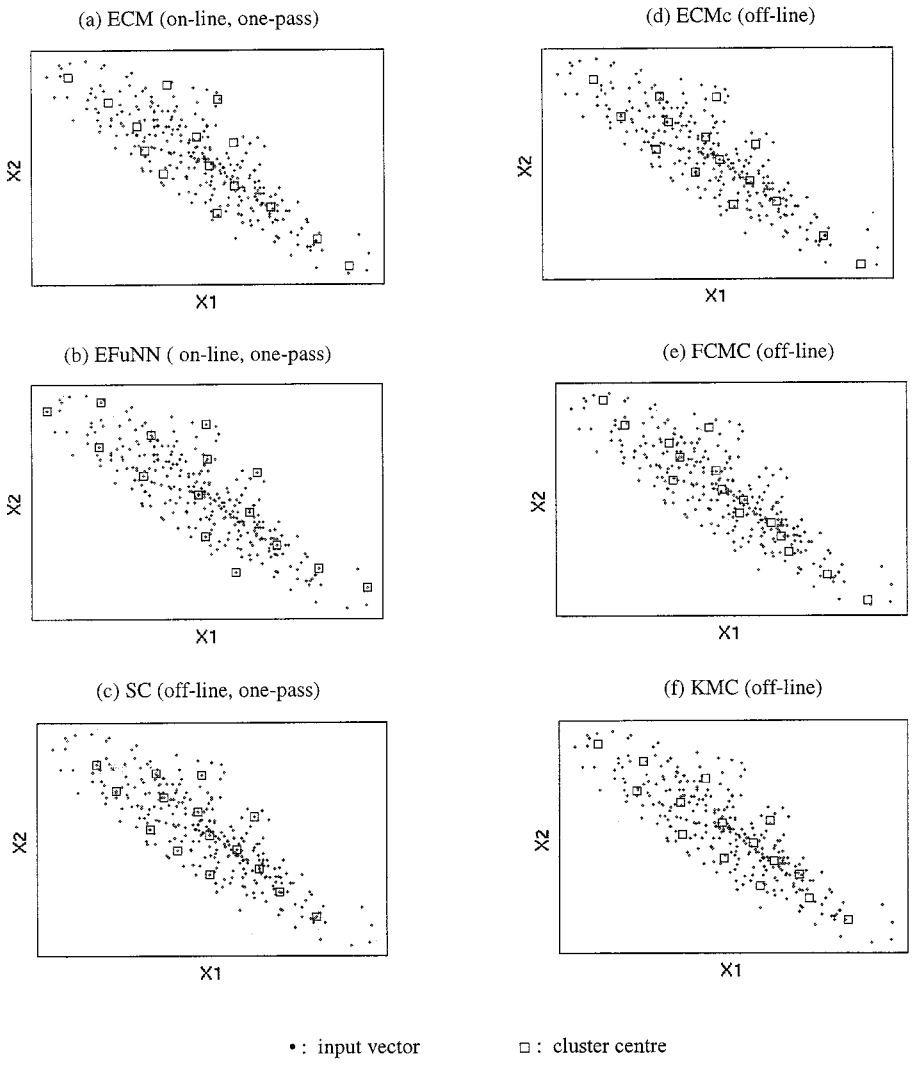

Fig. 3. Results of clustering the gas-furnace data set by several clustering methods.

- Step 1: Initialize the cluster center $\mathbf{Cc}_j, j = 1, 2, \ldots, n$, that come from the result of ECM clustering process.
- Step 2: Determine the membership matrix $\boldsymbol{U}$ by (7).
- Step 3: Employ the *constrained minimization method* [59] with (5) and (6) to get new cluster centers.
- Step 4: Calculate the objective function J according to (6). Stop if the result is below a certain tolerance value, or its improvement over previous iteration is below a certain threshold, or the iteration number of minimization operations is over a certain value. Else, the algorithm returns to Step 2.

C. Application: Clustering of the Gas-Furnace Data Set

The gas-furnace time series is a well-known bench-mark data set and has been frequently used by many researches in the area of neural networks and fuzzy system for control, prediction and adaptive learning [6], [20]. It consists of 296 consecutive data pairs of methane at a time moment $(t-4)$, and the carbon dioxide (CO_2) produced in a furnace at a time moment $(t-1)$ as input variables, with the produced CO_2 at the moment (t) as an output variable. In this case, the clustering simulations are implemented in the input space. For a comparative analysis, the following six clustering methods are implemented and applied to the gas-furnace data set:

a) ECM (online, one-pass);
b) EFuNN (online, one pass) [40];
c) SC (offline, one pass);
d) ECMc (offline);
e) FCMC (offline);
f) KMC (offline).

Each of them partitions the data into NoC $(=15)$ clusters. The maximum distance, $\mathrm{Max}\,D$, between an example point and the corresponding cluster center and the value of objective function J, defined by (2), are taken as criteria for comparison. The results are shown in Table I and Fig. 3. We can see from Table I, that ECMs, both ECM and ECMc, can obtain the minimum value of $\mathrm{Max}\,D$, which means that these methods partition the data set more uniformly than the other methods. From another point of view, we can say that if all six clustering methods obtained the same values for $\mathrm{Max}\,D$, the ECM and the ECMc could achieve less number of partitions than the others.

