Evolving Connectionist Systems for On-line, Knowledge-based Learning: Principles and Applications [1]

Nikola Kasabov
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 connectionist systems (ECOS) as an effective approach to building on-line, adaptive intelligent systems. ECOS evolve 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. The ECOS framework is presented and illustrated on a particular type of evolving neural networks - evolving fuzzy neural networks (EFuNNs). EFuNNs can learn spatial-temporal sequences in an adaptive way, through one pass learning. Rules can be inserted and extracted at any time of the system operation. The characteristics of ECOS and EFuNNs are illustrated on several case studies that include: adaptive pattern classification; adaptive, phoneme-based spoken language recognition; adaptive dynamic time-series prediction; intelligent agents.
Key words: evolving connectionist systems; evolving fuzzy neural networks; on-line learning; spatial-temporal adaptation; adaptive speech recognition.

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. 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. Seven major requirements of the present IS (that are addressed in the ECOS framework presented later) are listed below [2,35,36,38]:
(1) IS should *learn fast* from a large amount of data (using fast training, e.g. one-pass training).
(2) IS should be able to *adapt incrementally* in both real time, and in an on-line mode, where new data is accommodated as they become available. The system should tolerate and accommodate imprecise and uncertain facts or knowledge and refine its knowledge.
(3) IS should have an *open structure* where new features (relevant to the task) can be introduced at a later stage of the system's operation. IS should dynamically create new modules, new inputs and outputs, new connections and nodes. That should occur either in a supervised, or in an unsupervised mode, using one modality or another, accommodating data, heuristic rules, text, images, etc.
(4) IS should be *memory-based,* i.e. they should keep a reasonable track of information that has been used in the past and be able to retrieve some of it for the purpose of inner refinement, or for answering an external query.
(5) IS should improve continuously (possibly in a life–long mode) through active *interaction* with other IS and with the environment they operate in.

[1] **Technical Report 99/02** Department of Information Science, University of Otago, P.O.Box 56, Dunedin, New Zealand. Submitted to IEEE Systems, Man, and Cybernetics, 8 March 1999

(6) IS should be able to *analyse themselves* in terms of behaviour, global error and success; to explain what has been learned; to make decisions about its own improvement; to manifest introspection.
(7) IS should adequately represent *space and time* in their different scales; should have parameters to represent such concepts as spatial distance, short-term and long-term memory, age, forgetting, etc.

Unless the above seven issues are addressed in the current and the future IS, it is unlikely that a significant progress is made in areas, such as adaptive speech recognition and language acquisition, adaptive intelligent prediction and control systems, intelligent agent systems, mobile robots, visual monitoring systems, multi-modal information processing, and many more.

Several investigations [18,28,43,55,65,66,67,69,74] proved that the most popular neural network models and algorithms are not suitable for adaptive, on-line learning, that includes multilayer perceptrons trained with the backpropagation algorithm, radial basis function networks [58], self-organising maps SOMs [47,48] and these NN models were not designed for on-line learning in the first instance. At same time some of the seven issues above have been acknowledged and addressed in the development of several NN models for adaptive learning and for structure and knowledge manipulation as discussed below.

Adaptive learning is aiming at solving the well-known stability/plasticity dilemma [3,4,7,8,9,13,47,48]. Several methods for adaptive learning are related to the work presented here, namely incremental learning, lifelong learning, on-line learning.

Incremental learning is the ability of a NN to learn new data without destroying (or at least fully destroying) the learned patterns from old data, and without a need to be trained on the whole old and new data. Significant progress in incremental learning has been achieved due to the Adaptive Resonance Theory (ART) [7,8,9] and its various models, that include unsupervised models (ART1, ART2, FuzzyART) and supervised versions (ARTMAP, Fuzzy ARTMAP- FAM). Lifelong learning is concerned with the ability of a system to learn during its entire existence in a changing environment [82, 69,35,36]. Growing, as well as pruning operation, are involved in the learning process. 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. NN models for on-line learning are introduced and studied in [1,2,4,7,11,17,22,28,31,35,36,42,44,46,53,69].

