Evolving Fuzzy Neural Networks for Supervised/Unsupervised On-line Knowledge-Based Learning

Article *in* IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society · January 2002

DOI: 10.1109/3477.969494 · Source: IEEE Xplore

1 author:

Nikola Kirilov Kasabov
Auckland University of Technology
815 PUBLICATIONS **17,991** CITATIONS

IEEE Transactions of Systems, Man and Cybernetics, Part B – Cybernetics (2001), vol31,No.6, December,2001

Evolving Fuzzy Neural Networks for Supervised/Unsupervised On-line, Knowledge-Based Learning

Nikola Kasabov, *Senior Member of IEEE*

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

Abstract. The paper introduces evolving fuzzy neural networks (EFuNNs) as a means for the implementation of the evolving connectionist systems (ECOS) paradigm that is aimed at building on-line, adaptive intelligent systems that have both their structure and functionality evolving in time. EFuNNs evolve their structure and parameter values through incremental, hybrid supervised/unsupervised, on-line learning. They can accommodate new input data, including new features, new classes, etc. through local element tuning. New connections and new neurons are created during the operation of the system. EFuNNs can learn spatial-temporal sequences in an adaptive way through one pass learning, and automatically adapt their parameter values as they operate. Fuzzy or crisp rules can be inserted and extracted at any time of the EFuNN operation. The characteristics of EFuNNs are illustrated on several case study data sets for time series prediction and spoken word classification. Their performance is compared with traditional connectionist methods and systems. The applicability of EFuNNs as general purpose on-line learning machines is discussed what concerns systems that learn from large databases, life-long learning systems, on-line adaptive systems in different areas of Engineering.

Key words: evolving connectionist systems; evolving fuzzy neural networks; on-line learning; knowledge-based neural networks; sleep learning.

1. Introduction

The complexity and the dynamics of real-world problems, such as adaptive speech recognition and language acquisition [21,34,41], adaptive intelligent prediction and control systems [1], intelligent agent-based systems and adaptive agents on the Web [81], mobile robots [20], visual monitoring systems and multi-modal information processing [37,54], large Bio-informatics data processing, and many more [2,4], require sophisticated methods and tools for building on-line, adaptive, knowledge-based intelligent systems (IS). Such systems should be able to: (1) *learn fast* from a large amount of data (using fast training); (2) *adapt incrementally* in an on-line mode; (3) dynamically create new modules – have open structure; (4) *memorise information* that can be used at a later stage; (5) *interact continuously* with the environment in a "life-long" learning mode; (6) deal with *knowledge* (e.g. rules), as well as with data; (7) adequately represent *space and time* [2,5,35,36,38,61,66,71]. Developing a computational model called evolving fuzzy neural network (EFuNN) that meets the seven requirements above is the objective of the current paper.

Several methods and systems have been developed so far that meet some of the criteria above and that have influenced the development of EFuNNs. These are methods and systems for: *adaptive learning* [4,5,7,8,14,30,46,47,48]; i*ncremental learning* [6,7,8,9,19,53,58,61,71]; *lifelong learning* [69,35,36,82]; *on-line learning* [17,21,22,28,31,35,36,42,44,61,66,67,69]; *constructivist structural learning* [15,19,11,14,9] that is supported by biological facts [14,62,73,77,82]; s*electivist structural learning* [26,29,49,56,59,64,50,32]; hybrid constructivist/selectivist structural learning

[52,66,70,31]; k*nowledge-based learning neural networks (KBNN)* [57,24,25,30,33,38,44, 45,51,63,76,77,83].

The EFuNN model presented in the paper has elements from all the groups above. The model is called evolving because of the nature of the structural growth and structural adaptation of the whole evolving connectionist system (ECOS) it is part of. In terms of on-line neuron allocation, the EFuNN model is similar to the Resource Allocating Network (RAN) suggested by Platt [61] and improved in other related models [21,66]. The RAN model allocates a new neuron for a new input example (x,y) if the input vector x is not close in the input space to any of the already allocated radial basis neurons (centers), and also – if the output error evaluation (y-y'), where y' is the produced by the system output for the input vector x, is above an error threshold. Otherwise, centers will be adapted to minimise the error for the example (x,y) through a gradient descent algorithm. In terms of adaptive optimisation of many individual linear units, EFuNN is close to the Receptive Field Weight regression (RFWR) model by Schaal and Atkeson [71]. EFuNNs have also similarities with the Fritzke's Growing Cell Structures and Growing Neural Gas models [19], and with other dynamic radial basis function networks (RBFN) [58,5,78] and with the counter-propagation networks [27] in terms of separating the unsupervised learning, which is performed first, from the supervised learning, applied next, in a two-tyre structure. Creating new nodes is a feature also of SCONN [12] and VC network [84]. The EFuNN learning algorithm differs from the above in many aspects, mainly in the local element tuning, in the employment of simpler and faster learning modes, in more flexibility when evolving internal structures and representations, and in the knowledge-based orientation. A comparative analysis between EFuNNs and other similar models on benchmark problems shows that while EFuNNs are comparable with the other methods in terms of accuracy of the obtained results, they are much faster, more

controllable, and evolve meaningful internal representations. EFuNNs suggest a new neuro-fuzzy systemic approach that employs more sophisticated supervised/unsupervised, knowledge-based learning methods. The functionality of EFuNNs can be fully utilised when EFuNNs are used as elements of an ECOS framework for adaptive, intelligent, knowledge-based systems.

2. The ECOS framework

Evolving connectionist systems (ECOS) are systems that evolve their structure and functionality over time through interaction with the environment – fig.1 [35]. They have some ("genetically") pre-defined parameters (knowledge) but they also learn and adapt as they operate. They emerge, evolve, develop, unfold through learning, and through changing their structure in order to better represent incoming data.

[Figure 1]

A block diagram of the ECOS framework is given in fig.2. ECOS are multi-level, multi-modular structures where many neural network modules (denoted as NNM) are connected with inter-, and intra- connections.

[Figure 2]

The main blocks of ECOS are described below.

Feature selection part. It performs filtering of the input information, feature extraction and forming the input vectors.

Representation (memory) part, where information (patterns) are stored. It is a multi-modular, evolving structure of NNMs organised in groups. This is the most important part of ECOS. One realisation of a NNM is the EFuNN, presented in the next section.

Higher-level decision part. It consists of modules that receive inputs from the representation part and also feedback from the environment.

(1) Action part. These are modules that take input values from the decision part and pass output information to the environment.

Knowledge-based part. This part extracts compressed abstract information from the representation modules and from the decision modules in different forms of rules, abstract associations, etc. This part requires that the NNM should operate in a knowledge-based learning mode and provide with the knowledge about the problem under consideration.

Adaptation part. This part uses statistical, evolutionary (e.g. genetic algorithms [23,79]), and other techniques to evaluate and optimise the parameters of the ECOS during its operation.

The ECOS operation principles correspond to the seven requirements to the intelligent systems presented in section 1 [35]. They are also based on some biological facts and biological principles (see for example [55,62,68,72,82]). Implementing NNMs of the ECOS framework require connectionist models that support these principles. Such model is the evolving fuzzy neural network (EFuNN).

3. The Evolving Fuzzy Neural Network (EFuNN) Model

3.1. General principles of EFuNNs

Fuzzy neural networks are connectionist structures that implement fuzzy rules and fuzzy inference [25,51,63,83,38]. FuNNs represent a class of them [38,33,39,40]. The EFuNN model presented here is principally different from all the fuzzy neural network models introduced so far despite some structural similarities. EFuNNs evolve according to the ECOS principles. A brief first introduction of EFuNN was given in [36]. Here the EFuNN architecture and functionality are further developed, illustrated and analysed in details.

EFuNNs have a five-layer structure, similar to the structure of FuNNs (fig.3a). But here 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 (see fig.3b). The layer of feedback connections could be used if temporal relationships of input data are to be memorised structurally.

[Figure 3 a,b]

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 for a particular input variable. Different membership functions (MF) can be attached to these neurons (triangular –fig.4, Gaussian, etc.).

[Figure 4]

The number and the type of MF can be dynamically modified which is explained in sub-section 3.4. 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, or a Gaussian 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 values of the output variables. Here a linear activation function is used to calculate the defuzzified values for the output variables.

A partial case of EFuNN would be a three layer network without the fuzzy input and the fuzzy output layers. In this case a slightly modified versions of the algorithms described below is applied, mainly in terms of measuring Euclidean distance and using Gaussian activation functions.

The evolving process is based on either of the two assumptions: (1) no rule nodes exist prior to learning and all of them are created (generated) during the evolving process; or (2) there is an initial set of rule nodes that are not connected to the input and output nodes and become connected through the learning (evolving) process. The latter case is more biologically plausible [82]. The EFuNN evolving algorithm presented in the next section does not make a difference between these two assumptions.

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 (see fig.5a), 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 the example 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

normalised output error Err= ||y – y'|| / Nout is smaller than an error threshold E, Nout is the number of the outputs and y' is the produced by EFuNN output vector. The error parameter E sets 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 (see fig.4). 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. In RBF networks Gaussian radial basis functions are allocated to the nodes and used as activation functions to calculate the distance between the node and the input vector across the whole input space.

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_j, and in the fuzzy output space depending on the output error trough the Widrow-Hoff LMS algorithm (delta algorithm) [80], as it is shown in fig.5a. This adjustment can be represented mathematically by the change in the connection weights of the rule node r_j from $W1(r_j^{(t)})$ and $W2(r_j^{(t)})$ to $W1(r_j^{(t+1)})$ and $W2(r_j^{(t+1)})$ respectively according to the following vector operations:

$$W1(r_j^{(t+1)}) = W1(r_j^{(t)}) + l_{j,1} \cdot (W1(r_j^{(t)}) - x_f) \quad (2)$$

$$W2(r_j^{(t+1)}) = W2(r_j^{(t)}) + l_{j,2} \cdot (A2 - y_f) \cdot A1(r_j^{(t)}) \quad (2')$$

where: $A2 = f_2(W2.A1)$ is the activation vector of the fuzzy output neurons in the EFuNN structure when x is presented; $A1(r_j^{(t)}) = f_1(D(W1(r_j^{(t)}), x_f))$ is the activation of the rule node $r_j^{(t)}$; a simple linear function can be used for f_1 and f_2, e.g. $A1(r_j^{(t)}) = 1 - D(W1(r_j^{(t)}), x_f))$; $l_{j,1}$ and $l_{j,2}$ are the current learning rates of rule node r_j for its input layer and its output layer of connections respectively; further in the paper we will assume that the two learning rates have the same value calculated as $l_j = 1/ Nex(r_j)$, where $Nex(r_j)$ is the number of examples currently associated with rule node r_j. The statistical rationale behind this is that the more examples are associated with a rule node the less it will "move" in the input space when a new example has to be accommodated by this rule node.

When a new example is associated with a rule node r_j not only its location in the input space, but also its receptive field expressed as its radius Rj, and its sensitivity threshold Sj change as follows:

$$Rj^{(t+1)} = Rj^{(t)} + D(W1(r_j^{(t+1)}), W1(r_j^{(t)})), \text{ respectively} \quad (3)$$

$$Sj^{(t+1)} = Sj^{(t)} - D(W1(r_j^{(t+1)}), W1(r_j^{(t)}))$$

The learning process in the fuzzy input space is illustrated in fig.5b on four data points d_1, d_2, d_3 and d_4. Fig.5b shows how the centre $r_j^{(1)}$ of the rule node r_j adjusts (after learning each new data point) to its new positions $r_j^{(2)}$, $r_j^{(3)}$, $r_j^{(4)}$ when one pass learning is applied. Fig.5c shows how the rule node position would move to new positions $r_j^{(2(2)}$, $r_j^{(3(2)}$, and $r_j^{(4(2)}$, if another pass of learning was applied.

[Figure 5a,b,c]

The weight adjustment formulas (2) and (3) define a standard EFuNN that has the first part updated in an unsupervised mode and the second – in a supervised mode similar to standard RBF networks [58] and their modifications [5,78]. But here the formulas are

applied once for each example (x,y) in an on-line mode, that is similar to the RAN model [61] and its modifications [66]. The standard supervised/unsupervised learning EFuNN is denoted as EFuNN-s/u. In two other modifications of EFuNN, namely double pass learning EFuNN (EFuNN-dp), and gradient descent learning EFuNN (EFuNN-gd) slightly different update functions are used as explained in the next sub-section.

