IEEE Transactions on Fuzzy Systems (2002)

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

Nikola Kasabov, *Senior Member of IEEE,* and Qun Song

Department of Information Science

University of Otago, P.O Box 56, Dunedin, New Zealand

Phone: +64 3 479 8319, fax: +64 3 479 8311

nkasabov@otago.ac.nz, qsong@infoscience.otago.ac.nz

Abstract. This paper introduces a new type of fuzzy inference systems, denoted as DENFIS (dynamic evolving neural-fuzzy inference system), for adaptive on-line and off-line 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 on-line model; (2) creation of a first-order Takagi-Sugeno type fuzzy rule set, or an expanded high-order one, for a DENFIS off-line model. A set of fuzzy rules can be inserted into DENFIS before, or during its learning process. Fuzzy rules can also be extracted during the learning process or after it. An evolving clustering method (ECM), which is employed in both on-line and off-line 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.

IEEE Transactions on Fuzzy Systems (2002)

Key words: dynamic evolving neural-fuzzy inference system; on-line adaptive learning; on-line clustering; time series prediction; hybrid systems.

1. Introduction

The complexity and dynamics of real-world problems, especially in engineering and manufacturing, require sophisticated methods and tools for building on-line, adaptive intelligent systems (IS). 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 AI problems such as adaptive speech and image recognition, multi-modal information processing, adaptive prediction, adaptive on-line control, intelligent agents on the WWW [7, 67].

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

On-line learning is concerned with learning data as the system operates (usually in a real time) and the data might exist only for a short time. Several investigations [21, 22, 32, 62, 63, 64] proved that the most popular neural network models and algorithms that include multi-layer perceptrons (MLP) trained with the back propagation (BP) algorithm, radial basis function networks (RBF), and self-organising maps (SOM) are not suitable for adaptive, on-line learning.

At the same time several models that have elements of adaptive, on-line learning or structure and knowledge adaptation, have been developed, that include connectionist models [1, 2, 3, 4, 10, 11,

12, 16, 19, 21, 23, 25, 26, 30, 31, 33, 34, 45, 46, 47, 48, 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, 36, 37, 38, 41, 42, 44, 51, 55, 68], evolving fuzzy-neural networks [37, 39, 40, 44] and evolving self-organising maps [17].

The evolving connectionist systems framework (ECOS) [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. ECOS 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, 44] the EFuNN model is further developed mainly in respect of dynamic parameter self-optimisation. EFuNNs are fuzzy logic systems that have five-layer structures (Figure 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 (MF) 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. 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 co-ordinates of the center of the sphere in the fuzzy input space, and the $W2 (r_j)$ – the co-ordinates 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 normalised fuzzy difference between x_f and $W1(r_j)$ is smaller than the radius R_j, and the

local normalised 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 normalised 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 pre-defined 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 normalised fuzzy difference. In EFuNNs the local normalised 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 centres 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:

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

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

where: A2 is the activation vector of the fuzzy output neurons in the EFuNN structure when x is presented, $A1(r_j^{(1)}) = 1 - D(W1(r_j^{(1)}), x_f)$ 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: on-line; off-line; 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 EFuNN's dynamic features which make DENFIS suitable for on-line 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 organised as follows: Section 2 gives a description of an evolving clustering method (ECM) and its extension - evolving clustering method with constrained minimisation (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 3 introduces the DENFIS on-line model, and in section 4, DENFIS on-line model is applied to Mackay-Glass time series [13, 15] prediction; the results are compared with the results obtained with the use of resource-allocation network (RAN) [60], evolving fuzzy-neural network (EFuNN) [40] and evolving self-organising maps (ESOM) [17]. In section 5, two DENFIS off-line models are introduced, and in section 6, DENFIS off-line models are applied to Mackay-Glass 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 the final section.

The comparative analysis indicates clearly the advantages of DENFIS when used for both off-line, and especially on-line learning applications. In addition to this, the evolving clustering methods, ECM and ECMc, can perform well as on-line; or off-line, self-organised generic clustering model.

2. Evolving Clustering Method: ECM

Here, evolving, on-line, maximum distance-based clustering method, called *evolving clustering method* (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 on-line learning systems, and the second one is more suitable for off-line learning systems. The on-line evolving clustering method (ECM) is used in the DENFIS on-line model. The off-line evolving clustering method with constrained minimisation (ECMc) is an extension of the on-line mode. It takes the result from the on-line mode as initial values. An optimisation is then applied, that makes a pre-defined objective function based on a distance measure to reach a minimum value subject to given constraints.

2.1 On-line Evolving Clustering Method

Without any optimisation, the on-line evolving clustering method 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

centres in the input data space. It is a distance-based connectionist clustering method. With this method, cluster centres 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 centre, 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:

$$\| \boldsymbol{x} - \boldsymbol{y} \| = (\sum_{i=1}^{q} | x_i - y_i |^2)^{1/2} / q^{1/2} \quad (4)$$

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 centre, $\boldsymbol{Cc}$, 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 centres' 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*.