TABLE I
RESULTS OBTAINED BY USING DIFFERENT CLUSTERING METHODS FOR CLUSTERING THE GAS-FURNACE DATA SET INTO 15 CLUSTERS

Methods	*MaxD*	Objective function value: *J*
ECM (on-line, one pass)	0.1	12.9
EFuNN (on-line, one pass)	0.11	13.3
SC (off-line, one-pass)	0.15	11.5
ECMc(off-line)	0.1	11.5
FCM (off-line learning)	0.14	12.4
KM (off-line learning)	0.12	11.8

III. DENFIS: A DYNAMIC EVOLVING NEURAL-FUZZY INFERENCE SYSTEM

A. General Principle

The DENFIS, both online and offline models, use Takagi–Sugeno type fuzzy inference engine [66]. Such inference engine used in DENFIS is composed of m fuzzy rules indicated as shown in the equation at the bottom of the page, where "x_j is R_{ij}," $i = 1,2,\dots m; j = 1,2,\dots q$, are $m \times q$ fuzzy propositions as m antecedents form m fuzzy rules respectively; $x_j, j = 1,2,\dots,q$, are antecedent variables defined over universes of discourse $X_j, j = 1,2,\dots,q$, and $R_{ij}, i = 1,2,\dots m; j = 1,2,\dots,q$, are fuzzy sets defined by their fuzzy membership functions μ_{Rij}: $X_j \rightarrow [0,1], i = 1,2,\dots m; j = 1,2,\dots,q$. In the consequent parts, y is a consequent variable, and polynomial functions $f_i, i = 1,2,\dots m$, are employed.

In both DENFIS online and offline models, all fuzzy membership functions are triangular type functions which depend on three parameters as given by the following equation:

$$\mu(x) = mf(x,a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0. & c \le x \end{cases} \tag{8}$$

where: b is the value of the cluster center on the x dimension, $a = b - d \times Dthr$ and $c = b + d \times Dthr$, $d = 1.2$–2; the threshold value, *Dthr*, is a clustering parameter.

If the consequent functions are crisp constants, i.e., $f_i(x_1,x_2,\dots,x_q) = C_i, i = 1,2,\dots m$, we call such system a zero-order Takagi–Sugeno type fuzzy inference system. The system is called a first-order Takagi–Sugeno type fuzzy inference system if $f_i(x_1,x_2,\dots,x_q), i = 1,2,\dots m$, are linear functions. If these functions are nonlinear functions, it is called high-order Takagi–Sugeno fuzzy inference system.

For an input vector $x^0 = [x_1^0\ x_2^0\ \cdots\ x_q^0]$, the result of inference, y^0 (the output of the system) is the weighted average of each rule's output indicated as follows:

$$y^0 = \frac{\sum_{i=1}^{m} \omega_i f_i\left(x_1^0, x_2^0, \dots, x_q^0\right)}{\sum_{i=1}^{m} \omega_i}$$

where, $\omega_i = \prod_{j=1}^{q} \mu_{Rij}(x_j^0); i = 1,2,\dots m, j = 1,2,\dots,q$.

B. Learning Processes in DENFIS Online Model

In the DENFIS online model, the first-order Takagi–Sugeno type fuzzy rules are employed and the linear functions in the consequences can be created and updated by linear least-square estimator (LSE) [28], [33] with learning data. Each of the linear functions can be expressed as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1 + \cdots + \beta_q x_q.$$

For obtaining these functions a learning data set, which is composed of p data pairs $\{([x_{i1}, x_{i2}, \dots, x_{iq}], y_i), i = 1,2,\dots,p\}$, is used and the least-square estimator (LSE) of $\boldsymbol{\beta} = [b_0\ b_1\ b_2\ \cdots\ b_q]^T$ are calculated as the cofficients $mbib = [b_0\ b_1\ b_2\ \cdots\ b_q]^T$, by applying the following formula:

$$\boldsymbol{b} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{y} \tag{9}$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1q} \\ 1 & x_{21} & x_{22} & \cdots & x_{2q} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{p1} & x_{p2} & \cdots & x_{pq} \end{pmatrix}$$

and

$$\boldsymbol{y} = [y_1\ y_2 \cdots, y_p]^T.$$

In the DENFIS models, we use a weighted least-square estimation method [28], [33]

$$\boldsymbol{b}_w = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\boldsymbol{y} \tag{10}$$

where

$$\mathbf{W} = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & w_p \end{pmatrix}$$

and w_j is the distance between jth example and the corresponding cluster center, $j = 1,2,\dots p$. We can rewrite the (9) and (10) as follows:

$$\begin{cases} \mathbf{P} = (\mathbf{A}^T\mathbf{A})^{-1} \\ \boldsymbol{b} = \mathbf{P}\mathbf{A}^T\boldsymbol{y}. \end{cases} \tag{11}$$