While the connection weights W1 and W2 capture fuzzy co-ordinates of the learned prototypes (exemplars) represented as centres of hyper-spheres, the temporal layer of connection weights W3 from fig.3b captures temporal dependencies between consecutive data examples. If the winning rule node at the moment (t-1) (to which the input data vector at the moment (t-1) was associated) was $r_{max}^{(t-1)}$, and the winning node at the moment t is $r_{max}^{(t)}$, then a link between the two nodes is established as follows:

$$W3(r_{max}^{(t-1)},r_{max}^{(t)})=W3(r_{max}^{(t-1)},r_{max}^{(t)}) + l_3.\ A1(r_{max}^{(t-1)})\ A1(r_{max}^{(t)})) \qquad (4)$$

where: $A1(r^{(t)})$ denotes the activation of a rule node r at a time moment (t); l_3 defines a learning rate - the degree to which the EFuNN associates links between rule nodes (clusters, prototypes) that include consecutive data examples. If $l_3=0$, no temporal associations are learned in an EFuNN structure and the EFuNN from fig.3b becomes the one from fig.3a.

The learned temporal associations can be used to support the activation of rule nodes based on temporal pattern similarity. Here, temporal dependencies are learned through establishing structural links. These dependencies can be further investigated and enhanced through synaptic analysis (at the synaptic memory level) rather than through neuronal activation analysis (at the behavioural level). The ratio spatial-similarity/temporal-correlation can be balanced for different applications through two parameters Ss and Tc such that the activation of a rule node r for a new data example d_{new} is defined through the following vector operations:

$$A1(r) = \left| 1 - Ss.\, D(W1(r), d_{newf}) + Tc.W3(r_{max}^{(t-1)}, r) \right|_{[0,1]} \quad (5)$$

where: $|.|_{[0,1]}$ is a bounded operation in the interval [0,1]; $D(W1(r),d_{newf})$ is the normalised local fuzzy distance value and $r_{max}^{(t-1)}$ is the winning neuron at the previous time moment. Here temporal connections can be given a higher importance in order to tolerate a higher spatial distance. If T_c=0, then temporal links are excluded from the functioning of the system.

Figure 6 shows a schematic diagram of the process of evolving of three rule nodes and setting the temporal links between them for data taken from consecutive frames of a spoken word "eight", also used in section 4.1.

[Figure 6]

The EFuNN system was explained so far with the use of one rule node activation (the winning rule node for the current input data). The same formulas are applicable when the activation of m rule nodes is propagated and used (the so called 'many-of-n' mode, or 'm-of-n' for short). Usually m=3.

The supervised learning in EFuNN is based on the above explained principles, so when a new data example d=(x,y) is presented, the EFuNN either creates a new rule node r_n to memorise the two input and output fuzzy vectors $W1(r_n)= x_f$ and $W2(r_n)= y_f$, or adjusts the winning rule node r_j (or *m* rule nodes respectively).

After a certain time (when certain number of examples have been presented) some neurons and connections may be pruned or aggregated. Aggregation techniques are explained in 3.4. Different pruning rules can be applied for a successful pruning of unnecessary nodes and connections. One of them is given below:

IF (Age(r_j) > OLD) AND (the total activation TA(r_j) is less than a pruning parameter Pr times Age (r_j)) THEN prune rule node r_j,

where Age(r_j) is calculated as the number of examples that have been presented to the EFuNN after r_j had been fist created; OLD is a pre-defined age limit; Pr is a pruning parameter in the range [0,1], and the total activation TA(r_j) is calculated as the number of examples for which r_j has been the correct winning node (or among the m winning nodes in the m-of-n mode of operation).

The above pruning rule requires that the fuzzy concepts of OLD, HIGH, etc. are defined in advance. As a partial case, a fixed value can be used, e.g. a node is OLD if it has existed during the evolving of a FuNN from more than p examples. The pruning rule and the way the values for the pruning parameters are defined, depend on the application task.

There are other functions applied on an EFuNN during its operation. Such functions are: rule extraction (see section 3.3.), MF modification (see 3.4), parameter optimisation (see 3.7).

3.2. EFuNN supervised learning algorithms and inference methods

Three supervised learning algorithms are outlined here that differ in the weight adjustment formulas:

EFuNN-s/u Learning Algorithm:

Set initial values for the system parameters: number of membership functions; initial sensitivity threshold S and maximum radius Rmax; error threshold E; aggregation parameter Nagg - number of consecutive examples after which aggregation is performed (to be explained in section 3.4); pruning parameters OLD and Pr; number m of nodes which activation is propagated to the output layer (for the 'm-of-n' mode); thresholds T_1 and T_2 for rule extraction.

Set the first rule node to memorise the first example (x,y):

$W1(r_0)= x_f$ and $W2(r_0)=y_f$; (6)

Loop over presentations of input-output pairs (x,y)

{

Evaluate the local normalised fuzzy distance D between x_f and the existing rule node connections W1 (formulae (1)).

Calculate the activation A1 of the rule node layer. Find the closest rule node r_k (or the closest m rule nodes in case of m-of-n mode) to the fuzzy input vector x_f so that the input vector is in the receptive field of this rule node.

if A1(r_k) < S_k (sensitivity threshold for the node r_k), create a new rule node for (x_f,y_f)

else

Find the activation of the fuzzy output layer A2=W2.A1(1-D(W1,x))) and the output error Err= || y- y'|| / Nout.

If Err > E,

create a new rule node to accommodate the current example (x_f,y_f)

else

update W1(r_k) and W2(r_k) according to (2) and (3) (in case of *m-of-n* EFuNN update all the *m* rule nodes with the highest A1 activation).

Apply *aggregation* procedure to the existing rule nodes after each group of N_{agg} examples are presented (see explanation in section 3.4).

Update the values for the rule node r_k parameters S_k, R_k, Age(r_k), TA (r_k).

Prune rule nodes if necessary, as defined by pruning parameters.

Modify membership functions if necessary (see section 3.4).

Extract rules from the rule nodes (as explained in 3.3)

}

The two other learning algorithms presented next are exceptions and if it is not explicitly mentioned otherwise, the denotation EFuNN will mean EFuNN-s/u.

(b) EFuNN –dp Learning Algorithm

This is different from the EFuNN-s/u in the weight adjustment formula for W2 that is a modification of (3):

$W2(r_j^{(t+1)}) = W2(r_j^{(t)}) + l_j . (A2 - y_f) . A1(r_j^{(t+1)})$, (7)

meaning that after the first propagation of the input vector and error Err calculation, if the weights are going to be adjusted, W1 weights are adjusted first with the use of (2) and then the input vector x is propagated again through the already adjusted rule node r_j to its position $r_j^{(t+1)}$ in the input space; a new error Err is calculated and after that the W2 weights of the rule node r_j are adjusted. This is a finer weight adjustment than the adjustment in EFuNN-s/u that may make a difference in learning short sequences, but for learning longer sequences it may not cause any difference in the results obtained through the simpler and faster EFuNN-s/u. This is demonstrated in the experiments in section 4.

(c) EFuNN-gd Learning Algorithm

This algorithm is different from the EFuNN-s/u in the way the W1 connections are adjusted which is no more unsupervised, but here a one step gradient descent algorithm is used similar to the RAN model [61]:

$W1(r_j^{(t+1)}) = W1(r_j^{(t)}) + l_j . (W1(r_j^{(t)}) - x_f) (A2 - y_f) A1(r_j^{(t)}) W2(r_j^{(t)})$ (8)

Formulae (8) should be extended when m-of-n mode is applied. The EFuNN-gd algorithm is no more supervised/unsupervised and the rule nodes are no more allocated at the cluster centers of the input space. This leads to a slower learning and more difficult interpretation of the EFuNN structure in terms of extracting meaningful rules. These are the main reasons why this algorithm is not used further in the paper.

An important characteristic of the EFuNN learning is the local element tuning. Only one (or *m*, in the *m-of-n* mode) rule node will be either updated, or created for each data example. This makes the learning procedure very fast (especially in the case when

specialised parallel hardware platforms are used). Another advantage is that learning a new data example does not cause forgetting of old ones [18,65]. A third advantage is that new input and new output variables can be added during the learning process, thus making the EFuNN system more flexible to accommodate new information, once such becomes available, without disregarding the already learned information.

The use of MFs and membership degrees (layer two of neurons), and also the use of normalised local fuzzy difference, makes it possible to deal with *missing attribute value*s. In such cases, the membership degrees of all MFs will be set to 0.5 indicating that the value, if it existed, may belong to them. Preference, in terms of which fuzzy MF the missing value might belong to, can also be represented through assigning appropriate membership degrees, e.g. 0.7 degrees to "Small" means that the value is more likely to be small rather than "Medium", or "Large".

The supervised learning algorithms above allow for an EFuNN system to always evolve and learn when a new input-output pair of data becomes available. This is an *active mode of learning*. In another mode, *passive (inner, sleep) learning*, learning is performed when there is no input pattern presented. This may be necessary after an initial learning has already been applied. In this case existing connections, that store previously fed input patterns (prototypes), are used as "echo" data (here denoted as ECO) to reiterate the learning process. This type of learning may be applied in case of a short on-line presentation of the data, when only small portion of data is learned, and then the training is refined through the ECO learning method [31,35].

The evolved EFuNN can perform *inference* when recalled on new input data. *The EFuNN inference method* consists of calculating the output activation value and is part of the EFuNN supervised learning method when only the input vector x is propagated through the EFuNN. In case of *one-of-n* learning mode, the inference is based on the highest rule

node activation where the new input data falls into the node's receptive field, or just the highest rule node activation if the new input vector does not fall into any of the node receptive fields. In the *m-of-n* mode there will be *m* rules used in the EFuNN inference process.

3.3. Knowledge–based learning in EFuNNs: rule insertion and rule extraction

EFuNNs are adaptive rule-based systems. Manipulating rules is essential for their operation. This includes rule insertion, rule extraction, and rule adaptation.

At any time (phase) of the evolving (learning) process fuzzy or exact rules can be *inserted* and extracted. Insertion of fuzzy rules is achieved through setting a new rule node r_j for each new rule, such that the connection weights $W1(r_j)$ and $W2(r_j)$ of the rule node represent this rule. For example, the fuzzy rule *(IF x_1 is Small and x_2 is Small THEN y is Small*) can be inserted into an EFuNN structure by setting the input connections of a new rule node from the fuzzy input nodes x_1- Small and x_2- Small to a value of 1, and setting the output connection of this rule node to the fuzzy output node y-Small to a value of 1. The rest of the connections are set to a value of zero. Similarly, an exact rule can be inserted into an EFuNN structure, e.g. *IF x_1 is 3.4 and x_2 is 6.7 THEN y is 9.5.* Here, the membership degrees to which the input values x_1 = 3.4 and x_2=6.7, and the output value *y=9.5* belong to the corresponding fuzzy values are calculated and attached to the corresponding connection weights.

Rule insertion and rule extraction are important operations of an EFuNN as it is a knowledge-based connectionist model. Each rule node r_j can be expressed as a fuzzy rule, for example:

r_j: IF x_1 is Small 0.85 and x_1 is Medium 0.15 and x_2 is Small 0.7 and x_2 is Medium 0.3

(Radius of the receptive field Rj=0.1, maxRj=0.75)

THEN y is Small 0.2 and y is Large 0.8 (20 out of 175 examples are associated with this rule),

where the numbers attached to the fuzzy labels denote the degree to which the centers of the input and the output hyper-spheres belong to the respective MFs. The degrees associated to the condition elements are the connection weights from the matrix W1. Only values that are greater than a threshold T1 are left in the rules. The degrees associated with the conclusion part are the connection weights from W2 that are greater than a threshold of T2. The other parameters associated with the rule represent the importance and the strength of the rule. An example of rules extracted from a bench-mark dynamic time series data is given in sub-section 3.5. The two thresholds T1 and T2 are used to disregard the connections from W1 and W2 that represent small and insignificant membership degrees (e.g., less than 0.1).

3.4. Rule node aggregation and membership function modification

Another knowledge-based technique applied to EFuNNs is *rule node aggregation*. Through this technique several rule nodes are merged into one as it is shown in fig.7a,b and c on an example of three rule nodes r_1, r_2 and r_3 (only the input space is shown there).

[Figure 7a,b,c]

For the aggregation of three rule nodes r_1 , r_2, and r_3 the following two aggregation rules can be used to calculate the W1 connections for the new aggregated rule node r_{agg} (the same formulas are used to calculate the W2 connections)

- as a geometrical center of the three nodes:

$$W1(r_{agg})=(W1(r_1)+W1(r_2)+W1(r_3))/ 3 \quad (9)$$

- as a weighted statistical center:

$$W2(r_{agg})=(W2(r_1).Nex(r_1)+W2(r_2).Nex(r_2)+W2(r_3).Nex(r_3))/Nsum \quad (10)$$

where: $Nex(r_{agg})= Nsum = Nex(r_1)+Nex(r_2)+Nex(r_3)<=Rmax$; $Rr_{agg} = d(W1(r_{agg}), W1(r_j)) + Rj$; r_j is the rule node from the three nodes that has a maximum distance from the new node r_{agg} and Rj is its radius of the receptive field. The three rule nodes will aggregate only if the radius of the aggregated node receptive field is less than a pre-defined maximum radius Rmax.