Figure 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 centre $\boldsymbol{Cc}_1^0$, and setting a value 0 for its cluster radius Ru_1 (Figure 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 centres $\boldsymbol{Cc}_j$, $D_{ij} = \| \boldsymbol{x}_i - \boldsymbol{Cc}_j \|$, $j = 1, 2, \ldots, n$, are calculated.
- Step 2: If there is any distance value, $D_{ij} = \| \boldsymbol{x}_i - \boldsymbol{Cc}_j \|$, equal to, or less than, at least one of the radiuses, 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

  $D_{im} = \| \boldsymbol{x}_i - \boldsymbol{Cc}_m \| = min (\| \boldsymbol{x}_i - \boldsymbol{Cc}_j \|)$ subject to the constraint $D_{ij} \leq Ru_j$,

  $j = 1, 2, \ldots, n$,

  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 Figure 2); the algorithm returns to Step 1. Else – go to the next step.
- Step 3: Find cluster C_a (with centre $\boldsymbol{Cc}_a$ and cluster radius Ru_a) from all n existing cluster centres through calculating the values $S_{ij} = D_{ij} + Ru_j$, $j = 1, 2, \ldots, n$, and then choosing the cluster centre $\boldsymbol{Cc}_a$ with the minimum value S_{ia}:

  $S_{ia} = D_{ia} + Ru_a = min(S_{ij}),\ j = 1, 2, \ldots, n.$
- 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 created in the same way as described in Step 0 (the cases of $\boldsymbol{x}_3$ and $\boldsymbol{x}_8$ in Figure 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 centre, $\boldsymbol{Cc}_a$, and increasing the value of its radius, Ru_a. The updated radius Ru_a^{new} is set to be equal to $S_{ia} / 2$ and the new centre $\boldsymbol{Cc}_a^{new}$ is located at the point on the line connecting the $\boldsymbol{x}_i$ and $\boldsymbol{Cc}_a$, and the distance from the new centre $\boldsymbol{Cc}_a^{new}$ to the point $\boldsymbol{x}_i$ is equal to Ru_a^{new} (the cases of $\boldsymbol{x}_2$, $\boldsymbol{x}_5$, $\boldsymbol{x}_7$ *and* $\boldsymbol{x}_9$ in Figure 2). The algorithm returns to Step 1.

In this way, the maximum distance from any cluster centre 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.

2.2 Constrained Optimisation and Off-line Evolving Clustering (ECMc)

The off-line evolving clustering method, called ECMc (evolving clustering method with constrained minimisation), applies an optimisation procedure to the resulted cluster centres 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, …, n, and finds a cluster centre in each cluster, to minimise 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 centre, $\boldsymbol{Cc}_j$, the objective function is defined by the following equation:

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

where $J_j = \sum_{\boldsymbol{x}_i \in C_j} \| \boldsymbol{x}_i - \boldsymbol{C}c_j \|$ is the objective function within cluster j;

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

and the constraints are defined by the next equation:

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

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 centres $\boldsymbol{Cc}_j$ are fixed, the minimising u_{ij} for Equations (5) and (6) is derived as follows:

if $\| \boldsymbol{x}_i - \boldsymbol{Cc}_j \| \leq \| \boldsymbol{x}_i - \boldsymbol{Cc}_k \|$, for each $j \neq k$;

$u_{ij} = 1$ else $u_{ij} = 0$. (7)

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

- Step1: Initialise the cluster centre $\boldsymbol{Cc}_j$, $j = 1, 2, \ldots, n$, that come from the result of ECM clustering process.
- Step2: Determine the membership matrix $\boldsymbol{U}$ by Equation (7).
- Step3: Employ the *constrained minimisation method* [59] with Equation (5) and (6) to get new cluster centres.
- Step4: Calculate the objective function J according to Equation (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 minimisation operations is over a certain value. Else, the algorithm returns to Step2.

2.3 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, evolving clustering method (on-line, one-pass)

(b) EFuNN, evloving fuzzy-neural network clustering (on-line, one pass) [40]

(c) SC, subtractive clustering (off-line, one pass)

(d) ECMc, evolving clustering with constrained minimisation (off-line)

(e) FCMC, fuzzy C-means clustering (off-line)

(f) KMC, K-means clustering (off-line)

Each of them partitions the data into *NoC* (= 15) clusters. The maximum distance, *MaxD*, between an example point and the corresponding cluster centre and the value of objective function *J*, defined by Equation (2), are taken as criteria for comparison. The results are shown in Table 1 and Figure 3. We can see from Table 1, that evolving clustering methods, both ECM and ECMc, can obtain the minimum value of *MaxD*, 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 *MaxD*, the ECM and the ECMc could achieve less number of partitions than the others.

3. DENFIS: A Dynamic Evolving Neural-Fuzzy Inference System

3.1 General principle

The dynamic evolving neural-fuzzy system, DENFIS, both on-line and off-line models, use Takagi-Sugeno type fuzzy inference engine [66]. Such inference engine used in DENFIS is composed of *m* fuzzy rules indicated as follows:

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

where "x_j is R_{ij}", $i = 1, 2, \ldots m$; $j = 1, 2, \ldots q$, are $m \times q$ fuzzy propositions as m antecedents form m fuzzy rules respectively; x_j, $j = 1, 2, \ldots, q$, are antecedent variables defined over universes of discourse X_j, $j = 1, 2, \ldots, q$, and R_{ij}, $i = 1, 2, \ldots m$; $j = 1, 2, \ldots, q$, are fuzzy sets defined by their fuzzy membership functions μ_{Rij}: $X_j \rightarrow [0, 1]$, $i = 1, 2, \ldots m$; $j = 1, 2, \ldots, q$. In the consequent parts, y is a consequent variable, and polynomial functions f_i, $i = 1, 2, \ldots m$, are employed,

In both DENFIS on-line and off-line models, all fuzzy membership functions are triangular type functions which depend on three parameters as given by the following equation.

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

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

If the consequent functions are crisp constants, i.e. $f_i(x_1, x_2, \ldots, x_q) = C_i$, $i = 1, 2, \ldots m$, 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 $f_i(x_1, x_2, \ldots, x_q)$, $i = 1, 2, \ldots m$, are

linear functions. If these functions are non-linear functions, it is called high-order Takagi-Sugeno fuzzy inference system.