The issue of NN structure, the bias/variance dilemma, has been acknowledged by several authors [6,7,13,65,68]. The dilemma is concerned with the situation where if the structure of a NN is too small, the NN is biased to certain patterns, and if the NN structure is too large there are too many variances that result in over-training, and poor generalisation, etc. In order to avoid this problem, a NN (or an IS) structure should dynamically adjust during the learning process to better represent the patterns in the data from a changing environment. Three approaches have been taken so far for the purpose of creating dynamic IS structures: constructivism, selectivism, and a hybrid approach.

Constructivism is concerned with developing NNs that have a simple initial structure and grow during its operation through insertion of new nodes and new connections when new data items arrive. This approach can also be implemented with the use of an initial set of neurons that are sparsely connected and that become more and more wired with the incoming data [62,73,15,19]. The latter implementation is supported by biological facts [62,73,77]. Node insertion can be controlled by either a similarity measure, or by the output error measure, or by both. There are other methods that insert nodes based on the evaluation of the local error, e.g. the Growing Cell Structure, Growing Neural Gas, Dynamic Cell Structure [19,11,13]. Other methods insert nodes based on a global error evaluation of the performance of the whole NN. Such method is the Cascade-Correlation [15]. Methods that use both similarity and output error for node insertion are used in Fuzzy ARTMAP [9]. Cellular automata systems have also been used to implement the constructivist approach [11,4]. These systems grow by creating connections between neighbouring cells in a regular cellular structure. Simple rules, embodied in the cells, are used to achieve the growing effect. Unfortunately in most of the implementations the rules for growing do not change during the evolving process. This limits the adaptation of the growing structure. The brain-building system is an example of this class [11].

Selectivism is concerned with pruning unnecessary connections in a NN that starts its learning with many, in most cases redundant, connections [26,29, 49,56,59,64]. Pruning connections that do not contribute to the performance of the system can be done by using several methods, e.g.: optimal-brain damage [50]; optimal brain surgeon [26]; structural learning with forgetting [29,49]; training-and-zeroing [32]; regular pruning [56].

Genetic algorithms (GA) and other evolutionary computation techniques that constitute a heuristic search technique for finding the optimal, or near optimal solution from a solution space, have also been widely applied for optimising a NN structure [20,23,13,39,40,71,79,80]. Unfortunately, most of the evolutionary computation methods developed so far assume that the solution space is compact and bounded, i.e. the evolution takes place within a pre-defined problem space and not in a dynamically changing and open one, therefore not allowing for continuous, on-line adaptation. The GA implementations so far have also been very time-consuming.

Some NN models use a hybrid constructivist/selectivist approach [52,61,70]. The framework proposed here also belongs to this group.

Some of the above seven issues have also been addressed in the knowledge-based neural networks (KBNN) [24,33,38, 63,76,83] as knowledge is the essence of what an IS system has learned. KBNN have operations to deal with both data and knowledge, that include learning from data, rule insertion, rule extraction, adaptation and reasoning. KBNN have been developed mainly as a combination of symbolic AI systems and NN [24, 30,76], or as a combination of fuzzy logic systems and NN [25,30,33,38,39,44,45,51,63,83], or as a combination of a statistical technique and NN [2,4,12,57].

It is clear that in order to fulfil the seven major requirements of the current IS, radically different methods and systems are essential in both learning algorithms and structure development. A framework called ECOS (Evolving COnnectionist Systems) that addresses all seven issues above is introduced in the paper, along with a method of training called ECO training. The major principles of ECOS are presented in section 2. The principles of ECOS are applied in section 3 to develop evolving fuzzy neural network model called EFuNN. Several learning strategies of ECOS and EFuNNs are introduced in section 3. In section 4 ECOS and EFuNNs are illustrated on several case study problems of adaptive phoneme recognition, dynamic time series prediction, and intelligent agents. Section 5 suggests directions for further development of ECOS.