In order to select for a given node r_j the other nodes it will aggregate with, two subsets of nodes are formed – the subset of nodes r_k that if activated to a degree of 1 will produce an output value $y'(r_k)$ that is different from $y'(r_j)$ in less than the error threshold E, and the subset of nodes that if activated to a degree of 1 will cause output values different from $y'(r_k)$ in more than E. These two subsets are derived from the W2 connections. Than all the rule nodes from the first subset that are closer to r_j in the input space than the closest to r_j node from the second subset in terms of W1 distance, get aggregated if the radius of the new node r_{agg} is less than the pre-defined limit Rmax (as illustrated in fig.7c.

Instead of aggregating all the rule nodes that are closer to a rule node r_j than the closest node from a neighbouring class (these nodes form an aggregation pool for r_j), it is possible to keep the closest node from the aggregation pool to the other class out of the aggregation procedure - as a separate node – a "guard" (see fig.8), thus preventing miss-classification between the two classes on the bordering area. The "guardian" node represents the same class as r_j.

[Figure 8]

Through node creation and consecutive aggregation, an EFuNN system can adjust over time to changes in the data stream and at the same time – preserve its generalisation capabilities.

Through analysis of the weights W3 of an evolved EFuNN, *temporal correlation* between time consecutive exemplars can be expressed in terms of rules and conditional probabilities, e.g.:

IF $r_1^{(t-1)}$ THEN $r_2^{(t)}$ (0.3) (11)

The meaning of the above rule is that some examples that belong to the rule (prototype) r_2 follow in time examples from the rule prototype r_1 with a relative conditional probability of 0.3.

Changing (evolving) MF is another knowledge-based operation that may be needed for a refined performance after a certain time moment of the EFuNN's operation. Changing the shape of the MF in a fuzzy neural structure such as FuNN through gradient descent algorithm is suggested in [39,45]. Another approach to changing MF is using the data distribution for each variable, i.e. calculating in an on-line mode the histogram of each variable and placing the centers of its MFs at the middle of the areas that are of highest density. Both changing the number and the shape of MFs may be needed for a better performance of an EFuNN. For example, instead of three MF, the system may perform better if it had five MF for some of the variables. In EFuNNs there are several possibilities to implement such dynamical changes of MF as it is graphically illustrated on fig.9a,b:

(a) New MF are created (fuzzy nodes are inserted) in the most dense areas of the input space without a need for the old MF to be changed (fig.9a). The degree to which each cluster centre (each rule node) belongs to the new MF can be calculated through: (i) defuzzifying the centres into real values with the use of the existing MF; (ii) finding the membership degrees to which these values belong to the new MF; (iii) creating new connections that have as weights these membership degrees;

(b) All MFs change in order new ones to be introduced. For example, all stored fuzzy exemplars in W1 and W2 that have three MFs, are defuzzified (e.g., through the center of gravity deffuzification technique) and than used to evolve a new EFuNN structure that has five MFs (fig.9b)

[Figure 9a,b]

3.5. Examples of EFuNN learning, aggregation and rule extraction methods for a chaotic benchmark time series prediction task

Here the operation of EFuNNs is illustrated on the Mackey-Glass chaotic time series data (see also [16,45,61,5]) defined by the Mackey Glass time delay differential equation:

$$\frac{d(x)}{d(t)} = \frac{a\, x(t-\tau)}{1 + x^{10}(t-\tau)} - b\, x(t) \qquad (12)$$

This series behaves as a chaotic time series for some values of the parameters x(0) and τ [16,45]. Here x(0) = 1.2, τ = 17, a=0.2, b=0.1 and x(t) = 0 for t < 0 were assumed. The input-output data for evolving an EFuNN from the Mackey Glass time series data is in the following format: input vector: [x(t), x(t-6), (t-12) x(t-18))]; output vector [x(t+6)]. The task is to predict future values x(t+6) from 4 points spaced at six time intervals in the past.

Example 1. The following values for the EFuNN parameters were set: initial value for sensitivity threshold S of 0.9; error threshold E=0.1; a maximum radius Rmax=0.2; a rule extraction threshold of 0.5; aggregation is performed after each consecutive group of 100 examples is presented; “m-of-n” mode, where m=1, is used; the number of membership

functions MF is 5; 1000 consecutive data examples are used. Some experimental results of the on-line evolving of an EFuNN are shown in fig.10, namely: (a) the desired versus the predicted six steps ahead values through one-pass on-line learning; (b) the absolute-, the local on-line RMSE (LRMSE), and the local on-line NDEI (LNDEI) error over time as described below; (c) the number of the rule nodes created and aggregated over time; (d) a plot of the input data vectors (circles) and the evolved rule nodes (crosses) projected in the two dimensional input space of the first two input variables x(t) and x(t-6).

[Figure 10a,b,c,d]

The generalisation error of an EFuNN on a next new input vector (or vectors) from the input stream calculated through the evolving process is called *local on-line generalisation error*. The local on-line generalisation error at the moment *t* for example, when the input vector is x(t), and the calculated by the evolved EFuNN output vector is y(t)' is expressed as Err(t)= (y(t) - y(t)')/Nout, where Nout is the number of the outputs of the EFuNN. The local on-line root mean square error, and the local on-line non-dimensional error index LNDEI(t) can be calculated at each time moment *t* as:

$$LRMSE(t)=\sqrt{(\Sigma_{i=1,2\ldots,t}(Err(i)^2)/t)};\ LNDEI(t)=LRMSE(t)/std(y(1):y(t)), \qquad (13)$$

where std(y(1):y(t)) is the standard deviation of the output data points from 1 to t.

For the chosen values of the parameters, there were 86 rule nodes evolved each of them represented as one rule (some rules are shown in tabl.1). These rules and the EFuNN inference mechanism define a system that is equivalent to the equation (12) in terms of the chosen inputs and output variables subject to the calculated error.

[Table 1]

After certain time moment the LRMSE and LNDEI converge to constant values subject to a small error, in the example from fig.10 - LRMSE=0.102, LNDEI=0.191. Generally speaking, in the case of compact and bounded problem space the error can be made

sufficiently small subject to appropriate selection of the parameter values for the EFuNN, an issue discussed in 3.7. In the experiment above the chosen error tolerance was comparatively high, but the evolved EFuNN was kept of a small size. If the chosen error threshold E was smaller (e.g. 0.05, or 0.02) more rule nodes would have been evolved and better prediction accuracy could have been achieved. Different EFuNNs have different optimal parameter values, which also depends on the task (e.g. time series prediction, classification).

The example here demonstrates that an EFuNN can learn a complex chaotic function through on-line evolving from one-pass data propagation. But the real strength of the EFuNNs is in learning time-series that change their dynamics through time, e.g. changing values for the parameter τ from equation (12). Time series processes with changing dynamics could be of different origin, e.g.: biological, financial and economics, environmental, industrial processes, control.

EFuNNs can also be used for off-line training and testing similar to other standard NN techniques. This is illustrated in another example shown in fig.11.

Example 2. The following parameter values are set before an EFuNN is evolved: MF=5; initial S=0.9; E=0.05; m=1; Rmax=0.2. The EFuNN is evolved on the first 500 data examples from the same Mackey-Glass time series as in example1. Fig.11a shows the desired versus the predicted on –line values on the first 500 examples of the time series. After the EFuNN is evolved, it is tested for a global generalisation on the second 500 examples. Figure 11b shows the desired versus the predicted by the EFuNN values for these examples in an off-line mode.

In a general case, the global generalisation root mean square error RMSE and the non-dimensional error index are evaluated on a set of p new (test) examples from the problem space as follows:

$RMSE=\sqrt{(\Sigma_{i=1,2,...p}[(y_i - y_i')^2]/p}$; NDEI=RMSE/std(1:p),

(14)

where: *std* (1:p) is the standard deviation of the data from 1 to p in the test set . The evaluated in this example RMSE is 0.01, and the NDEI is 0.046. After having evolved an EFuNN on a small, but representative part of the whole problem space, its global generalisation error can saturate to a satisfactory value.

[Figure 11]

An EFuNN can also be tested for *on-line test error* on a test data while further training on this data is still performed. The on-line local test error is slightly smaller for the example above.

Generally speaking, if a continuous, incremental learning on all data is possible (in many cases of time series prediction it is) EFuNNs should be continuously evolved all the time in an adaptive, life-long learning mode, always improving their performance. Typical application of EFuNN would be modelling and predicting of a continuous financial time series, modelling of large DNA data sequences, or adaptive spoken word classification (section 4 and table 3).

3.6. Unsupervised and hybrid (supervised/unsupervised) learning in EFuNNs

The unsupervised method of learning assumes that there are no desired output values available and the EFuNN evolves its rule nodes from the input space. A node allocation is based only on the defined maximum radius Rmax. If a new data item d activates a certain rule node r_j above the level of $S_j = (1- R_j)$, this rule node is adjusted to accommodate the new data item according to formulae (2), otherwise a new rule node is created. The

EFuNN unsupervised learning method is based on the steps described as part of the supervised learning method when only the input vector x is available for the current input data item d. And the input vector can be either fuzzy or non-fuzzy.

This method of learning is illustrated here on speech data that represents a spoken word "eight" (in a phonemic representation: /silence//ei//t//silence/). In the experimental results shown in fig.12, three time lags of 26 mel scale coefficients taken from a window of 12 ms of the speech signal, with an overlap of 50%, are used to form 78–element input vectors. The input vectors are plotted over time in fig.12b.

Each new input vector from the spoken word is either associated with an existing rule node that is modified to accommodate this data, or a new rule node is created. Regularly the rule nodes are aggregated. After the whole word is presented the aggregated rule nodes represent the centres of the anticipated phoneme clusters without the concept of phonemes being introduced to the system (fig.12a). The latter figure shows clearly that three rule nodes were evolved after aggregation, that represent the stable sounds as follows: frames 0-53 and 96-170 are allocated to rule node 1 that represents /silence/; frames 56-78 are allocated to rule node 2 that represents the phoneme /ei/; frames 85-91 are allocated to rule node 3 that represents the phoneme /t/; the rest of the frames represent transitional states, e.g. frames 54-55 the transition between /silence/ and /ei/, frames 79-84 – the transition between /ei/ and /t/, and frames 92-96 – the transition between /t/ and silence/, are allocated to some of the closest rule nodes in the input space. If a higher sensitivity threshold was used, there would have been additional rule nodes evolved to represent these short transitional sounds.

[Figure 12a,b]

When more pronunciations of the word 'eight' are presented to the unsupervised EFuNN system, the system refines the phoneme regions and the phoneme rule nodes.

Both the supervised and the unsupervised learning methods for EFuNNs are based on the same principles of building the W1 layer of connections. Either of them can be applied on an evolving EFuNN, so that if there are known output values the system will use a supervised learning method, otherwise – it will apply the unsupervised learning method on the same structure. For example, after having evolved in an unsupervised way an EFuNN from the spoken word "eight" input data, more data that are labelled with the appropriate phoneme labels, can be used to continue the learning process of this EFuNN, now in a supervised mode.

The EFuNNs also allow for reinforcement learning when instead of exact desired values, only their fuzzy labels become known during the evolving process. Such values are for example, Small, Not Very Good, etc. This type of learning is outside the scope of the paper.

3.7. On-line parameter adaptation and feature evaluation in EFuNNs

Once set, the initial values of the following EFuNN parameters can be either kept fixed during the entire operation of the system, or can be adapted (optimised) according to the incoming data: number of membership functions; value for m-of-n parameter; error threshold E; maximum receptive field Rmax; rule extraction thresholds T1 and T2; number of examples for aggregation Nagg; pruning parameters OLD and Pr. Adaptation can be achieved through the analysis of the behaviour of the system, or through a feedback connection from higher level modules in the ECOS architecture.

One scenario for on-line parameter adaptation is that after some number of examples presented, or certain time, the initial parameter values will start to adapt to the characteristics of the input data, such as distribution, dimensionality, time-dependence, etc. For example, after certain number of examples or if the distribution of the incoming

data does not change significantly from one time segment to another, the restriction Rmax can be removed as the receptive field of every rule node will be restricted by the closest to this rule node example that belongs to another class. This is true if there are sufficient data points that cover most of the areas of the problem space. Another parameter, Nagg, can also be removed as a restriction after certain distribution of the input-output space is achieved and aggregation can be applied after every single example is learned.