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

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

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

3.2 Learning Processes in DENFIS On-line Model

In the DENFIS on-line 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:

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

For obtaining these functions a learning data set, which is composed of p data pairs $\{([x_{i1}, x_{i2}, \ldots, x_{iq}], y_i), i = 1, 2, \ldots, p\}$, is used and the least-square estimator (LSE) of $\boldsymbol{\beta} = [\beta_0\ \beta_1\ \beta_2\ \ldots\ \beta_q]^T$, are calculated as the coefficients $\boldsymbol{b} = [b_0\ b_1\ b_2\ \ldots\ b_q]^T$, by applying the following formula:

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

where

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

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

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

$$\boldsymbol{b}_{\mathrm{w}} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \boldsymbol{y}, \tag{10}$$

where

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

and w_j is the distance between j-th example and the corresponding cluster centre, $j = 1, 2, \dots p$.

We can rewrite the equations(9) and (10) as follows:

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

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

Let the k-th row vector of matrix A defined in Equation (9) be $a_k^T = [1\ x_{k1}\ \ x_{k2}\ \dots\ x_{kq}]$ and the k-th element of $\boldsymbol{y}$ be y_k, then $\boldsymbol{b}$ can be calculated iteratively as follows:

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

Here, the initial values of $\mathbf{P}_n$ and $\boldsymbol{b}_n$ can be calculated directly from Equation (12) with the use of the first n data pairs from the learning data set.

The Equation (13) is the formula of recursive LSE [28]. In the DENFIS on-line model, we use a weighted recursive LSE with forgetting factor defined as the following equation.

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

where w is the weight defined in Equation (10) and λ is a forgetting factor which typical value is between 0.8 and 1.

In the on-line DENFIS model, the rules are created and updated at the same time with the input space partitioning using on-line evolving clustering method (ECM) and Equation (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 centres.

(3) For every cluster centre $\boldsymbol{C}_i$, find p_i data points whose positions in the input space are closest to the centre, $i = 1, 2, \ldots, m$.

(4) For obtaining a fuzzy rule corresponding to a cluster centre, create the antecedents of the fuzzy rule using the position of the cluster centre and Equation(8). Using Equation (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 centre are taken as the weights in Equation(12).

In the above steps, m, n_0 and p are the parameters of the DENFIS on-line 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 centre is found by the ECM. The antecedent of the new fuzzy rule is formed by using Equation (8) with the position of the cluster centre (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 Equation(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 Equation (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 Equation(8).

3.3 Takagi-Sugeno Fuzzy Inference In DENFIS

The Takagi-Sugeno fuzzy inference system utilised in DENFIS is a dynamic inference. In addition to dynamically creating and updating fuzzy rules the DENFIS on-line 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 case of two input vectors that are very close to each other, especially in the DENFIS off-line model, the inference system may have the same fuzzy rule inference group. In the DENFIS on-line 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 Figure 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 2D input space as shown in fig.4a and fig.4b 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.4a group, but for the group of rules from fig.4b 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.

4 Time Series Modelling and Prediction with the DENFIS On-line Model

In this section the DENFIS on-line model will be applied to modelling and predicting the future values of a chaotic time series - the Mackey-Glass (MG) data set [13], which has been used as a bench-mark 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 below:

$$\frac{d\,x(t)}{d\,t} = \frac{0.2\,x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (15)$$

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 on-line learning models applied on the same task. These models are Neural gas [23], resource-allocating network (RAN) [60], evolving self-organising maps (ESOM) [17] and evolving fuzzy-neural network (EFuNN) [40]. Here, we take the non-dimensional error index (NDEI) [15] which is defined as the root mean square error (RMSE) divided by the standard deviation of the target series.

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 mentioned above on-line models the learning data is used for the on-line 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 utilised where we first obtained a group of fuzzy rules from the first half of training data (1500 samples), using the DENFIS off-line model I (it will be introduced in next section); then we inserted these rules to

the DENFIS on-line 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 2 lists the prediction results (NDEI on test data after on-line learning) and the number of rules (nodes, units) evolved (used) in each model.

Figure 5 (a), (b) and (c) display the testing errors (from the recall processes on the test data) of DENFIS on-line model with different number of fuzzy rules:

(a) DENFIS on-line model with 58 fuzzy rules;

(b) DENFIS on-line model with 883 fuzzy rules (different parameter values are used from those in the model above);

(c) DENFIS on-line model with 883 fuzzy rules that is evolved after an initial set of rules was inserted.

5 DENFIS Model for Off-line Learning

The DENFIS on-line model presented in the previous section, can be used also for off-line, 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 on-line model is extended here to work efficiently in an off-line, batch training mode. Two DENFIS models for off-line 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 on-line model, is employed in model 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) multi-layer perceptrons to realise the function f in the consequent part of each fuzzy rule instead of using a pre-defined function.

The DENFIS off-line learning process is implemented in following way:

- cluster (partition) the input space to find n cluster centres (n rule nodes, n rules) by using the off-line evolving clustering method with constrained optimisation (ECMc),
- create the antecedent part for each fuzzy rule using Equation(8) and the current position of the cluster centre (rule node),
- find n data sets each of them including one cluster centre and p learning data pairs that are closest to the centre 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 Equation(10) or Equation(12) with n data sets; the distances between each data point and their corresponding centre 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 back propagation algorithm (BP).

6 Time Series Modelling and Prediction with the DENFIS Off-line Model

Dynamic time-series modelling 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 off-line models for the same task as in section 4. For comparison purposes two other well-known models, adaptive neural-fuzzy inference system (ANFIS) [35], and a multilayer perceptron

trained with the back propagation algorithm (MLP-BP) [58], are also used for this task under the same conditions.

In addition to the NDEI, in the case of off-line 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 3 lists the off-line 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.

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

(d) DENFIS off-line mode I with 116 fuzzy rules;

(e) DENFIS off-line mode I with 883 fuzzy rules;

(f) DENFIS off-line mode II with 58 fuzzy rules.