$$\begin{cases} \mathbf{P}_w = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1} \\ \boldsymbol{b}_w = \mathbf{P}_w\mathbf{A}^T\mathbf{W}\boldsymbol{y} \end{cases} \tag{12}$$

$$\begin{cases} \text{if } x_1 \text{ is } R_{11} \text{ and } x_2 \text{ is } R_{12} \text{ and } \cdots \text{ and } x_q \text{ is } R_{1q}, \text{ then } y \text{ is } f_1(x_1,x_2,\dots,x_q) \\ \text{if } x_1 \text{ is } R_{21} \text{ and } x_2 \text{ is } R_{22} \text{ and } \cdots \text{ and } x_q \text{ is } R_{2q}, \text{ then } y \text{ is } f_2(x_1,x_2,\dots,x_q) \\ \cdots \\ \text{if } x_1 \text{ is } R_{m1} \text{ and } x_2 \text{ is } R_{m2} \text{ and } \cdots \text{ and } x_q \text{ is } R_{mq}, \text{ then } y \text{ is } f_m(x_1,x_2,\dots,x_q) \end{cases}$$

Let the k-th row vector of matrix A defined in (9) be $a_k^T = [1\ x_{k1}\ x_{k2}\ \cdots\ x_{kq}]$ and the k-th element of $\boldsymbol{y}$ be y_k, then $\boldsymbol{b}$ can be calculated iteratively as shown in (13) at the bottom of the page. Here, the initial values of $\mathbf{P}_n$ and $\boldsymbol{b}_n$ can be calculated directly from (12) with the use of the first n data pairs from the learning data set.

Equation (13) is the formula of recursive LSE [28]. In the DENFIS online model, we use a weighted recursive LSE with forgetting factor defined as (14) shown at the bottom of the page, where w is the weight defined in (10) and λ is a forgetting factor which typical value is between 0.8–1.

In the online DENFIS model, the rules are created and updated at the same time with the input space partitioning using online ECM and (8) and (14). If no rule insertion is applied, the following steps are used for the creation of the first m fuzzy rules and for the calculation of the initial values $\mathbf{P}$ and $\boldsymbol{b}$ of the functions.

1) Take the first n_0 learning data pairs from the learning data set.
2) Implement clustering using ECM with these n_0 data to obtaining m cluster centers.
3) For every cluster center $\boldsymbol{C}_i$, find p_i data points whose positions in the input space are closest to the center, $i = 1, 2, \ldots, m$.
4) For obtaining a fuzzy rule corresponding to a cluster center, create the antecedents of the fuzzy rule using the position of the cluster center and (8). Using (12) on p_i data pairs calculate the values of $\mathbf{P}$ and $\boldsymbol{b}$ of the consequent function. The distances between p_i data points and the cluster center are taken as the weights in (12).

In the above steps, m, n_0 and p are the parameters of the DENFIS online learning model, and the value of p_i should be greater than the number of input elements, q.

As new data pairs are presented to the system, new fuzzy rules may be created and some existing rules are updated. A new fuzzy rule is created if a new cluster center is found by the ECM. The antecedent of the new fuzzy rule is formed by using (8) with the position of the cluster center (as a rule node). An existing fuzzy rule is found which rule node is the closest to the new rule node; the consequence function of this rule is taken as the consequence function for the new fuzzy rule. For every data pair, several existing fuzzy rules are updated by using (14) if their rule nodes have distances to the data point in the input space that are not greater than $2 \times Dthr$ (the threshold value, a clustering parameter). The distances between these rule nodes and the data point in the input space are taken as the weights in (14). In addition to this, one of these rules may also be updated through changing its antecedent so that, if its rule node position is changed by the ECM, the fuzzy rule will have a new antecedent calculated through (8).

C. Takagi-Sugeno Fuzzy Inference in DENFIS

The Takagi–Sugeno fuzzy inference system utilized in DENFIS is a dynamic inference. In addition to dynamically creating and updating fuzzy rules the DENFIS online model has some other major differences from the other inference systems.

First, for each input vector, the DENFIS model chooses m fuzzy rules from the whole fuzzy rule set for forming a current inference system. This operation depends on the position of the current input vector in the input space. In the case of two input vectors that are very close to each other, especially in the DENFIS offline model, the inference system may have the same fuzzy rule inference group. In the DENFIS online model, however, even if two input vectors are exactly the same, their corresponding inference systems may be different. It is due to the reason that these two input vectors are presented to the system at different time moments and the fuzzy rules used for the first input vector might have been updated before the second input vector has arrived.