In another scenario, genetic algorithms (GA) and evolutionary programming techniques [23,20,78,79] can be applied to optimise the EFuNNs structural and functional parameters through evolving populations of EFuNNs over generations and evaluating each EFuNN in the population at certain time intervals. After that, only the best EFuNNs are kept and reproduced in another population, which process continuous to evolve in an on-line mode. Evolutionary strategies that constitute another class of evolutionary computational methods, are also suitable techniques to optimise the EFuNN parameters over time.

The evaluation of the relevance of the input variables to the task can be done in an on-line mode through iterative calculation of the correlation of each input variable to each output class, or to each membership function of the output variables, e.g. Corr (x1, [y is Small, Medium, High]) = [0.7, 0.4, -0.3] thus producing a continuous information on the most relevant input features which information is added to the extracted rules and knowledge.

3.8. Adding new inputs and outputs to EFuNN during on-line learning

The local learning procedure and the local normalised fuzzy distance used in the EfuNN architecture allow for adding new inputs or/and outputs at any time of the system operation. In this respect the problem space, EFuNN is operating in, is open and the

accumulated knowledge in an EFuNN structure is incremental. This way of learning is typical for humans who always use new sources of information and add new input variables, classes, concepts in a continuous manner.

4. Comparative Analysis of EFuNNs and other NN and AI Techniques for On-Line, Knowledge-based Learning

4.1. EFuNN versus traditional NN and fuzzy-neural systems

EFuNNs are learning machines that can learn in an on-line mode any data set, regardless of the class of the problem (function approximation, time series prediction, classification, etc.) either in a supervised-, or in an unsupervised-, or in a hybrid learning mode, subject to appropriate parameter values selected and certain minimum number of examples presented.

When issues such as applicability of the EFuNN model, learning accuracy, generalisation and convergence are discussed for different tasks, two cases must be distinguished:

(a) The incoming data are from a compact and bounded data space. In this case the more data vectors are used for evolving an EFuNN, the better its generalisation is on the whole problem space. After a time moment T, if appropriate values for the EFuNN parameters are used, each of the fuzzy input and the fuzzy output spaces (they are compact and bounded) will be covered by hyper-spheres of the evolved rule nodes that will have different receptive fields in the general case. We can assume that by a certain time moment T a sufficient number of examples from the stream have been presented and rule node hyper-spheres cover the problem space to a desired accuracy. The local on-line error will saturate at this time because any two associated compact and bounded fuzzy spaces X_f and Y_f that represent a problem space, can be fully covered by a sufficient number of associated (possibly overlapping) fuzzy hyper-

spheres (see fig.5). The number of these spheres (the number of rule nodes) depends a great deal on the error threshold E, set before the training of the system.

(b) The incoming data are from an open problem space, where data dynamics and data distribution may change over time in a continuous way. In this case the local on-line error will depend on the closeness of the new input data to already stored prototypes in the existing rule nodes.

The well-established NN and AI techniques have difficulties when applied for on-line, knowledge based learning [18,65,38]. For example, the multi-layer perceptrons (MLP) and the backpropagation learning algorithm have the following problems: catastrophic forgetting [65,38], local minima problem [3,10], difficulties to extract rules [38], not able to adapt to new data without re-training on old ones [65], too long training when applied to large data sets. Many neuro-fuzzy systems, such as ANFIS [63], FuNN [39], neo-fuzzy neuron [83], cannot update the learned rules through continuous training on additional data without suffering the catastrophic forgetting. The SOM [47,48] may not be efficient when applied for unsupervised adaptive learning on new data as SOM assume a fixed structure and a fixed grid of nodes connected in a topological output space that may not be appropriate for the projection of a particular data set. The radial basis neural networks require clustering to be performed first and then the backpropagation algorithm applied [38]. They are not efficient for adaptive, on-line learning either, despite of some improvements [5,78].

The EFuNN evolving procedure leads to a similar local on-line error as RAN [61] and its modifications [66,21], but EFuNNs allow for meaningful rules to be extracted and inserted at any time of the operation of the system thus providing the knowledge about the problem and reflecting to changes in its dynamics. In this respect the EFuNN is a flexible, on-line, knowledge engineering and statistical model. As a statistical model the EfuNN

performs clustering of the input space. The EfuNN structure also reflects the data distribution of the input-output space.

The above said is illustrated here with two experiments. The first one deals with the task of on-line time series prediction of the Mackey-Glass data. The second one deals with a classification task on a case study data of spoken digits.

Experiment 1. Time series prediction

A description of the bench-mark data set of Mackey-Glass data was given in section 3.5. Here the standard CMU benchmark format of the time series is used. The data was generated with τ=17, using a second order Runge-Kutta method with a step size of 0.1, of four inputs: x(t), x(t-6), x(t-12) and x(t-18), and one output: x(t+85). Training data is from t=200 to t=3200, while test data is from t= 5000 to 5500. All the 3,000 training data were used to evolve three types of EFuNNs. For the EFuNN-s/u and EFuNN-dp the following initial values of the parameters were chosen manually: MF=3, S=0.9, E=0.05, m=1, Rmax=0.2, n_{agg}=100. For the EFuNN-gd model the following initial values of the parameters were used: MF=3, S=0.9, E=0.05, m=1, Rmax=0.2, n_{agg}=100. The number of the centers (rule nodes in EFuNN) and the local on-line LNDEI is calculated and compared with the one for the RAN model [61] and its modifications [66]. The results are shown in table 2. The three modifications of EFuNN result in a smaller on-line error than the other methods and in a reasonable number of rule nodes.

[Table 2]

If EFuNNs use linear equations for calculating the activation of the rule nodes (instead of Gaussian functions and exponential functions as it is in RAN) the EFuNN learning procedure is much faster than the learning procedure in RAN and its modifications.

EFuNN also produces a better on-line generalization, which is a result of a more accurate node allocation during the learning process.

Software simulators as well as data sets used in the experiments above are available from: *http://divcom.otago.ac.nz/infosci/kel/CBIIS.html (-> software -> efunns).*

Experiment 2: Spoken word classification

The task is recognition of speaker independent pronunciations of English digits from the Otago Corpus database (http://kel.otago.ac.nz/hyspeech/corpus/) [75]. 17 speakers (12 males and 5 females) are used for training and other 17 speakers (12 males and 5 females) are used for off-line testing. Each speaker utters 30 instances of English digits during recording session in a quiet room (clean data) for a total of 510 training and 510 testing utterances. Eight mel frequency scale cepstrum coefficients (MFSCC) and log-energy are used as acoustic features. In order to assess the performance of EFuNNs in this application, a comparison with Linear Vector Quantisation (LVQ) is accomplished. Clean training speech is used to train both LVQ and EFuNN. Office noise is introduced to the test speech data to evaluate the behaviour of the recognition systems in a noisy environment, with a Signal-to-Noise Ratio of 10dB. The classification off-line test accuracy for LVQ and EFuNN, and also the local on-line test accuracy for EFuNN, are evaluated and presented in table 3. The LVQ model has the following parameter values: code-book vectors – 396, training iterations 500 on the whole training set. The EFuNN has the following parameter values: 1 training iteration; 3 MFs, 157 rule nodes, initial S=0.9, E=0.1, Rmax=0.2, number of examples for aggregation n_{agg}=100.

The results show that the EFuNN with off-line learning and testing on new data performs much better than the LVQ method (Table 3). As EFuNN allows for continuous training on new data, further testing and also training of the EFuNN on the test data in an on-line mode leads to a significant improvement of the accuracy.

[Table 3]

EFuNNs in an on-line learning mode can be used as building blocks for creating adaptive speech recognition systems that are based on the ECOS framework (fig.2). Such systems would be able to adapt to new speakers and new accents, and add new words to their dictionaries at any time of their operation.

4.2. EFuNNs, Evolutionary Computation (EC) and Case-Based Reasoning (CBR) methods

EC is concerned with population-based search and optimisation of an individual system through generations of populations of systems [23]. EC is applied to the optimisation of different structures and processes, one of them being the connectionist structures and connectionist learning processes [13,78,79]. EC, and the genetic algorithms (GA) in particular, include in principal a stage of development of each individual system, so that a system develops, evolves through interaction with the environment that is also based on the genetic material embodied in the system. This process of development has been in many cases ignored or neglected as insignificant from the point of view of the long process of generating hundreds of generations each of them containing hundreds of individuals. EFuNNs deal with the process of interactive, on-line adaptive learning of a single system that evolves from incoming data or through interaction with the environment. The system can either have its parameters (genes) pre-defined, or self-adjusted during the learning process starting from some initial values. There are several ways in which EC and EFuNNs can be inter-linked. For example, it is possible to use EC to optimise the parameters of an EFuNN (or an ECOS) at a certain time of its operation, or to use EFuNNs for the development of the individual systems (individuals) as part of the global EC process.

EFuNNs and case-based reasoning methods [12] are similar in the sense that they store exemplars and measure similarities, but an EFuNN has a more flexible inference mechanism and it is connectionist based that brings all the advantages of the NN methods.

5. Conclusions and directions for further research

This paper presents evolving fuzzy neural network systems EFuNNs as a realisation of one of the main modules in a framework of evolving connectionist systems ECOS. ECOS and EFuNNs comprise methods and systems for general purpose on-line adaptive learning. EFuNNs have features that address the seven major requirements to the intelligent information systems presented in section one. A significant advantage of EFuNNs is the local learning, which allows for a fast adaptive learning that is robust to catastrophic forgetting. Only few connections and nodes are changed, or created after the entry of a new data item. This is in contrast to the global learning algorithms where, for each input vector, all connection weights change, thus making the system prone to catastrophic forgetting when applied for adaptive, on-line learning tasks.

ECOS incorporate important AI features, such as: adaptive learning; non-monotonic reasoning; knowledge manipulation in the presence of imprecision and uncertainties; knowledge acquisition and explanation. ECOS and EFuNNs have features of knowledge-based systems, logic systems, case-based reasoning systems, and adaptive connectionist-based systems, all together. Through self-organisation and self-adaptation during the learning process, they allow for solving difficult engineering tasks as well as for simulation of emerging, evolving biological and cognitive processes to be attempted. The life-long learning mode (also implemented in EFuNN) is the natural learning mode of all biological systems [14,44,73,82].

The EFuNN methods and the ECOS can be implemented in software or/and in hardware with the use of either conventional, or new computational techniques. EFuNN simulators are available from: http://divcom.otago.ac.nz/infosci/kel/CBIIS.html. The EFuNN applicability is broad and spans across several application areas of computer and information science where systems, that learn from data and improve continuously, are needed. That includes:

computer systems that learn speech and language;

adaptive process control;

autonomous navigation of intelligent robots;

intelligent agents on the WWW;

intelligent decision support systems;

bioinformatics;

domestic appliances and intelligent buildings;

systems that learn and control brain-, and body states from a biofeedback.

Acknowledgements

This research is part of a research programme “Connectionist-based intelligent information systems” funded by the New Zealand Foundation for Research Science and Technology, contract UOOX0016. I would like to thank Irena Koprinska, Qun Song and Georgi Iliev for their contribution to the implementation of the ECOS and EFuNN simulators. I would like to thank he reviewers for their useful comments and suggestions that helped me to improve the paper significantly from its initial version.

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, pp.220:227, Sept. (1975)
2. Amari, S. and Kasabov, N. eds, “Brain-like Computing and Intelligent Information Systems”, Springer Verlag,1998.
3. Amari, S., “Mathematical foundations of neuro-computing”, Proc. of IEEE, 78 (9), Sept. (1990)
4. Arbib, M. (ed) The Handbook of Brain Theory and Neural Networks, The MIT Press, 1995.
5. Blanzieri, E., P.Katenkamp, “Learning radial basis function networks on-line”, in: Proc. of Intern. Conf. On Machine Learning, 37-45 (1996)
6. Bottu and Vapnik, “Local learning computation”, Neural Computation, 4, 888-900 (1992)
7. Carpenter, G. and Grossberg S., Pattern recognition by self-organizing neural networks , The MIT Press, Cambridge, Massachusetts (1991)
8. Carpenter, G. and S. Grossberg, “ART3: Hierarchical search using chemical transmitters in self-organising pattern-recognition architectures”, Neural Networks, 3(2) 129-152(1990).
9. Carpenter, G. S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, “FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps,” IEEE Transactions of Neural Networks , vol.3, No.5, 698-713 (1991).