2. The ECOS framework

Evolving connectionist systems (ECOS) are systems that evolve in time through interaction with the environment. They have some (genetically) pre-defined parameters (knowledge) but they also learn and adapt as they operate. In contrast with the evolutionary systems they do not necessarily create copies of individuals and select the best ones for the future. They emerge, evolve, develop, unfold through innateness and learning, and through changing their structure in order to better represent data [14,31,35,36]. ECOS learn in an on-line and a knowledge–based mode, so they can accommodate any new incoming data from a data stream, and the learning process can be expressed as a process of rule manipulation.

A block diagram of the ECOS framework is given in fig.1. ECOS are multi-level, multi-modular structures where many neural network modules (denoted as NNM) are connected with inter-, and intra- connections. ECOS do not have a clear multi-layer structure, but rather a modular, “open” structure.

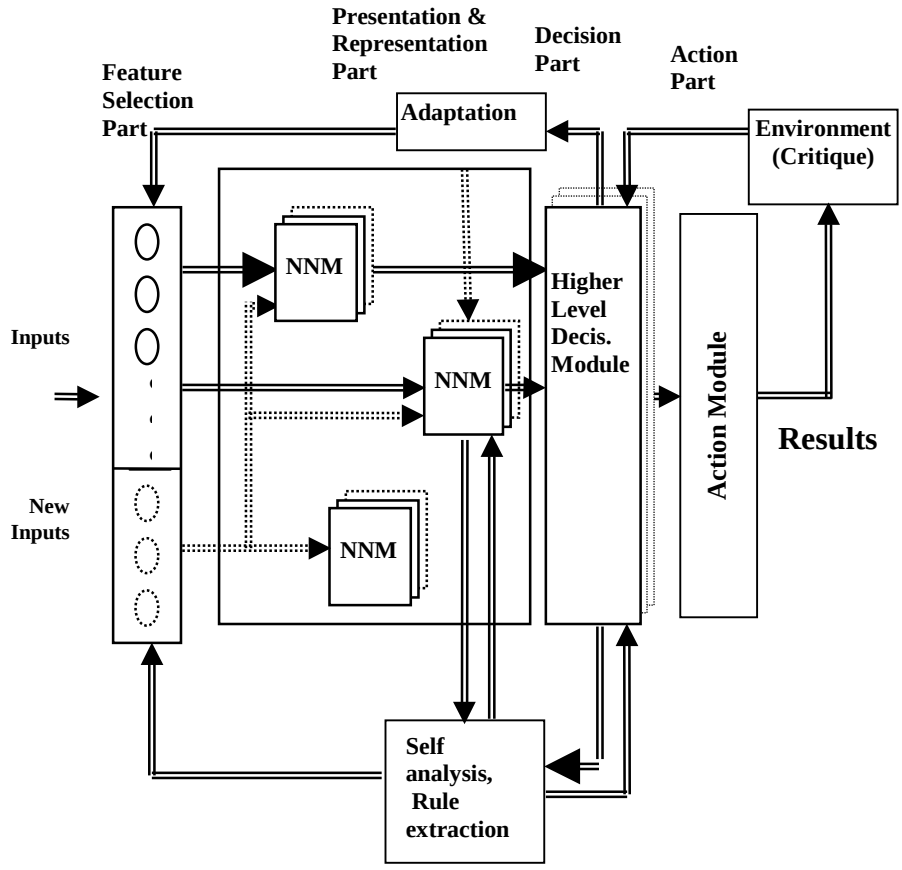


Fig.1 Block diagram of the ECOS framework.

The main parts of ECOS are described below.
(1) Feature selection part. It performs filtering of the input information, feature extraction and forming the input vectors. The number of inputs (features) can vary from example to example from the input data stream fed to the ECOS.
(2) Presentation and representation (memory) part, where information (patterns) are stored. It is a multi-modular, evolving structure of NNM organised in spatially distributed groups; for example one module can represent the phonemes in a spoken language (one NN representing one class phoneme).
(3) Higher-level decision part that consists of several modules, each taking decision on a particular problem (e.g., phoneme, word, concept). The modules receive feedback from the environment and make decisions about the functioning and the adaptation of the whole ECOS.
(4) Action modules, that take the output from the decision modules and pass output information to the environment.
(5) Self-analysis, and rule extraction modules. This part extracts compressed abstract information from the representation modules and from the decision modules in different forms of rules, abstract associations, etc.