Second, depending on the position of the current input vector in the input space, the antecedents of the fuzzy rules chosen to form an inference system for this input vector may vary. An example is illustrated in the Fig. 4 where two different groups of fuzzy inference rules are formed depending on two input vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, in a 2-D input space as shown in Fig. 4(a) and (b), respectively. We can see from this example that, for instance, the region C has a linguistic meaning 'large', in the X_1 direction for Fig. 4(a) group, but for the group of rules from Fig. 4(b) it denotes a linguistic meaning of 'small' in the same direction of $X1$. The region C is defined by different membership functions, respectively, in each of the two groups of rules.

$$\begin{cases} \boldsymbol{b}_{k+1} = \boldsymbol{b}_k + \mathbf{P}_{k+1}\mathbf{a}_{k+1}\left(y_{k+1} - \mathbf{a}_{k+1}^T\boldsymbol{b}_k\right) \\ \mathbf{P}_{k+1} = \mathbf{P}_k - \dfrac{\mathbf{P}_k\mathbf{a}_{k+1}\mathbf{a}_{k+1}^T\mathbf{P}_k}{1 + \mathbf{a}_{k+1}^T\mathbf{P}_k\mathbf{a}_{k+1}}, \quad k = n, n+1, \ldots p-1. \end{cases} \tag{13}$$

$$\begin{cases} \boldsymbol{b}_{k+1} = \boldsymbol{b}_k + w_{k+1}\mathbf{P}_{k+1}\mathbf{a}_{k+1}\left(y_{k+1} - \mathbf{a}_{k+1}^T\boldsymbol{b}_k\right) \\ \mathbf{P}_{k+1} = \frac{1}{\lambda}\left(\mathbf{P}_k - \dfrac{w_{k+1}\mathbf{P}_k\mathbf{a}_{k+1}\mathbf{a}_{k+1}^T\mathbf{P}_k}{\lambda + \mathbf{a}_{k+1}^T\mathbf{P}_k\mathbf{a}_{k+1}}\right), \quad k = n, n+1, \ldots p-1 \end{cases} \tag{14}$$

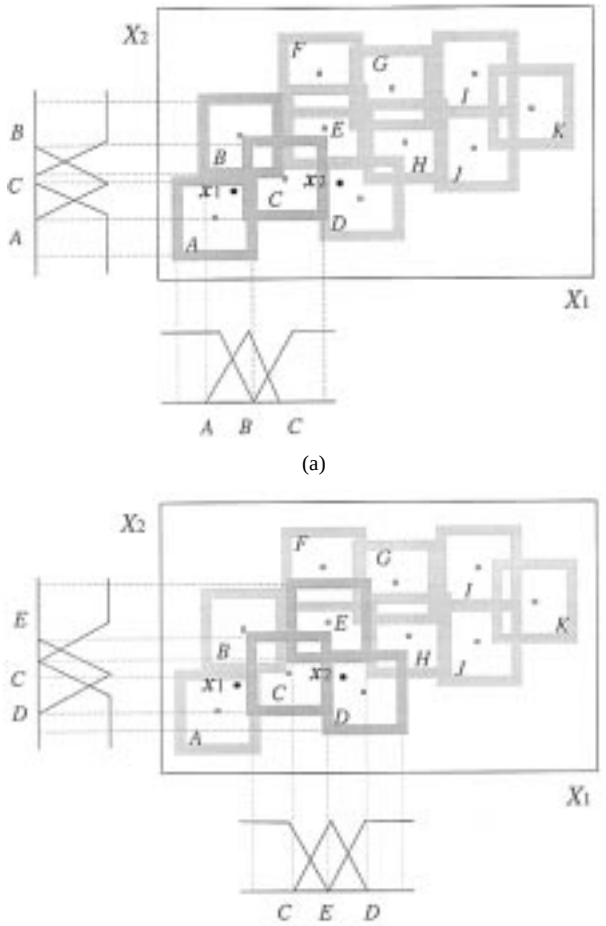

Fig. 4. Two fuzzy rule groups are formed by DENFIS to perform inference for an input vector $\boldsymbol{x}_1$ (a) and for an input vector $\boldsymbol{x}_2$ (b) that is entered at a later time moment, all represented in the 2-D space of the first two input variables $X1$ and $X2$.

IV. Time Series Modelling and Prediction With the DENFIS On-Line Model

In this section, the DENFIS online model will be applied to modeling and predicting the future values of a chaotic time series: the MG data set [13], which has been used as a benchmark example in the areas of neural networks, fuzzy systems and hybrid systems. This time series is created with the use of the MG time-delay differential equation defined as

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{15}$$

To obtain values at integer time points, the fourth-order Runge–Kutta method was used to find the numerical solution to the above MG equation. Here, we assume that time step is 0.1, $x(0) = 1.2, \tau = 17$ and $x(t) = 0$ for $t < 0$. The task is to predict the values $x(t+85)$ from input vectors $[x(t-18)\ x(t-12)\ x(t-6)\ x(t)]$ for any value of the time t. For the purpose of a comparative analysis, we also use some existing online learning models applied on the same task. These models are Neural gas [23], RAN [60], ESOM [17], and EFuNN [40]. Here, we take the nondimensional error index (NDEI) [15] which is defined as the root mean-square error (RMSE) divided by the standard deviation of the target series.