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

7. Conclusions and directions for further research

This paper presents the principles of a fuzzy inference system, DENFIS, for building both on-line and off-line knowledge-based, adaptive learning systems. Both DENFIS on-line and off-line 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 on-line learning, and with ANFIS [35] and MLP [38] in the case of off-line learning.

DENFIS uses a local generalisation, like EFuNN and CMAC neural networks [1], so it needs more training data than the models which use global generalisation 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 4.2, 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 on-line learning with self-modified parameter values; (2) application of the DENFIS model for adaptive process control and mobile robot navigation.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, contract UOOX0016. The authors express thanks to the reviewers whose important comments and suggestions lead to a significant improvement of the paper.

References

1. Albus, J.S., A new approach to manipulator control: The cerebellar model articulation controller (CMAC), Trans. of the ASME: Journal of Dynamic Systems, Measurement, and Control, vol.27, pp.220:227, Sept. (1975)
2. Amari, S. and Kasabov, N. eds, "Brain-like Computing and Intelligent Information Systems", Springer Verlag,1997.
3. Amari, S., Mathematical foundations of neuro-computing, Proc. of IEEE, 78 (9), 1443-1463, Sept. (1990)
4. Arbib, M. (ed) The Handbook of Brain Theory and Neural Networks,The MIT Press, 1995.
5. Bezdek, J.C., "Pattern Recognition with Fuzzy Objective Function Algorithms", Plenum Press, New York, 1981.
6. Bezdek, J (ed). Analysis of Fuzzy Information, 3 vols. Boca Raton, Fla, CRC Press, 1987.
7. Bollacker, K., S.Lawrence and L.Giles, CiteSeer: An autonomous Web agent for automatic retrieval and identification of interesting publications, 2nd International ACM conference on autonomous agents, ACM Press, 1998, 116-123
8. Bottu, L., and Vapnik, V. "Local learning computation", Neural Computation, vol. 4, No.6, 888-900 (1992)
9. Box, G.E.P., and Jenkins, G.M., "Time series analysis, forecasting and control", Holden Day, San Francisco, 1970.
10. Carpenter, G. and Grossberg S., Pattern recognition by self-organizing neural networks, The MIT Press, Cambridge, Massachusetts (1991)

11. Carpenter, G. and S. Grossberg, "ART3: Hierarchical search using chemical transmitters in self-organising pattern-recognition architectures", Neural Networks, 3(2) 129-152(1990).
12. Carpenter, G. S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, "FuzzyARTMAP: A neural network architecture for incremental supervised learning of analogue multi-dimensional maps," IEEE Transactions of Neural Networks , vol.3, No.5, 698-713 (1991).
13. Mackey, M. C., Glass, L., "Oscillation and Chaos in Physiological Control systems", Science, 197:287-289, 1977.
14. Chiu, S., "Fuzzy Model Identification Based on Cluster Estimation", Journal of Intelligent & Fuzzy System, Vol. 2, No. 3, Step. 1994.
15. Croder, III R. S., " Predicting the Mackey-Glass Timeseries with Cascade- correlation Learning", in: D. Touretzky, G. Hinton and T. Sejnowski eds., Proc. of the 1990 Connectionist Models Summer School, 117-123, Carnegic Mellon University, 1990.
16. Cybenko, G., Approximation by super-positions of sigmoidal function, Mathematics of Control, Signals and Systems, 2, 303-314 (1989)
17. Deng, D., Kasabov, N., "Evolving Self-Organizing Maps for On-line Learning, Data Analysis and Modelling" Shun-Ichi Amari, C. Lee Giles, Marco Gori, Vincenzo Piuri (eds) Proceedings of the IJCNN'2000 on Neural Networks Neural Computing: New Challenges and Perspectives for the New Millennium, IEEE Press, Vol VI, 3-8 (2000)
18. DeGaris, H., "Circuits of Production Rule - GenNets – The genetic programming of nervous systems", in: Albrecht, R., Reeves, C. and Steele, N. (eds) Artificial Neural Networks and Genetic Algorithms, Springer Verlag (1993)

19. Fahlman, C., and C. Lebiere, "The Cascade- Correlation Learning Architecture", in: Turetzky, D (ed) Advances in Neural Information Processing Systems, vol.2, Morgan Kaufmann, 524-532 (1990).
20. Farmer, J.D., and Sidorowitch, J.J. Predicting chaotic time series, Physical Review Letters, vol.59, No.8, 845-848 (1987)
21. Freeman, J., D. Saad, “On-line learning in radial basis function networks”, Neural Computation vol. 9, No.7 (1997), 1601-1622.
22. French, R.M., “Semi-destructive representations and catastrophic forgetting in connectionist networks, Connection Science, vol.4,No.3/4, 365-377 (1992)
23. Fritzke, B. “A growing neural gas network learns topologies”, Advances in Neural Information Processing Systems, No.7 (1995), 625-632.
24. Fukuda, T., Y. Komata, and T. Arakawa, "Recurrent Neural Networks with Self-Adaptive GAs for Biped Locomotion Robot", In: Proceedings of the International Conference on Neural Networks ICNN'97, IEEE Press (1997)
25. Funihashi, K., On the approximate realization of continuous mappings by neural networks, Neural Networks, 2, 183-192 (1989)
26. Gaussier, T., and S. Zrehen, “A topological neural map for on-line learning: Emergence of obstacle avoidance in a mobile robot”, In: From Animals to Animats No.3, 282-290, (1994).
27. Goodman, R., C.M. Higgins, J.W. Miller, P.Smyth, "Rule-based neural networks for classification and probability estimation", Neural Computation, Vol.4, No.6, 781-804 (1992).
28. Goodwin, G. C., Sin, K. S., “Adaptive Filtering Prediction and Control”, Prentice-Hall, Englewood Cliffs, N. J., 1984