10. Cybenko, G., Approximation by super-positions of sigmoidal function, Mathematics of Control, Signals and Systems, 2, 303-314 (1989)
11. 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)
12. Doo-Il Choi and Sang-Hui Park, “Self creating and organizing neural networks”, IEEE Trans. Neural Networks, vol.5 (4) pp.561-575, Jul.1994
13. Duda and Hart, “Pattern classification and scene analysis”, New York: Willey (1973)
14. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992).
15. 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).
16. Farmer, J.D., and Sidorowich, Predicting chaotic time series, Physical Review Letters, 59, 845 (1987)
17. Freeman, J., D. Saad, “On-line learning in radial basis function networks”, Neural Computation vol. 9, No.7 (1997).
18. French, “Semi-destructive representations and catastrophic forgetting in connectionist networks, Connection Science, 1, 365-377 (1992)
19. Fritzke, B. “A growing neural gas network learns topologies”, Advances in Neural Information Processing Systems, vol.7 (1995).
20. 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)

21. Furlanello, C., D.Giuliani, E.Trentin, “Connectionist speaker normalisation with generalised resource allocation network, in: Advances in NIPS 7 (eds D.Toretzky, G.Tesauro, T.Lean) MIT Press, 1704-1707 (1995)
22. 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).
23. Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley (1989)
24. Goodman, R., C.M. Higgins, J.W. Miller, P.Smyth, "Rule-based neural networks for classification and probability estimation", Neural Computation, 14, 781-804 (1992).
25. 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).
26. Hassibi and Stork, “Second order derivatives for network pruning: Optimal Brain Surgeon,” in: Advances in Neural Information Processing Systems, 4, 164-171, (1992).
27. Hech-Nielsen, R. “Counter-propagation networks”, IEEE First int. conference on neural networks, San Diego, vol.2, pp.19-31 (1987)
28. Heskes, T.M., B. Kappen, “On-line learning processes in artificial neural networks”, in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233, (1993).
29. Ishikawa, M., "Structural Learning with Forgetting", Neural Networks 9, 501-521, (1996).
30. Kasabov, N. "Adaptable connectionist production systems”. Neurocomputing, 13 (2-4) 95-117, (1996).

31. Kasabov, N. The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems, Journal of Advanced Computational Intelligence, 2 (6) 1998, 1-8
32. Kasabov, N., "Investigating the adaptation and forgetting in fuzzy neural networks by using the method of training and zeroing", Proceedings of the International Conference on Neural Networks ICNN'96, Plenary, Panel and Special Sessions volume, 118-123 (1996).
33. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2-20 (1996).
34. Kasabov, N., "A framework for intelligent conscious machines utilising fuzzy neural networks and spatial temporal maps and a case study of multilingual speech recognition", in: Amari, S. and Kasabov, N. (eds) Brain-like computing and intelligent information systems, Springer Verlag, 106-126 (1998)
35. Kasabov, N., "ECOS: A framework for evolving connectionist systems and the ECO learning paradigm", Proc. of ICONIP'98, Kitakyushu, Japan, Oct. 1998, IOS Press, 1222-1235
36. Kasabov, N., "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in: Yamakawa and Matsumoto (eds), Methodologies for the Conception, design and Application of Soft Computing, World Scientific, 1998, 271-274
37. Kasabov, N., E. Postma, and J. Van den Herik, "AVIS: A Connectionist-based Framework for Integrated Audio and Visual Information Processing", in: Yamakawa and Matsumoto (eds), Methodologies for the Conception, design and Application of Soft Computing, World Scientific, 1998, 422-425

38. Kasabov, N., Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA, (1996).
39. Kasabov, N., J. S Kim, M. Watts, A. Gray, "FuNN/2- A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition", Information Sciences", 101(3-4): 155-175 (1997)
40. Kasabov, N., M. Watts, "Genetic algorithms for structural optimisation, dynamic adaptation and automated design of fuzzy neural networks", in: Proceedings of the International Conference on Neural Networks ICNN'97, IEEE Press, Houston (1997).
41. Kasabov, N., R. Kozma, R. Kilgour, M. Laws, J. Taylor, M. Watts, and A. Gray, "A Methodology for Speech Data Analysis and a Framework for Adaptive Speech Recognition Using Fuzzy Neural Networks and Self Organising Maps", in: Kasabov and Kozma (eds) Neuro-fuzzy techniques for intelligent information systems, Physica Verlag (Springer Verlag) 1999
42. Kasabov, N., Song, Q. "Dynamic, evolving neuro- fuzzy inference systems, *IEEE Transactions of Fuzzy Systems*, accepted for publication (2001)
43. Kasabov, N., Watts, M. Spatial-temporal evolving fuzzy neural networks and applications for adaptive phoneme recognition, TR 99/03 Department of Information Science, University of Otago (1999)
44. Kasabov, N., Adaptive learning system and method, Patent, Reg.No. 503882, New Zealand.
45. Kim, J.S. and Kasabov, N. HyFIS: adaptive neuro-fuzzy systems and their application to non-linear dynamical systems, *Neural Networks*, 12 (9) 1301-1319 (1999)
46. Kawahara, S., Saito, T. "On a novel adaptive self-organising network", Cellular Neural Networks and Their Applications, 41-46 (1996)

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. Le Cun, Y., J.S. Denker and S.A. Solla, “Optimal Brain Damage”, in: Touretzky, D.S., ed., Advances in Neural Information Processing Systems, Morgan Kaufmann, 2, 598-605 (1990).
51. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).
52. Maeda, M., Miyajima, H. and Murashima, S., “A self organising neural network with creating and deleting methods, Non-linear Theory and its Applications, 1, 397-400 (1996)
53. Mandziuk, J., Shastri, L. “Incremental class learning approach and its application to hand-written digit recognition, Proc. of the fifth int. conf. on neuro-information processing, Kitakyushu, Japan, Oct. 21-23, 1998
54. Massaro, D., and M.Cohen, "Integration of visual and auditory information in speech perception", Journal of Experimental Psychology: Human Perception and Performance, Vol 9, pp.753-771, (1983).
55. McClelland, J., B.L. McNaughton, and R.C. Reilly "Why there are Complementary Learning Systems in the Hippocampus and Neo-cortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory", CMU TR PDP.CNS.94.1, March, (1994).
56. Miller, D.J., Zurada and J.H. Lilly, "Pruning via Dynamic Adaptation of the Forgetting Rate in Structural Learning," Proc. IEEE ICNN'96, Vol.1, p.448 (1996).
57. Mitchell, M.T., "Machine Learning", MacGraw-Hill (1997)

58. Moody, J., Darken, C., Fast learning in networks of locally-tuned processing units, Neural Computation, 1(2), 281-294 (1989)
59. 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).
60. Murphy, P. and Aha, D. “UCI Repository of machine learning databases, Irvin, CA: University of California, Department of Information and Computer Science (1994), (http://www.ics.uci.edu/~mlearn/MLRepository.html)
61. Platt, J., “A resource allocating network for function interpolation, Neural Computation, 3, 213-225 (1991)
62. Quartz, S.R., and T.J. Sejnowski, “The neural basis of cognitive development: a constructivist manifesto”, Behaviour and Brain Science, 1999
63. R. Jang, “ANFIS: adaptive network-based fuzzy inference system”, IEEE Trans. on Syst., Man, Cybernetics, 23(3), May-June, 665-685, (1993).
64. Reed, R., “Pruning algorithms - a survey”, IEEE Trans. Neural Networks, 4 (5) 740-747, (1993).
65. Robins, A. and Frean, M. “Local learning algorithms for sequential learning tasks in neural networks, Journal of Advanced Computational Intelligence, vol.2, 6 (1998)
66. Rosipal,R., M.Koska, I.Farkas, “Prediction of chaotic time-series with a resource-allocating RBF network, Neural Processing Letters, 10:26 (1997)
67. Rummery, G.A., and M. Niranjan, “On-line Q-learning using connectionist systems”, Cambridge University Engineering Department, CUED/F-INENG/TR 166 (1994)
68. S.R.H. Joseph, “Theories of adaptive neural growth”, PhD Thesis, University of Edinburgh, 1998
69. Saad, D. (ed) On-line learning in neural networks, Cambridge University Press, 1999

70. Sankar, A., and R.J. Mammone, "Growing and pruning neural tree networks", IEEE Trans. Comput. 42(3) 291-299 (1993).
71. Schaal, S. and C. Atkeson, "Constructive incremental learning from only local information, Neural Computation, 10, 2047-2084 (1998)
72. Segalowitz, S.J. Language functions and brain organization, Academic Press, 1983
73. Segev, R. and E.Ben-Jacob, From neurons to brain: Adaptive self-wiring of neurons, TR /98 Faculty of Exact Sciences, Tel-Aviv University (1998)
74. Selverston, A. (ed) Model neural networks and behaviour, Plenum Press, 1985
75. Sinclair, S., and C. Watson, "The Development of the Otago Speech Database", In Kasabov, N. and Coghill, G. (Eds.), Proceedings of ANNES '95, Los Alamitos, CA, IEEE Computer Society Press (1995).
76. Towel, G., J. Shavlik, and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks", Proc. of the 8th National Conf. on Artificial Intelligence AAAI'90, Morgan Kaufmann, 861-866 (1990).
77. Thrun, S. and Mitchel, T.M. "Integrating inductive neural network learning and explanation-based learning", in: R.Bajcsy (ed) Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan-Kaufmann (1993)
78. Topchy, A., O.Lebedko, V.Miagkikh and N.Kasabov, "Adaptive training of radial basis function networks based on co-operative evolution and evolutionary programming", in: Progress in connectionist-based information systems, N.Kasabov et al (eds), Springer, 253-258 (1998)
79. Watts, M., and N. Kasabov, "Genetic algorithms for the design of fuzzy neural networks", in Proc. of ICONIP'98, Kitakyushu, Oct. 1998, IOS Press, 793-796 (1998)
80. Widrow, B., and Hoff, M. Adaptive switching circuits, In 1960 IRE WESCON Convention record, pp.96-104, IRE, New York (1960)

81. Woldrige, M., and N. Jennings, “Intelligent agents: Theory and practice”, The Knowledge Engineering review (10) 1995.
82. Wong, R.O. “Use, disuse, and growth of the brain”, Proc. Nat. Acad. Sci. USA, 92 (6) 1797-99, (1995).
83. 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).
84. Wang, J.H. and W.D.Sun, “On-line learning vector quantization: a harmonic competition approach based on conservation network”, *IEEE Trans. Systems, Man, and Cybernetics*, vol.29, part B, No.5, 642-653 (1999)

Fig.1

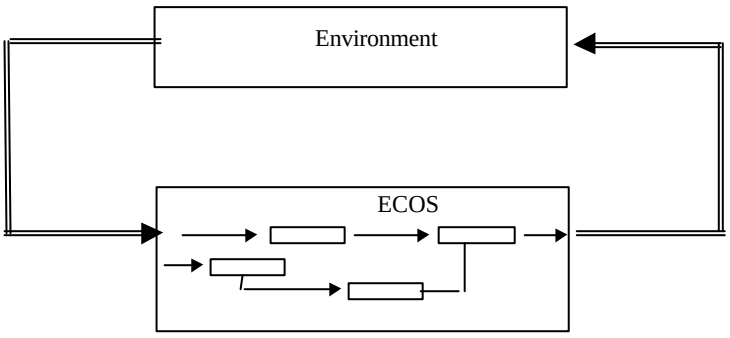


Fig.2

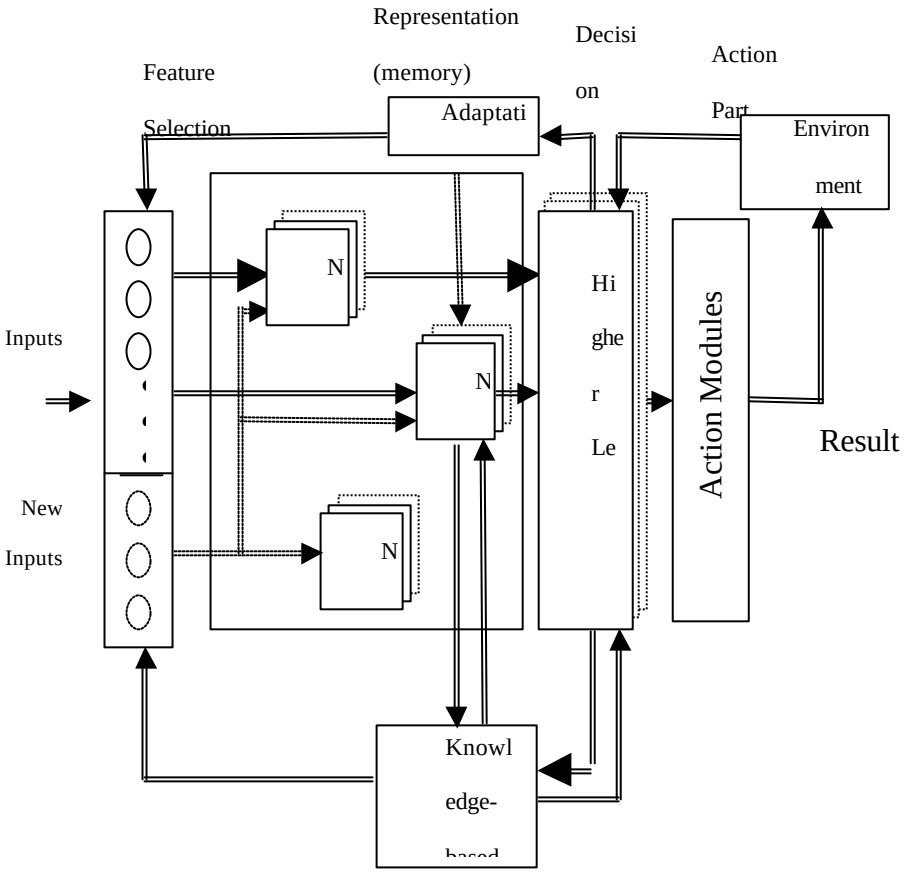

Representation
(memory)
Decisi
on
Action
Part
Feature
Selection
Adaptati
Environ
ment
Inputs
N
N
N
Hi
ghe
r
Le
Action Modules
Result
New
Inputs
Knowl
edge-

Fig.3a.