TABLE II
Prediction Results of On-Line Learning Models on the Mackey-Glass Test Data

Methods	Fuzzy rules (DENFIS) Rule nodes (EFuNN) Units (others)	NDEI for testing data
Neural gas [23]	1000	0.062
RAN [60]	113	0.373
RAN [60]	24	0.17
ESOM [17]	114	0.32
ESOM [17]	1000	0.044
EFuNN[40]	193	0.401
EFuNN[40]	1125	0.094
DENFIS	58	0.276
DENFIS	883	0.042
DENFIS with rule insertion	883	0.033

The following experiment was conducted: 3000 data points, for $t = 201$ to 3200, are extracted from the time series and used as learning (training) data; 500 data points, for $t = 5001$ to 5500, are used as testing data. For each of the aforementioned online models, the learning data is used for the online learning processes, and then the testing data is used with the recalling procedure.

In another experiment, the properties of rule insertion and rule extraction were utilized where we first obtained a group of fuzzy rules from the first half of training data (1500 samples), using the DENFIS offline model I (it will be introduced in next section); then we inserted these rules to the DENFIS online model and let it learn continuously from the next half of the learning data (1500 samples). Then, we tested the model on the test data.

Table II lists the prediction results (NDEI on test data after online learning) and the number of rules (nodes, units) evolved (used) in each model.

Fig. 5(a), (b), and (c) display the testing errors (from the recall processes on the test data) of DENFIS online model with different number of fuzzy rules

a) DENFIS online model with 58 fuzzy rules;
b) DENFIS online model with 883 fuzzy rules (different parameter values are used from those in the model above);
c) DENFIS online model with 883 fuzzy rules that is evolved after an initial set of rules was inserted.

V. DENFIS Model for Offline Learning

The DENFIS online model presented in the previous section, can be used also for offline, batch mode training, but it may not be very efficient when used on comparatively small data sets. For the purpose of batch training the DENFIS online model is extended here to work efficiently in an offline, batch training mode. Two DENFIS models for offline learning are developed and presented here: (1) a linear model, model I, and (2) a MLP-based model, model II.

A first-order Takagi–Sugeno type fuzzy inference engine, similar to the DENFIS online model, is employed in model

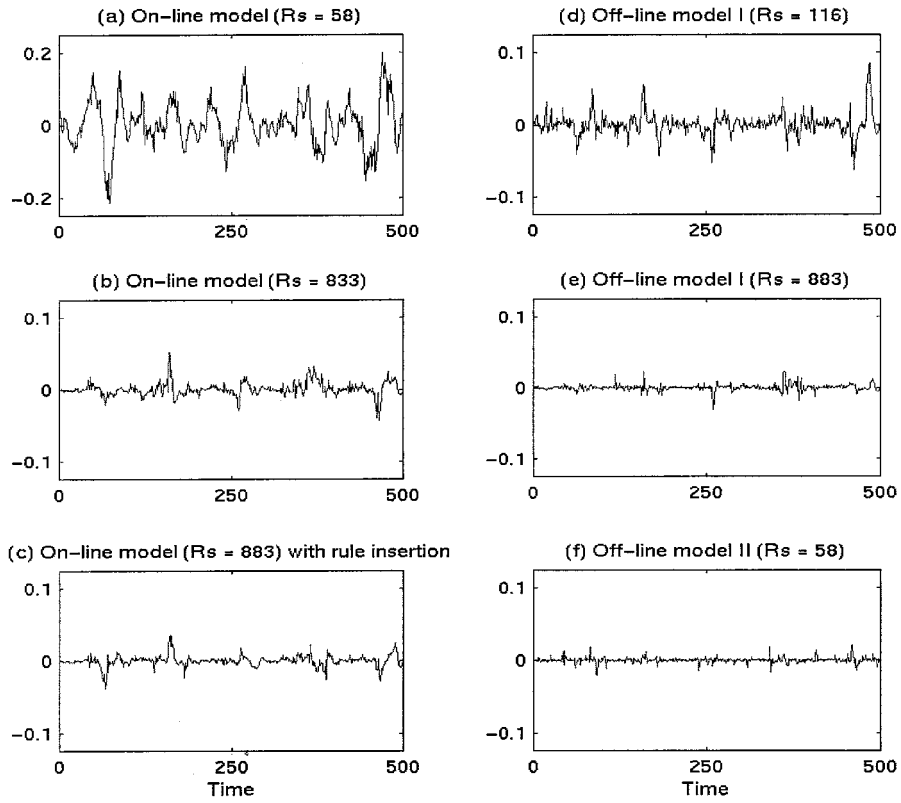

Fig. 5. Prediction error of DENFIS online (a), (b), (c) and offline (d), (e), (f) models on test data taken from the MG time series.

TABLE III
PREDICTION RESULTS OF OFF-LINE LEARNING MODELS ON MACKEY-GLASS TRAINING AND TEST DATA

Methods	Neurons or Rules	Epochs	Training Time (s)	Training NDEI	Testing NDEI
MLP-BP	60	50	1779	0.083	0.090
MLP-BP	60	500	17928	0.021	0.022
ANFIS	81	50	23578	0.032	0.033
ANFIS	81	200	94210	0.028	0.029
DENFIS I	116	2	352	0.068	0.068
DENFIS I	883	2	1286	0.023	0.019
DENFIS II	58	100	351	0.017	0.016

I, while an extended high-order Takagi–Sugeno fuzzy inference engine is used in model II. The latter employs several small-size, two-layer (the hidden layer consists of two or three neurons) multilayer perceptrons to realize the function f in the consequent part of each fuzzy rule instead of using a predefined function.