29. Hashiyama, T., T. Furuhashi, Y Uchikawa,. “A Decision Making Model Using a Fuzzy Neural Network”, in: Proceedings of the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka, Japan, 1057-1060, (1992).
30. Hassibi,B., and Stork, D.G.“Second order derivatives for network pruning: Optimal Brain Surgeon,” in: Advances in Neural Information Processing Systems, 4, 164-171, (1992).
31. Hech-Nielsen, R. “Counter-propagation networks”, IEEE First int. conference on neural networks, San Diego, vol.2, pp.19-31 (1987)
32. Heskes, T.M., B. Kappen, “On-line learning processes in artificial neural networks”, in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233, (1993).
33. Hsia, T. C., “System Identification: Least-Squares Methods”, D.C. Heath and Company, 1977.
34. Ishikawa, M., "Structural Learning with Forgetting", Neural Networks 9, 501-521, (1996).
35. Jang, R., “ANFIS: adaptive network-based fuzzy inference system”, IEEE Trans. on Systems, Man, and Cybernetics, 23(3), May-June, 665-685, (1993).
36. Kasabov, N. "Adaptable connectionist production systems”. Neurocomputing, 13 (2-4) 95-117, (1996).
37. Kasabov, N. “Evolving Fuzzy Neural Networks for On-line, Adaptive, Knowledge-based Learning”, IEEE Transactions of Systems, Man, and Cybernetics, B – Cybernetics, No.13, December, 2001
38. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems, 82 (2) 2-20 (1996).
39. Kasabov, N., “ECOS: A framework for evolving connectionist systems and the eco learning paradigm”, Proc. of ICONIP'98, Kitakyushu, Oct. 1998, IOS Press, 1222-1235.

40. Kasabov, N., "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in: Yamakawa, T. and G.Matsumoto (eds) Methodologies for the conception, design and application of soft computing, World Scientific, 1998, 271-274
41. Kasabov, N., Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA, (1996).
42. Kasabov, N., J. S Kim, M. Watts, A. Gray, "FuNN/2- A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition", Information Sciences - Applications, 101(3-4): 155-175 (1997)
43. Kasabov, N., Woodford, B. Rule Insertion and Rule Extraction from Evolving Fuzzy Neural Networks: Algorithms and Applications for Building Adaptive, Intelligent Expert Systems, in Proc. of FUZZ-IEEE, Seoul, August 1999 (1999)
44. Kasabov, N., Adaptive learning method and system, Patent reg.No503882, University of Otago, New Zealand (2000)
45. Kawahara, S., Saito, T. "An adaptive self-organising algorithm with virtual connection", in Progress in Connectionist-based inf. Systems , N.Kasabov et al. (ed) Springer, vol.1,1997,338-341
46. Kim, and N. Kasabov, HyFIS: Adaptive neuro-fuzzy systems and their application to non-linear dynamical systems, *Neural Networks*, v.12 (9), pp.1301-1321 (1999)
47. Kohonen, T., "The Self-Organizing Map", Proceedings of the IEEE, vol.78, N-9, pp.1464-1497, (1990)
48. Kohonen, T., Self-Organizing Maps, second edition, Springer Verlag, 1997

49. Krogh, A., and J.A. Hertz, "A simple weight decay can improve generalisation", Advances in Neural Information Processing Systems, 4 951-957, (1992).

50. Lapedes, A.S. and R. Farber. Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, 87545, 1987.

51. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).

52. MacQueen, J., "Some Methods for Classification and Analysis of Multi-variate Observations", in: Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probability, eds. L.M. LeCam and J. Neyman Berkeley: U. California Press, 281 (1967).

53. Maeda, M., Miyajima, H. and Murashima, S., "A self organizing neural network with creating and deleting methods, Nonlinear theory and its applications, 1, 397-400 (1996)

54. Mandziuk, J., Shastri, L. Incremental class learning approach and its applications to hand-written digit recognition, TR-98-015, International Computer Science Institute, California (1998)

55. Mitchell, M.T., "Machine Learning", MacGraw-Hill (1997)

56. Moody, J., Darken, C., Fast learning in networks of locally-tuned processing units, Neural Computation, 1, 281-294 (1989)

57. Mozer, M., and P. Smolensky, "A technique for trimming the fat from a network via relevance assessment", in: D. Touretzky (ed) Advances in Neural Information Processing Systems, vol.2, Morgan Kaufmann, 598-605 (1989).

58. Neural Network Toolbox user's guide, The MATH WORKS Inc., 5: 33-34, 1996

59. Optimization Toolbox User's Guide, The MATH WORKS Inc., 3: 19-24, 1996.

60. Platt, J., "A Resource Allocating Network for Function Interpolation", Neural Comp., 3, 213-225, 1991.

61. Reed, R., "Pruning algorithms - a survey", IEEE Trans. Neural Networks, 4 (5) 740-747, (1993).

62. Robins, A. and Frean, M. "Local learning algorithms for sequential learning tasks in neural networks, Journal of Advanced Computational Intelligence, vol.2, 6 (1998)

63. Rummery, G.A., and M. Niranjan, "On-line Q-learning using connectionist systems", Cambridge University Engineering Department, CUED/F-INENG/TR 166 (1994)

64. Saad, D. (ed) On-line learning in neural networks, Cambridge University Press, 1999

65. Sankar, A., and R.J. Mammone, "Growing and Pruning Neural Tree Networks", IEEE Trans. Comput. 42(3) 291-299 (1993).

66. Takagi, T. and Sugeno, M., Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Systems, Man, and Cybernetics, 15: 116-132, 1985.