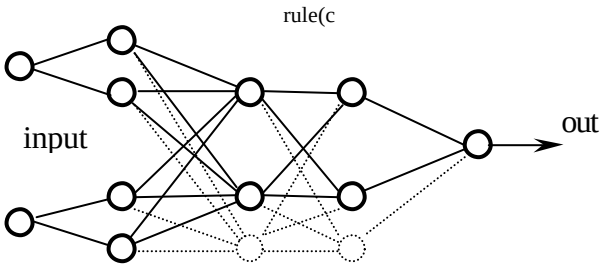

rule(c
input
out

Fig.3 b

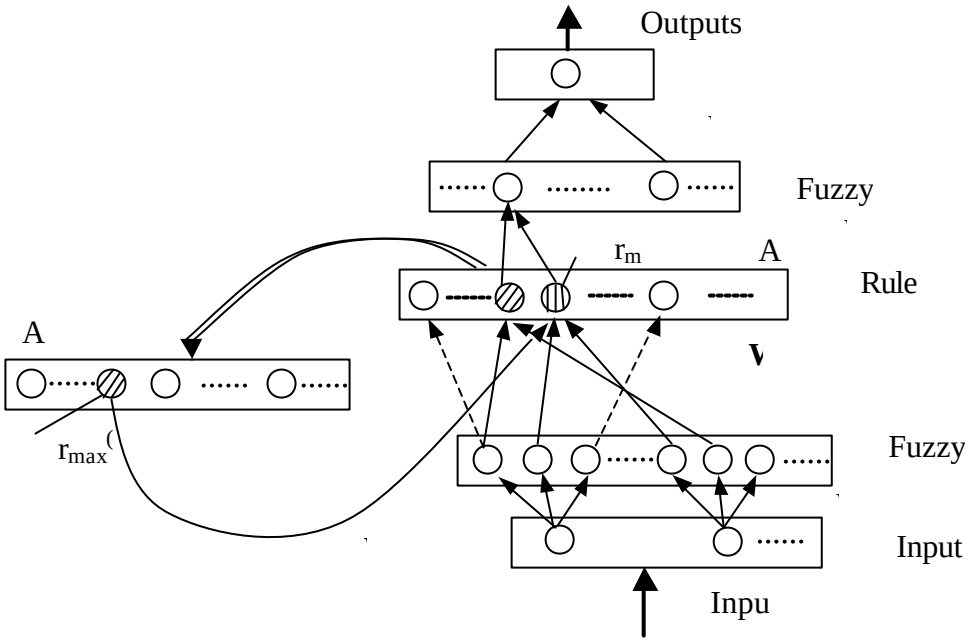
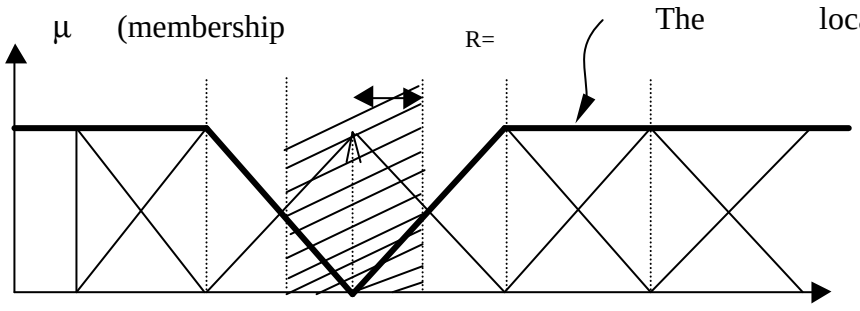