The DENFIS offline learning process is implemented in following way:

- cluster (partition) the input space to find n cluster centers (n rule nodes, n rules) by using the offline ECMc;
- create the antecedent part for each fuzzy rule using (8) and the current position of the cluster center (rule node);
- find n data sets each of them including one cluster center and p learning data pairs that are closest to the center in the input space. In the general case, one data pair can belong to several clusters;
- for model I, estimate the functions f to create the consequent part for each fuzzy rule using (10) or (12) with n data sets; the distances between each data point and their corresponding center is represented as a connection weight;
- for model II, each consequent function f of a fuzzy rule (rule node, cluster center) is learned by a corresponding MLP network after training it on the corresponding data set with the use of the MLP-BP.

VI. TIME SERIES MODELLING AND PREDICTION WITH THE DENFIS OFFLINE MODEL

Dynamic time-series modeling of complex time series is a difficult task, especially when the type of the model is not known in advance [50]. In this section, we applied the two DENFIS offline models for the same task as in Section IV. For comparison

purposes two other well-known models, adaptive neural-fuzzy inference system (ANFIS) [35], and a multilayer perceptron trained with MLP-BP [58], are also used for this task under the same conditions.

In addition to the NDEI, in the case of offline learning, the learning time is also measured as another comparative criterion. Here, the learning time is the CPU-time (in seconds) consumed by each method during the learning process in the same computing environment (MATLAB, UNIX version 5.5).

Table III lists the offline prediction results of MLP, ANFIS and DENFIS, and these results include the number of fuzzy rules (or rule nodes) in the hidden layer, learning epochs, learning time (CPU-time), NDEI for training data and NDEI for testing data. The best results are achieved in the DENFIS II model.

Fig. 5(d), (e), and (f) shows the prediction error of three DENFIS models tested on the same test data as follows:

(d) DENFIS offline mode I with 116 fuzzy rules;
(e) DENFIS offline mode I with 883 fuzzy rules;
(f) DENFIS offline mode II with 58 fuzzy rules.

The prediction error of DENFIS model II with 58 rule nodes is the lowest one.

VII. Conclusion and Directions for Further Research

This paper presents the principles of a fuzzy inference system, DENFIS, for building both online and offline knowledge-based, adaptive learning systems. Both DENFIS online and offline models are based on the Takagi–Sugeno fuzzy inference system. They use the m highly activated fuzzy rules to dynamically compose an inference system for calculating the output vector for a given input vector. The proposed systems demonstrate superiority when compared with Neural gas [23], RAN [60], EFuNN [40], and ESOM [17] in the case of online learning, and with ANFIS [35], and MLP [38] in the case of offline learning.

DENFIS uses a local generalization, like EFuNN and CMAC neural networks [1], so it needs more training data than the models which use global generalization such as ANFIS and MLP. During the learning process DENFIS forms an area of partitioned regions, but these regions may not cover the whole input space. In the recall process, DENFIS would give satisfactory results if the recall examples appear inside of these regions. In case of examples outside this area, like Case 1 in Section IV-B, DENFIS is likely to produce results with a higher error rate.

Further directions for research include: (1) improvement of the DENFIS model for a better online learning with self-modified parameter values; and (2) application of the DENFIS model for adaptive process control and mobile robot navigation.

Acknowledgment

The authors express thanks to the reviewers whose important comments and suggestions lead to a significant improvement of this paper.

References

[1] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Trans. ASME: J. Dyna. Syst., Measure., Control*, p. 220, 227, Sept. 1975.

[2] S. Amari and N. Kasabov, Eds., *Brain-Like Computing and Intelligent Information Systems*. New York: Springer Verlag, 1997.

[3] S. Amari, "Mathematical foundations of neuro-computing," *Proc. IEEE*, vol. 78, Sept. 1990.

[4] *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed., MIT Press, Cambridge, MA, 1995.

[5] J. C. Bezdek, *Pattern Recogntion with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[6] J. Bezdek, Ed., *Analysis of Fuzzy Information*. Boca Raton, FL: CRC Press, 1987, vol. 3.

[7] K. Bollacker, S. Lawrence, and L. Giles, "CiteSeer: An autonomous Web agent for automatic retrieval and identification of interesting publications," in *2nd Int. ACM Conf. Autonomous Agents*. New York, 1998, pp. 116–123.

[8] L. Bottu and V. Vapnik, "Local learning computation," *Neural Comput.*, vol. 4, pp. 888–900, 1992.

[9] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA: Holden Day, 1970.

[10] G. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*. Cambridge, MA: MIT Press, 1991.

[11] ——, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern-recognition architectures," *Neural Networks*, vol. 3, no. 2, pp. 129–152, 1990.