67. Woldrige, M., and N. Jennings, "Intelligent agents: Theory and practice", The Knowledge Engineering review, vol.10, No.2, 1995, 115-152

68. Yamakawa, T., H. Kusanagi, E. Uchino and T. Miki, "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017-1020, (1993).

69. Zadeh, L.A. Fuzzy sets. Information and control, 8: 338-353, 1965

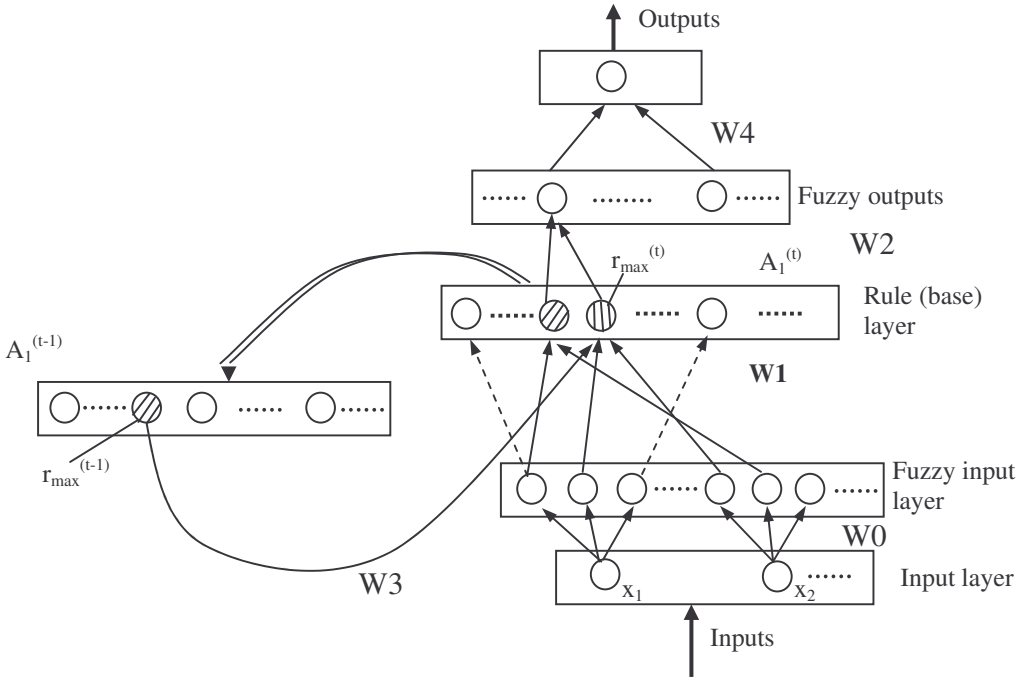

Figure 1. The structure of EFuNN

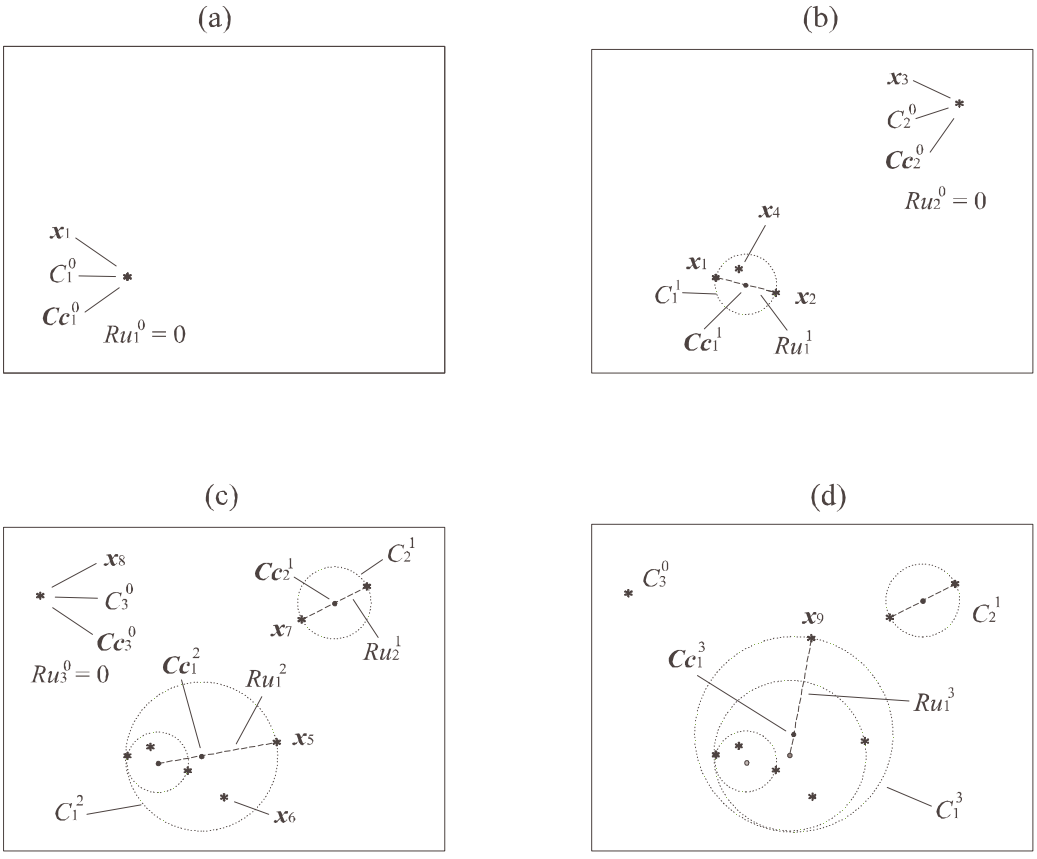

Figure 2. A brief clustering process using ECM with samples x_1 to x_9 in a 2-D space:

(a) The example x_1 causes the ECM to create a new cluster C_1^0

(b) x_2 : update cluster $C_1^0 \rightarrow C_1^1$

x_3 : create a new cluster C_2^0

x_4 : do nothing

(c) x_5 : update cluster $C_1^1 \rightarrow C_1^2$

x_6 : do nothing

x_7 : update cluster $C_2^0 \rightarrow C_2^1$

x_8 : create a new cluster C_3^0

(d) x_9 : update cluster $C_1^2 \rightarrow C_1^3$

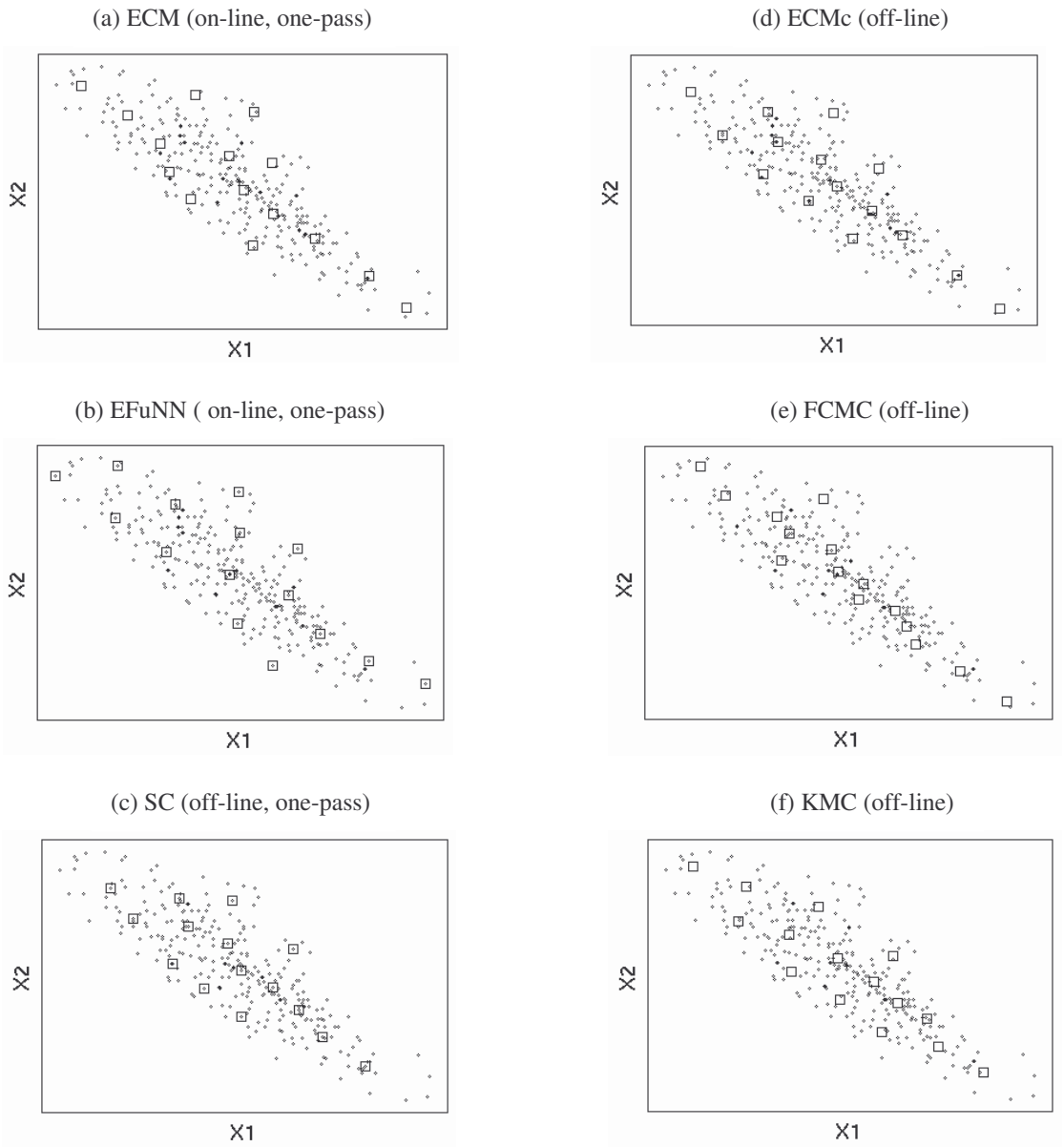

Figure 3. Results of clustering the gas-furnace data set by several clustering methods.