Initially an ECOS has a pre-defined structure of some NNMs, each of them being a mesh of nodes (neurons) and very few connections defined through prior knowledge, or "genetic" information. Gradually, the system becomes more and more "wired" through self-organisation, and through creation of new NNM and new connections.

The ECOS functioning is based on the following *general principles*:
(1) ECOS evolve incrementally in an on-line, hybrid, adaptive *supervised/unsupervised mode* through accommodating more and more examples when they become known from a continuous

input data stream. During the operation of ECOS the higher-level decision module may activate an adaptation process through the adaptation module.
(2) ECOS are memory-based and store exemplars (prototypes, rules) that represent groups of data from the data stream. New input vectors are stored in the NNMs based on their similarity to previously stored data both on the input and the desired output information. A node in an NNM is created and designated to represent an individual example if it is significantly different from the previously used examples (with a level of differentiation set through dynamic parameters). Learning is based on *locally tuned* elements from the ECOS structure thus making the learning process fast for real-time parallel implementation. Three ways to implement local learning in a connectionist structure are presented in [6,7, 47,58].
(3) There are *three levels* at which ECOS are functionally and structurally defined:
(a) Parameter (gene) level, i.e. a chromosome contains genes that represent certain parameters of the whole systems, such as: type of the structure (connections) that will be evolved; learning rate; forgetting rate; size of a NNM; NNM specialisation, thresholds that define similarity; error rate that is tolerated, and many more. The values of the genes are relatively stable, but can be changed through genetic operations, such as mutation of a gene, deletion and insertion of genes that are triggered by the self analysis module as a result of the overall performance of the ECOS.
(b) Representation (synaptic) level, that is the information contained in the connections of the NNM. This is the long-term memory of the system where exemplars of data are stored. They can be either retrieved to answer an external query, or can be used for internal ECOS refinement.
(c) Behavioural (neuronal activation) level, that is the short-term activation patterns triggered by input stimuli. This level defines how well the system is functioning in the end.
(4) ECOS evolve through *learning (growing), forgetting (pruning), and aggregation*, that are both defined at a genetic level and adapted during the learning process. ECOS allow for: creating/connecting neurons; removing neurons and their corresponding connections that are not actively involved in the functioning of the system thus making space for new input patterns to be learned; aggregating nodes into bigger-cluster nodes.
(5) There are two global modes of learning in ECOS:
(a) Active learning - learning is performed when a stimulus (input pattern) is presented and kept active.
(b) Passive (inner, ECO) learning mode - learning is performed when there is no input pattern presented to the ECOS. In this case the process of further elaboration of the connections in ECOS is done in a passive learning phase, when existing connections, that store previously fed input patterns, are used as "echo" (here denoted as ECO) to reiterate the learning process (see for example fig.9 explained later).

There are two types of ECO training:

- *cascade eco*-training: a new connectionist structure (a NN) is created in an on-line mode when conceptually new data (e.g., a new class data) is presented. The NN is trained on the positive examples of this class, on the negative examples from the following incoming data, and on the negative examples from previously stored patterns in previously created modules.
- *'sleep' eco*-training: NNs are created with the use of only partial information from the input stream (e.g., positive class examples only). Then the NNs are trained and refined on the stored patterns (exemplars) in other NNs and NNMs (e.g., as negative class examples).

(6) ECOS provide explanation information extracted from the NNMs through the self-analysis/ rule extraction module. Generally speaking, ECOS learn and store knowledge, rules, rather than individual examples or meaningless numbers.
(7) The ECOS principles above are based on some biological facts and biological principles (see for example [31,55,62,68,72,82]).