Outputs
Fuzzy
r_m
A
Rule
A
r_{max}(
Fuzzy
Input
Inpu

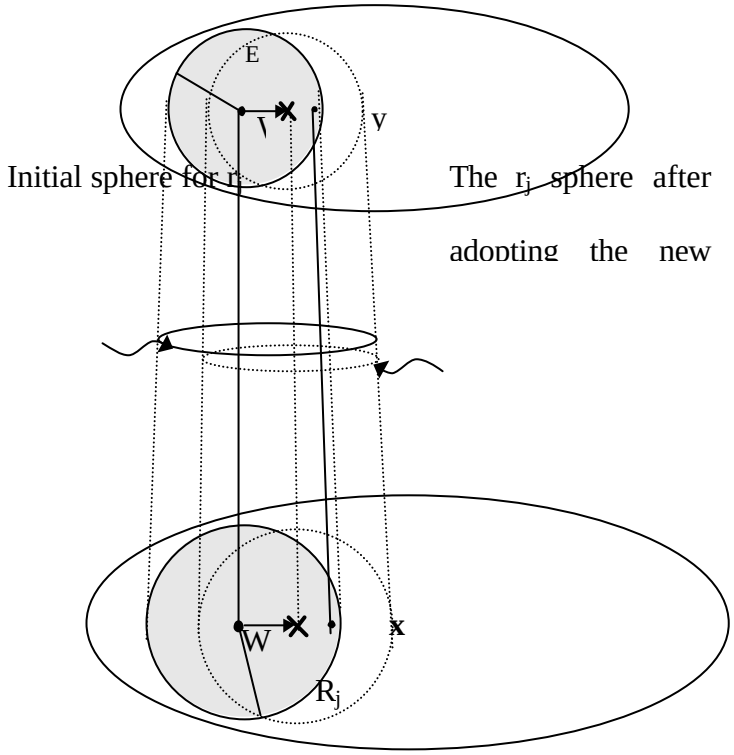
Fig.4

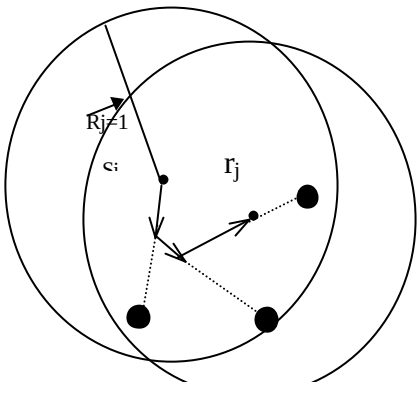


$\mathbf{d_{1f}}$ = (0, 0, 1, 0,

$D(d_1,d_{2)} = D(d_1,d_3) = D(d_1,d_5)$

Fig.5 a, b, c


E
V
v
Initial sphere for r
The r_j sphere after adopting the new
W
x
R_j


$R_j=1$
r_j

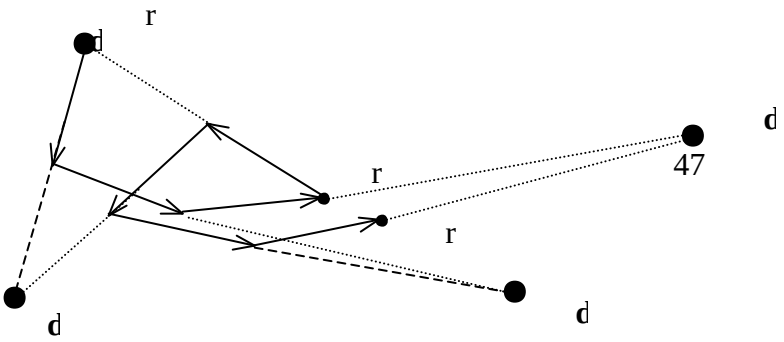

r
d
r
47
d
r
d
d

Fig.6

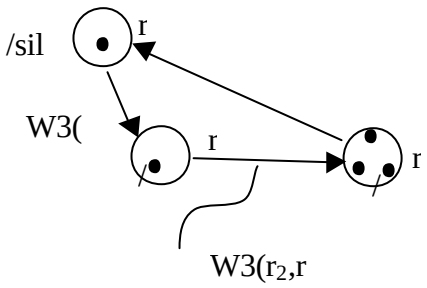


Fig.7a

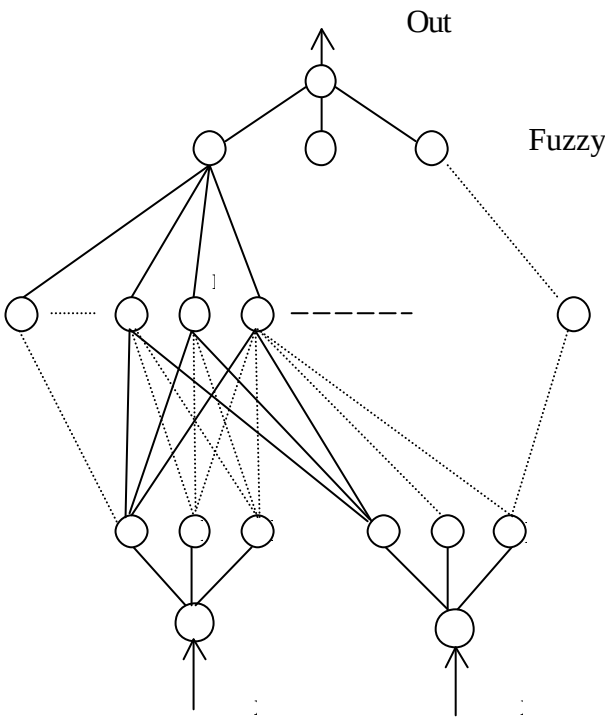


Fig.7b

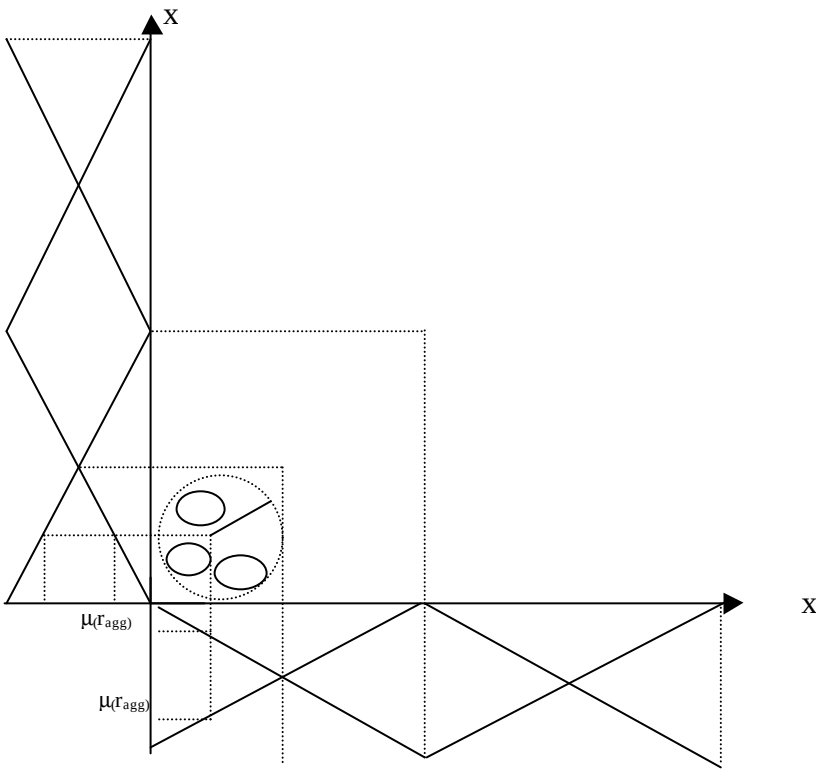


Fig.7c

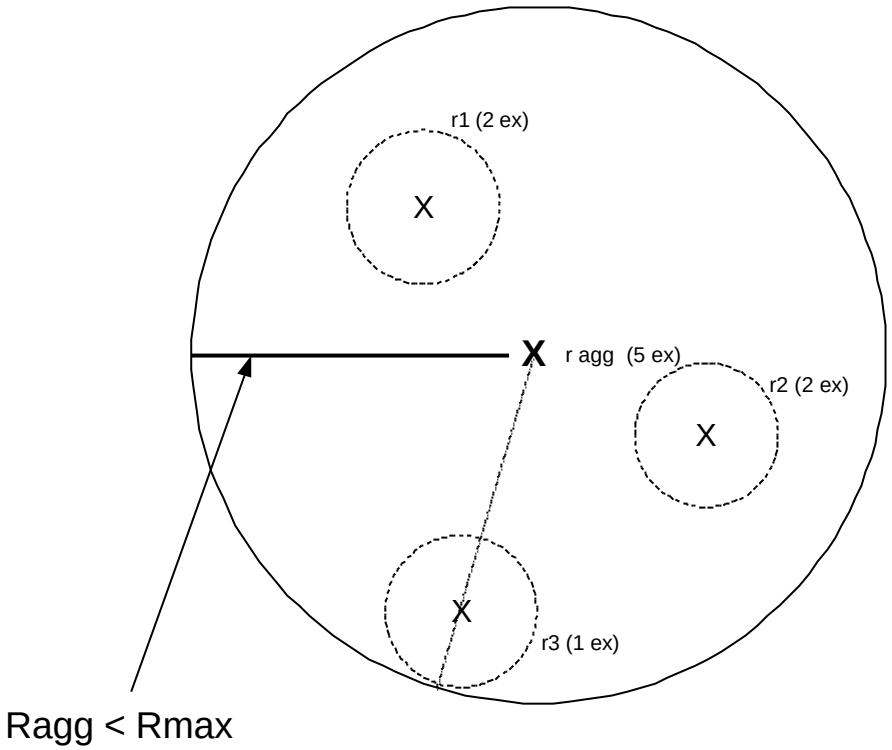


Fig8a

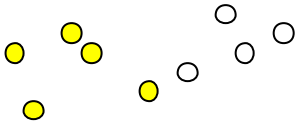


Fig.8b

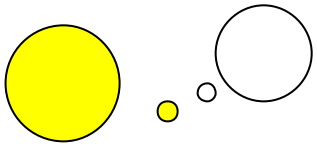
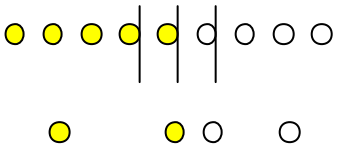


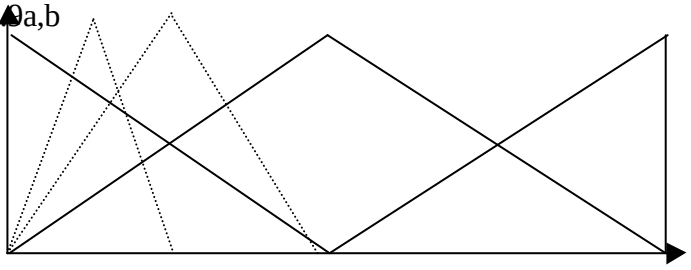
Fig.8c

Fig.8d



.

Fig 9a,b



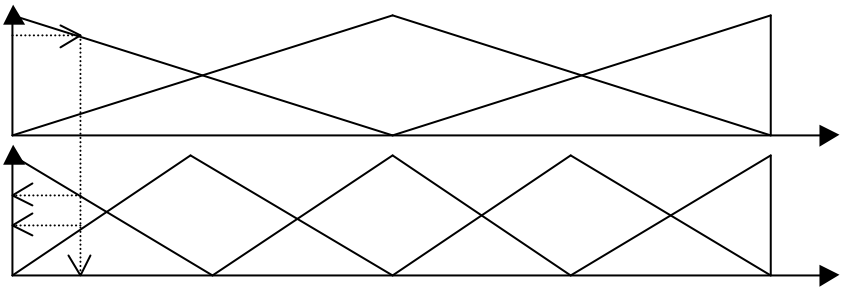
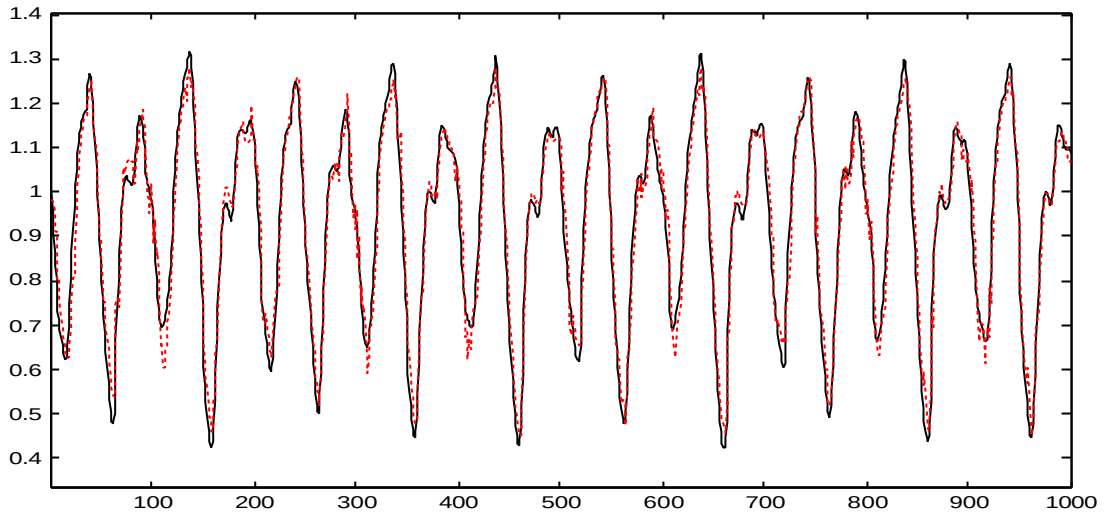


Fig.10a,b,c,d


1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
100
200
300
400
500
600
700
800
900
1000