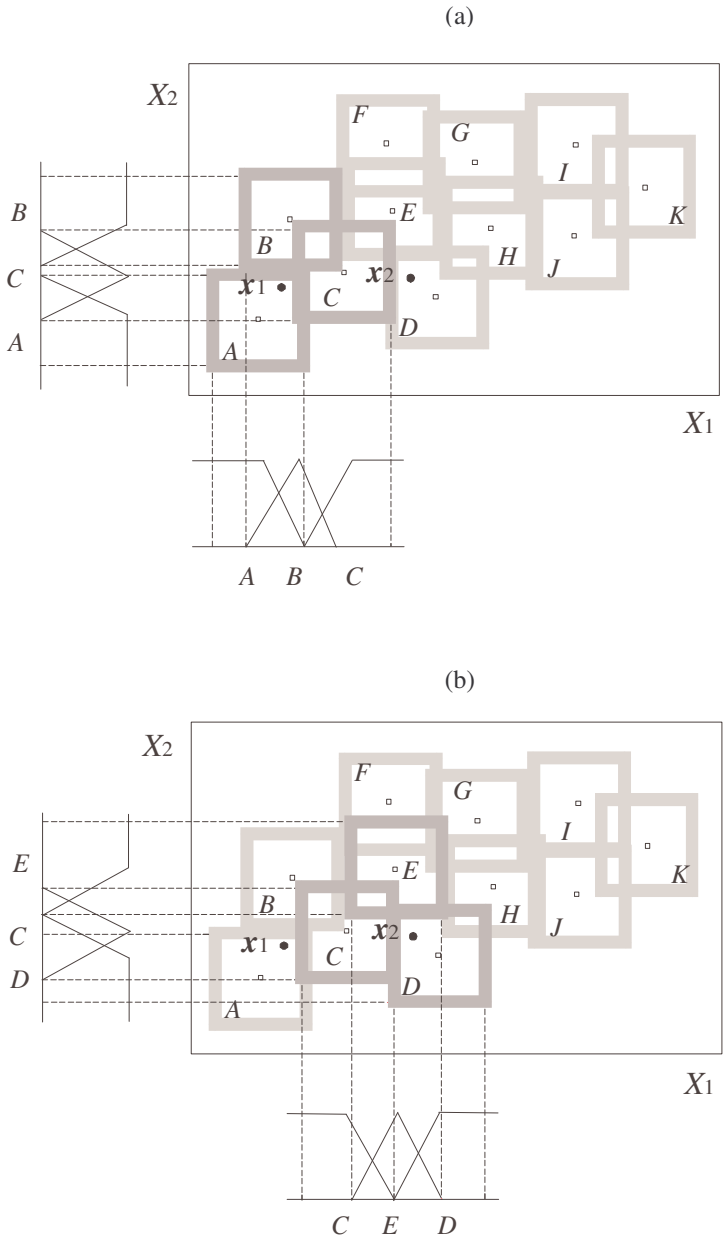

Figure 4. Two fuzzy rule groups are formed by DENFIS to perform inference for an input vector $\boldsymbol{x}_1$ (a) and for an input vector $\boldsymbol{x}_2$ (b) that is entered at a later time moment, all represented in the 2-D space of the first two input variables X1 and X2.

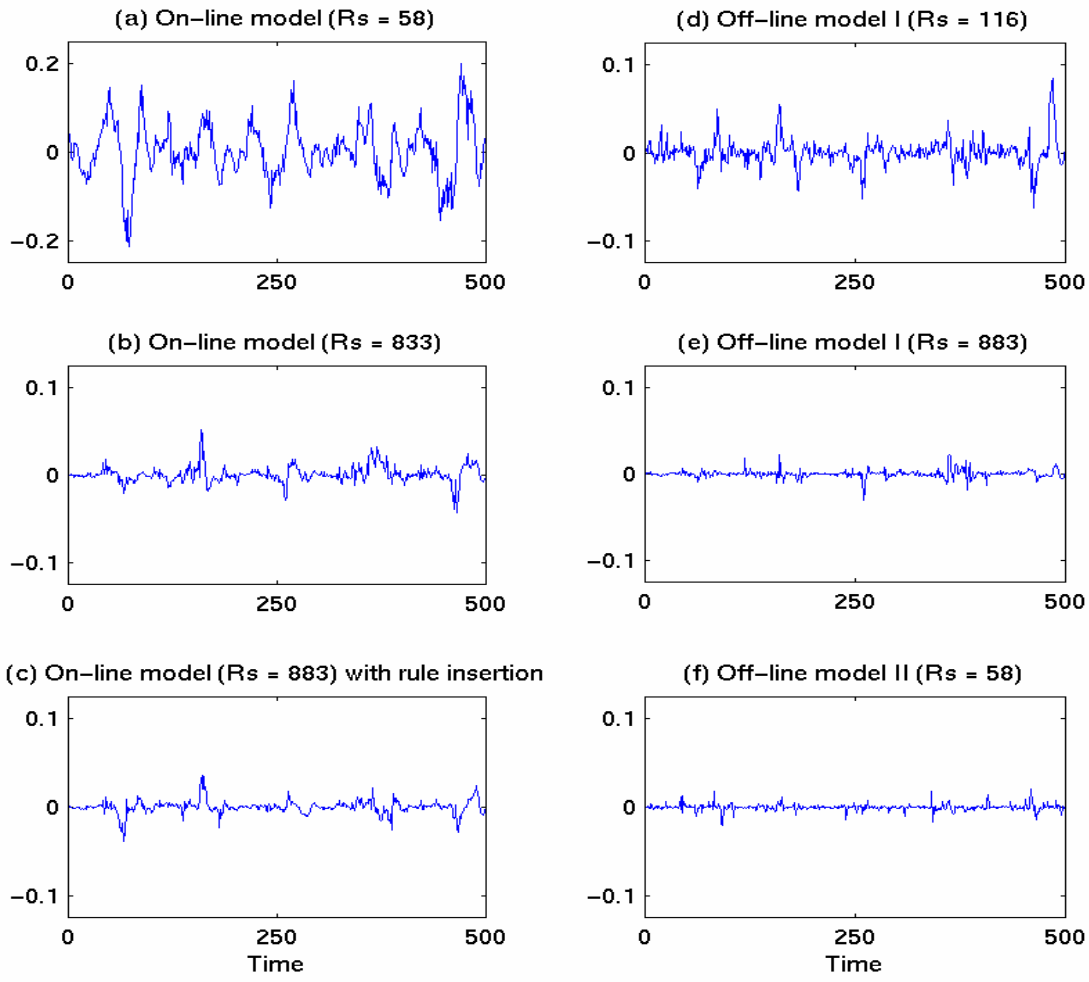

Figure 5. Prediction error of DENFIS on-line (a,b,c) and off line (d,e,f) models on test data taken from the Mackey-Glass time series

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

Table 1. Results obtained by using different clustering methods for clustering the gas-furnace data set into 15 clusters

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

Table 2. Prediction results of on-line learning models on the Mackey-Glass 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

Table 3. Prediction results of off-line learning models on Mackey-Glass training and test data