[12] G. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "FuzzyARTMAP: A neural network architecture for incremental supervised learning of analogue multi-dimensional maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 698–713, Oct. 1991.

[13] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, 1977.

[14] S. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intel. Fuzzy Syst.*, vol. 2, no. 3, Sept. 1994.

[15] R. S. Croder III, "Predicting the Mackey-glass timeseries with cascade—Correlation learning," in *Proc. 1990 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. Pittsburgh, PA, 1990, pp. 117–123.

[16] G. Cybenko, "Approximation by super-positions of sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, pp. 303–314, 1989.

[17] D. Deng and N. Kasabov, "Evolving self-organizing maps for online learning, data analysis and modeling," in *Proc. IJCNN'2000 Neural Networks Neural Computing: New Challenges Perspectives New Millennium*, vol. VI, S.-I. Amari, C. L. Giles, M. Gori, and V. Piuri, Eds. New York, 2000, pp. 3–8.

[18] H. DeGaris, "Circuits of production rule—GenNets—The genetic programming of nervous systems," in *Artificial Neural Networks and Genetic Algorithms*, R. Albrecht, C. Reeves, and N. Steele, Eds. New York: Springer Verlag, 1993.

[19] C. Fahlman and C. Lebiere, "The cascade—Correlation learning architecture," in *Advances in Neural Information Processing Systems*, D. Turetzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, vol. 2, pp. 524–532.

[20] J. D. Farmer and J. J. Sidorowitch, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59, p. 845, 1987.

[21] J. Freeman and D. Saad, "Online learning in radial basis function networks," *Neural Comput.*, vol. 9, no. 7, 1997.

[22] R. M. French, "Semi-destructive representations and catastrophic forgetting in connectionist networks," *Connection Sci.*, vol. 1, pp. 365–377, 1992.

[23] B. Fritzke, "A growing neural gas network learns topologies," *Adv. Neural Inform. Processing Syst.*, vol. 7, 1995.

[24] T. Fukuda, Y. Komata, and T. Arakawa, "Recurrent neural networks with self-adaptive GAs for biped locomotion robot," in *Proc. Int. Conf. Neural Networks*. New York, 1997.

[25] K. Funihashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[26] T. Gaussier and S. Zrehen, "A topological neural map for online learning: Emergence of obstacle avoidance in a mobile robot," *From Animals to Animats*, no. 3, pp. 282–290, 1994.

[27] R. Goodman, C. M. Higgins, J. W. Miller, and P. Smyth, "Rule-based neural networks for classification and probability estimation," *Neural Comput.*, vol. 14, pp. 781–804, 1992.

[28] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Upper Saddle River, NJ: Prentice-Hall, 1984.

[29] T. Hashiyama, T. Furuhashi, and Y. Uchikawa, "A decision making model using a fuzzy neural network," in *Proc. 2nd Int. Conf. Fuzzy Logic Neural Networks*, Iizuka, Japan, 1992, pp. 1057–1060.

[30] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *Adv. Neural Inform. Processing Syst.*, vol. 4, pp. 164–171, 1992.