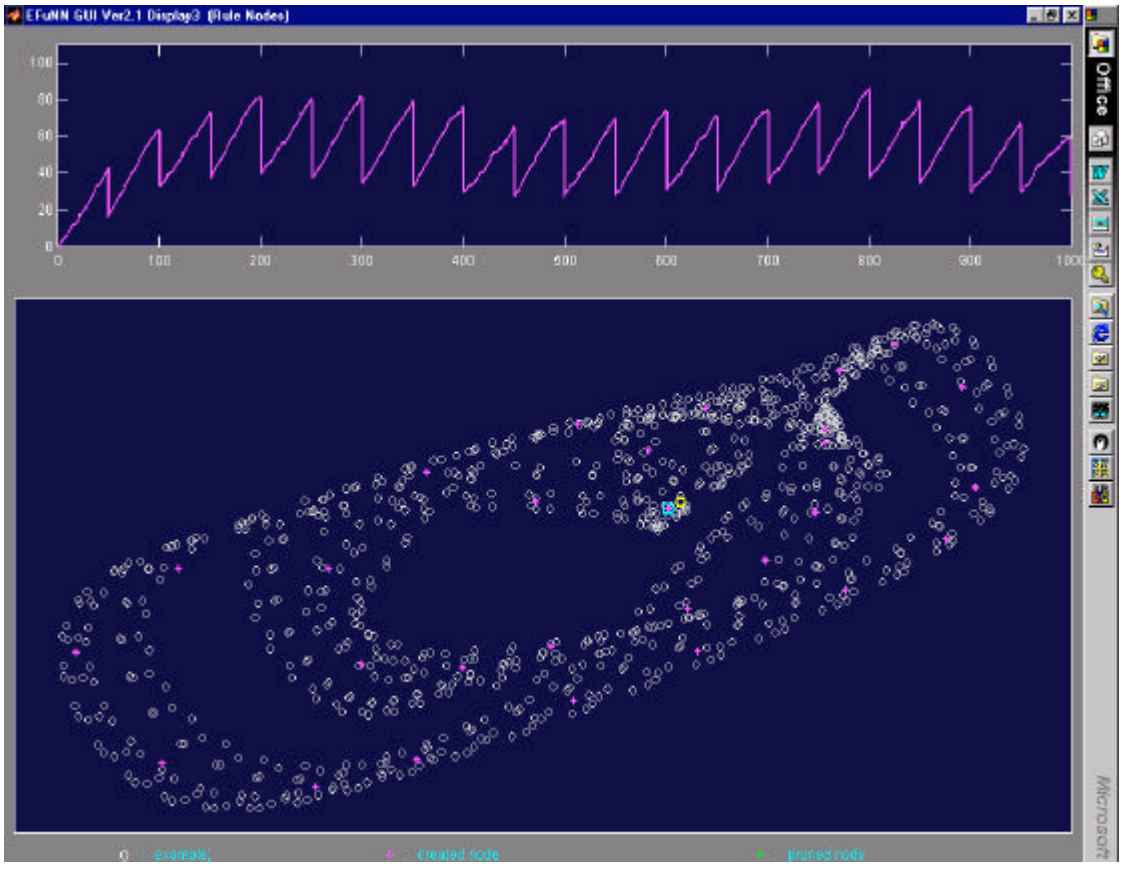

EFuNN GUI Ver2.1 Display3 (Rule Nodes)
Office
Microsoft

Fig.11a,b

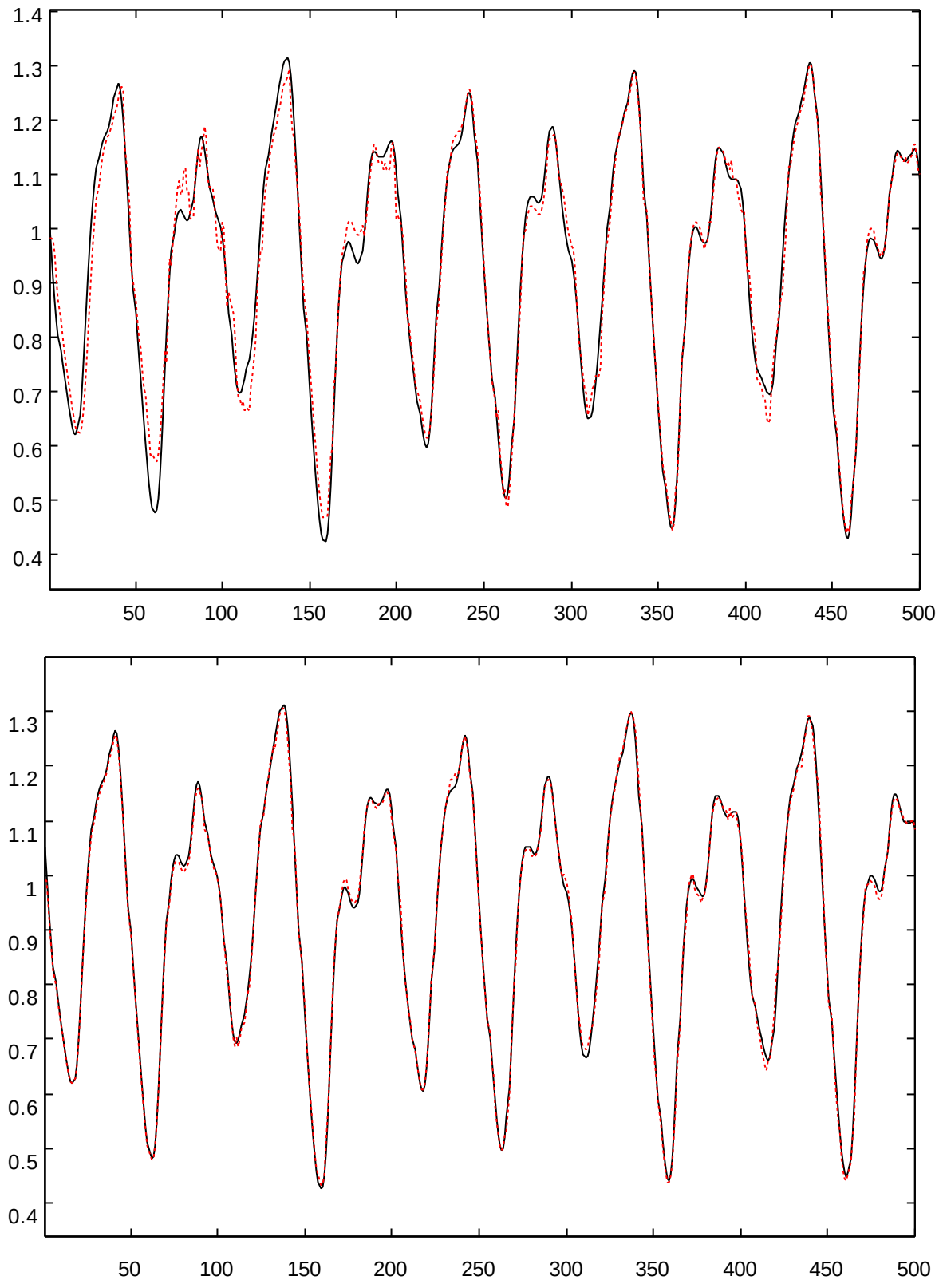
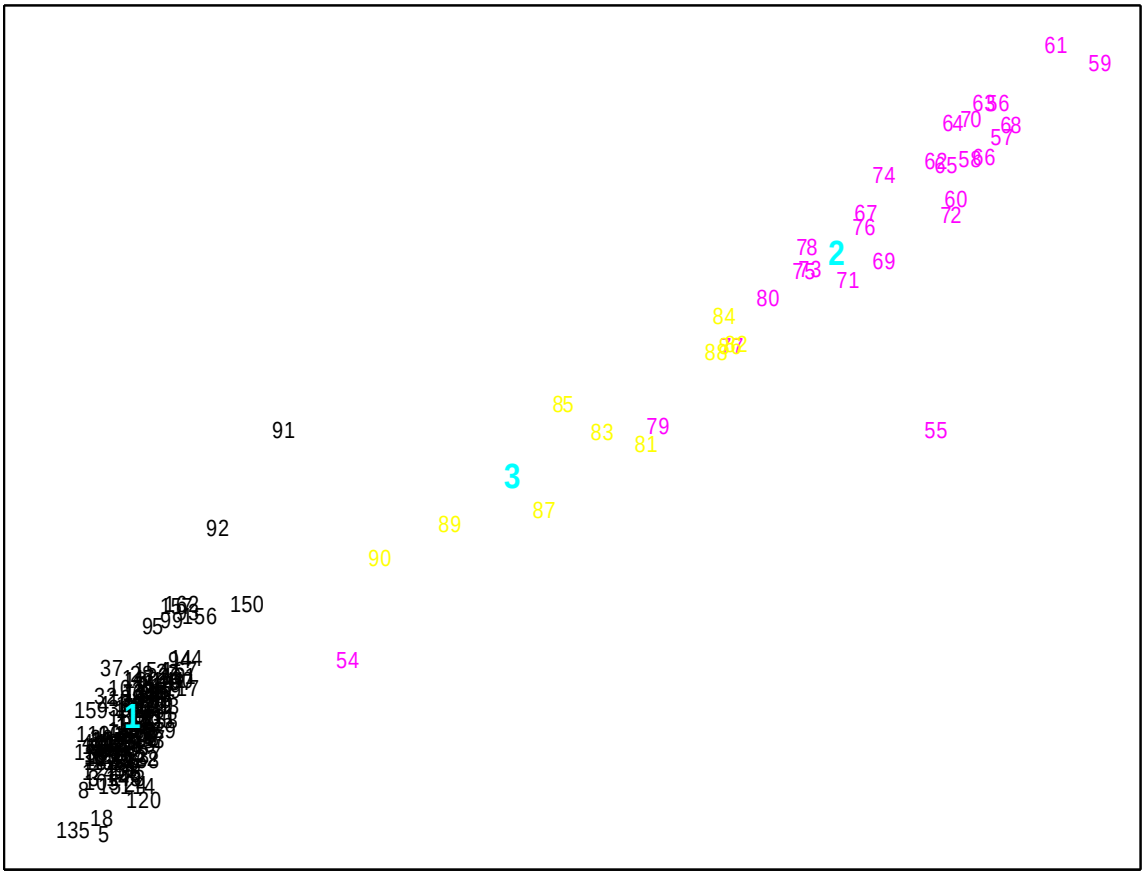

1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
50
100
150
200
250
300
350
400
450
500
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
50
100
150
200
250
300
350
400
450
500

Fig.12a,b


61
59
64
70
68
57
74
60
72
67
76
78
2
69
71
80
84
91
85
83
79
81
55
3
87
89
92
90
150
54
1
120
18
135
5

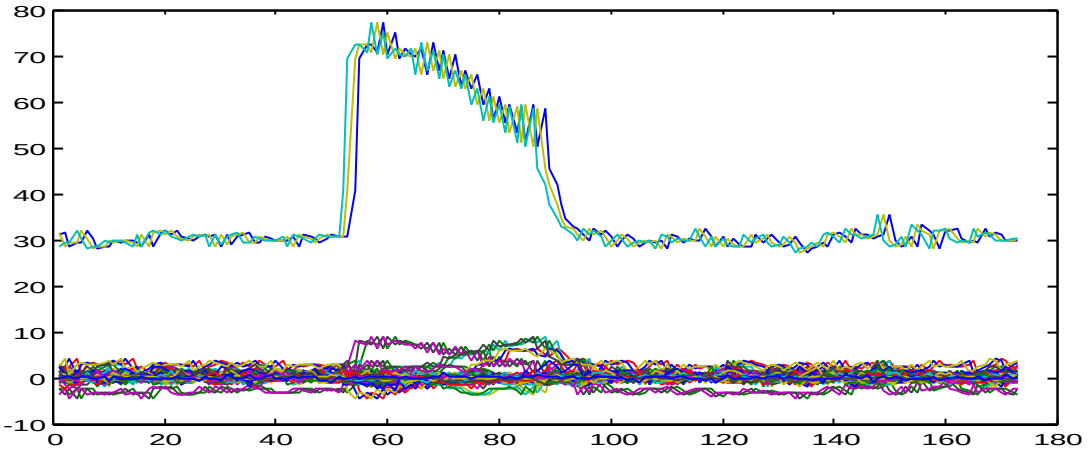

80
70
60
50
40
30
20
10
0
-10
0
20
40
60
80
100
120
140
160
180

Table 1

Rule 1: if [x1 is (3 0.658) AND [x2 is (4 0.884)] AND [x3 is (4 0.822)] AND

[x4 is (4 0.722)] [Radius of the receptive field R1 = 0.086]

then [y is (4 0.747)[accommodated training examples Nex(r1)= 6]

Rule 2: if [x1 is (3 0.511)] AND [x2 is (4 0.774)] AND [x3 is (4 0.852)] AND

[x4 is (4 0.825)] [Radius of the receptive field R2 = 0.179]

then [y is (3 0.913)][accommodated training examples Nex(r2)=2]

……

Rule 16: if [x1 is (2 0.532)]AND [x2 is (2 0.810)] AND [x3 is (3 0.783)] AND

[x4 is(4 0.928)] [Radius of the receptive field R16 = 0.073)

then [y is (5 0.516)] [accommodated training examples Nex(r16)=12]

Notation: The fuzzy values are denoted with numbers as follows: 1- very small, 2- small, 3-medium, 4 – large, 5 – very large; the antecedent and the consequent weights are rounded to the third digit after the decimal point; smaller values than 0.5 are ignored as 0.5 is used as a threshold T1=T2 for rule extraction)

Table 2.

Model	Parameter values	**Number of centers** (rule nodes in EFuNN)	**On-line LNDEI after learning 3000 examples**
RAN [61]	ε = 0.02	113	0.373
RAN-GRD [66]	ε = 0.01	50	0.165
RAN—P-GQRD [66]	ε = 0.02	31	0.160
EFuNN-su	E=0.05,Rmax=0.2	91	0.115

EFuNN-dp	E=0.05;Rmax=0.2	93	0.113
EFuNN-gd	E= 0.05;Rmax= 0.2	102	0.103

Table 3.

Method \ Test classification accuracy	**Size of the model**	**Number of training iterations**	**Global test classification accuracy (off-line)**	**Local test classification accuracy (on-line)**
LVQ	386 reference vectors	500	57%	N/A
EFuNN-s/u	175 rule nodes	1	77%	86.6%

.

Captions of figures

Figure 1. ECOS learn through interaction with the environment

Fig.2. A block diagram of the ECOS framework

Fig.3. Evolving fuzzy neural network EFuNN: (a) an example of a standard feed-forward EFuNN system; (b) an example of an EFuNN with a short- term memory and a feedback connection

Fig.4. Triangular membership functions (MF) and the local, normalised, fuzzy distance measure

Fig.5. Adaptive learning in EFuNN: (a) a rule node represents an association of two hyper spheres from the fuzzy input space and the fuzzy output space; the rule node "moves" when a new fuzzy vector pair ($\mathbf{x}_f$,$\mathbf{y}_f$) is accommodated; (b) accommodating 4 points in a rule node r_j; (c) two-pass learning of four points that fall in the receptive and the reactive fields of the rule node r_j.

Fig.6. The process of creation of temporal connections from consecutive frames taken from a pronounced word "eight" data. The three rule nodes represent the three major parts of the speech signal, namely: /silence/,/ei/,/t/.

Fig.7a. Aggregation of rule nodes in an EFuNN - an example of an evolved EFuNN structure; (b) The process of aggregation of three rule nodes r_1,r_2 and r_3 into one cluster

node $r_{agg.;}$ (c) The resulting node r_{agg} from the aggregation of the three rules has a receptive field radius R_{agg} less than a pre-defined (as a system parameter) value R_{max}.

Figure 8. Aggregation of rule nodes with the use of "guard" nodes. Rule nodes (presented as circles, the radius of which defining their receptive fields) are projected in the input space with the class that the nodes support denoted as the color of the circle: (a) before aggregation; (b) after aggregation – note that the receptive field of the new rule nodes have changed, but the receptive fields of the unchanged nodes, the 'guard' nodes, are unchanged; (c) and (d) – the process of aggregation as in (a) and (b) but here presented in one-dimensional space of the ordered rule nodes where spatial allocation of nodes is applied.

Fig.9. On-line membership function modification: (a) new MF are inserted without modifying the existing ones; (b) 5 new MF are created that substitute the old 3 MF.

Fig.10a,b,c,d. Experiments for on-line evolving of an EFuNN from the Mackey Glass chaotic time-series data. An EFuNN is evolved on 1,000 data examples from the Mackey-Glass time series (4 inputs: x(t), x(t-6), x(t-12) and x(t-18), and one output x(t+6), from the CMU data: http://legend.gwydion.cs.cmu/neural-bench/benchmarks/mackey-glass.html). (a) the desired versus the predicted six steps ahead value through one-pass on-line learning; (b) the absolute, the local on-line RMSE and the local on-line NDEI errors over time; (c) the process of creation and aggregation of rule nodes over time; (d) the input data vectors (circles) and the rule node co-ordinates (W1 connection weights) (crosses) projected in the two dimensional input space of the first two input variables x(t) and x(t-6). Some of the extracted rules are shown in tabl.1.

Fig.11. (a) An EFuNN is evolved on 500 data examples from the Mackey-Glass time series (4 inputs: x(t), x(t-6), x(t-12) and x(t-18), and one output x(t+6); the figure shows the desired versus the predicted on–line values of the time series; (b) after the EFuNN is evolved, it is tested for a global generalisation on 500 future data; the figure shows the desired versus the predicted by the EFuNN values in an off-line, batch mode.

Fig.12. Experimental results with the unsupervised learning EFuNN method on a single pronunciation data of the word "eight". From top to bottom: (a) the two dimensional input space of the first two mel scale coefficients of all data frames (each of them is a 78 element vector representing 26 mel scale coefficient in three time lags of 12 ms each with an overlap of 50%); each vector is represented as the consecutive number of the corresponding frame from the speech sequence; the evolved rule nodes after aggregation are denoted with their numbers (larger font); the emerged rule nodes 1,2, and 3 represent each of the three phases of the signal: /silence/, /ei/, /t/, /silence/; the data frames are numbered in the order they are presented to the EFuNNun that shows the time of the frame; (b) all the 78 element mel-vectors taken from the speech signal of the pronounced word "eight" over time.

Captions of tables

Table 1. Some of the fuzzy rules extracted from the evolved from the Mackey-Glass data EFuNN.

Table 2. Comparative analysis of different on-line learning models on the Mackey-Glass time series. Each model is evolved on 3,000 data examples from the Mackey-Glass time series (4 inputs: x(t), x(t-6), x(t-12) and x(t-18), and one output x(t+85), from the CMU data: http://legend.gwydion.cs.cmu/neural-bench/benchmarks/mackey-glass.html).

Table 3. The global test accuracy of an LVQ and an EFuNN models applied to the recognition of spoken English digit words, and the local test accuracy for EFuNN (similar to the local RMSE). The LVQ model has the following parameter values: code-book vectors – 396, training iterations – 500, while the EFuNN evolved 157 rule nodes and used 1 iteration for each data example.