[31] R. Hech-Nielsen, "Counter-propagation networks," in *IEEE 1st Int. Conf. Neural Networks*, vol. 2, San Diego, CA, 1987, pp. 19–31.
[32] T. M. Heskes and B. Kappen, "Online learning processes in artificial neural networks," in *Math. Found. Neural Networks*. Amsterdam, The Netherlands: Elsevier, 1993, pp. 199–233.
[33] T. C. Hsia, *System Identification: Least-Squares Methods*. Boston, MA: D.C. Heath, 1977.
[34] M. Ishikawa, "Structural learning with forgetting," *Neural Networks*, vol. 9, pp. 501–521, 1996.
[35] R. Jang, "ANFIS: Adaptive network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, May 1993.
[36] N. Kasabov, "Adaptable connectionist production systems," *Neurocomputing*, vol. 13, no. 2–4, pp. 95–117, 1996.
[37] ——, "Evolving fuzzy neural networks for online, adaptive, knowledge-based learning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 902–918, Dec. 2001.
[38] ——, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets Syst.*, vol. 82, no. 2, pp. 2–20, 1996.
[39] ——, "ECOS: A framework for evolving connectionist systems and the eco learning paradigm," in *Proc. ICONIP'98*. Kitakyushu, Japan, Oct. 1998, pp. 1222–1235.
[40] ——, "Evolving fuzzy neural networks—Algorithms, applications and biological motivation," in *Methodologies for the Conception, Design and Application of Soft Computing*, T. Yamakawa and G. Matsumoto, Eds, Singapore: World Scientific, 1998, pp. 271–274.
[41] ——, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press, 1996.
[42] N. Kasabov, J. S. Kim, M. Watts, and A. Gray, "FuNN/2—A fuzzy neural network architecture for adaptive learning and knowledge acquisition," *Inform. Sci.—Appl.*, vol. 101, no. 3–4, pp. 155–175, 1997.
[43] N. Kasabov and B. Woodford, "Rule insertion and rule extraction from evolving fuzzy neural networks: Algorithms and applications for building adaptive, intelligent expert systems," *Proc. FUZZ-IEEE*, Aug. 1999.
[44] N. Kasabov, "Adaptive Learning System and Method," PCT WO01/78003, April 20, 2000.
[45] S. Kawahara and T. Saito, "On a novel adaptive self-organizing network," *Cellular Neural Networks Appl.*, pp. 41–46, 1996.
[46] J. S. Kim and N. Kasabov, "HyFIS: Adaptive neuro-fuzzy systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, no. 9, pp. 1301–1321, 1999.
[47] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. N-9, pp. 1464–1497, 1990.
[48] ——, *Self-Organizing Maps*, 2nd ed. New York: Springer-Verlag, 1997.
[49] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," *Advances Neural Inform. Processing Syst.*, vol. 4, pp. 951–957, 1992.
[50] A. S. Lapedes and R. Farber, "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling," Los Alamos National Laboratory, Los Alamos, NM, vol. 87 545, Tech. Rep. LA-UR-87-2662, 1987.
[51] C. T. Lin and C. S. G. Lee, *Neuro Fuzzy Systems*: Prentice Hall, 1996.
[52] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. Mathematics, Statistics, Probability*, L. M. LeCam and J. N. Berkeley, Eds. Berkeley, CA, 1967, p. 281.
[53] M. Maeda, H. Miyajima, and S. Murashima, "A self organizing neural network with creating and deleting methods," *Nonlinear Theory Appl.*, vol. 1, pp. 397–400, 1996.
[54] J. Mandziuk and L. Shastri, "Incremental Class Learning Approach and Its Applications to Hand-Written Digit Recognition," International Computer Science Institute, CA, TR-98-015, 1998.
[55] M. T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[56] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.
[57] M. Mozer and P. Smolensky, "A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, vol. 2, pp. 598–605.
[58] *Neural Network Toolbox User's Guide*, vol. 5, The Math Works, Inc., Natick, MA, 1996, pp. 33–34.
[59] *Optimization Toolbox User's Guide*, vol. 3, The Math Works, Inc., Natick, MA, 1996, pp. 19–24.
[60] J. Platt, "A resource allocating network for function interpolation," *Neural Comp.*, vol. 3, pp. 213–225, 1991.
[61] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Oct. 1993.
[62] A. Robins and M. Frean, "Local learning algorithms for sequential learning tasks in neural networks," *J. Adv. Comput. Intell.*, vol. 2, no. 6, 1998.
[63] G. A. Rummery and M. Niranjan, "On-Line Q-Learning Using Connectionist Systems," Cambridge Univ. Engineering Dept., Cambridge, U.K., CUED/F-INENG/TR 166, 1994.
[64] *On-Line Learning in Neural Networks*, 1999.
[65] A. Sankar and R. J. Mammone, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. 42, pp. 291–299, Mar. 1993.
[66] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, 1985.
[67] M. Woldrige and N. Jennings, "Intelligent agents: Theory and practice," *Knowledge Eng. Rev.*, no. 10, 1995.
[68] T. Yamakawa, H. Kusanagi, E. Uchino, and T. Miki, "A new effective algorithm for neo fuzzy neuron model," in *Proc. Fifth IFSA World Congress*, 1993, pp. 1017–1020.
[69] L. A. Zadeh, "Fuzzy sets," *Inform. Control*, vol. 8, pp. 338–353, 1965.

Nikola K. Kasabov (M'94–SM'01) received the M.Sc. degree in computer science and the Ph.D. degree in mathematical sciences from the Technical University in Sofia, Bulgaria.

He is currently Professor and Chair, and Director of the Research Laboratory for Knowledge Engineering and Computational Intelligence in the Department of Information Science, University of Otago, Dunedin, New Zealand. He has published over 260 works, among them journal papers, conference papers, book chapters, text books, edited research books and monographs, edited conference proceedings, patents and authorship certificates in the area of intelligent systems, connectionist and hybrid connectionist systems, fuzzy systems, expert systems, speech recognition, and bioinformatics.

Dr. Kasabov is a Fellow of the Royal Society of New Zealand and the New Zealand Computer Society, Past President of the Asia Pacific Neural Network Assembly (APNNA), member of the TC12 group on Artificial Intelligence of IFIP, and also a member of INNS, NZRS, ENNS, and the IEEE Computer Society. He is the General Chairman of a series of biannual international conferences on Artificial Neural Networks and Expert Systems (ANNES).

Qun Song received the B.E. degree from the Department of Automation, East China Textiles Institute, the M.MMS. degree from the Division of Transportation Engineering, Tokyo University of Mercantile Marine, and the Ph.D. degree from the University of Otago, Dunedin, New Zealand, in 1982, 1995, and 2002, respectively.

He is currently a Research Fellow in the Department of Information Science, University of Otago, Dunedin, New Zealand. As an Automation Engineer, he worked in the Textile Scientific Research Academy of China from 1982 to 1991. From 1991 to 1993, he was a Foreign Researcher at Kansai University and Tokyo University, Japan. His current research interests include evolving neural-fuzzy systems and robotics.