Implementing the ECOS framework and the NNM from it requires connectionist models that comply with the ECOS principles. One of them, called evolving fuzzy neural network (EFuNN) is presented in the next section.

3. Evolving Fuzzy Neural Networks EFuNNs

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]. EFuNNs are FuNNs that evolve according to the ECOS principles. EFuNNs were introduced in [31,35,36] where preliminary results were given. Here EFuNNs are further developed.

EFuNNs have a five-layer structure, similar to the structure of FuNNs (fig.2a). 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.2b). The layer of feedback connections could be used if temporal relationships between input data are to be memorised structurally.

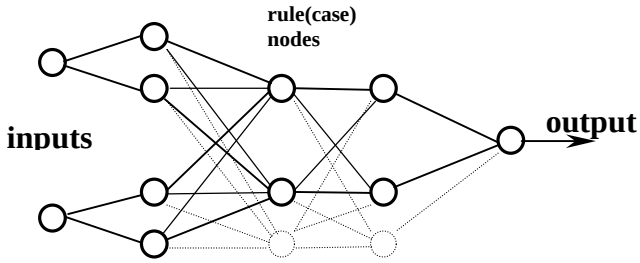


Fig.2a The five layers basic structure of the EfuNNs.

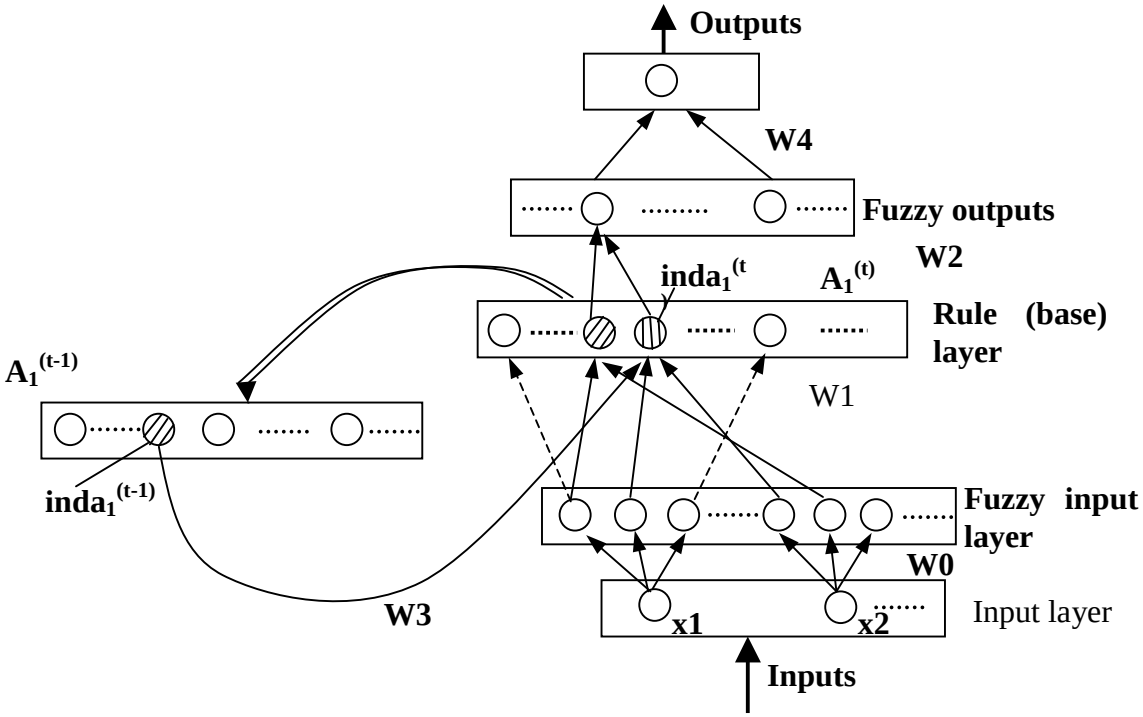


Fig2b EFuNN with a short term memory and a feedback connection

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.) (see fig.3).

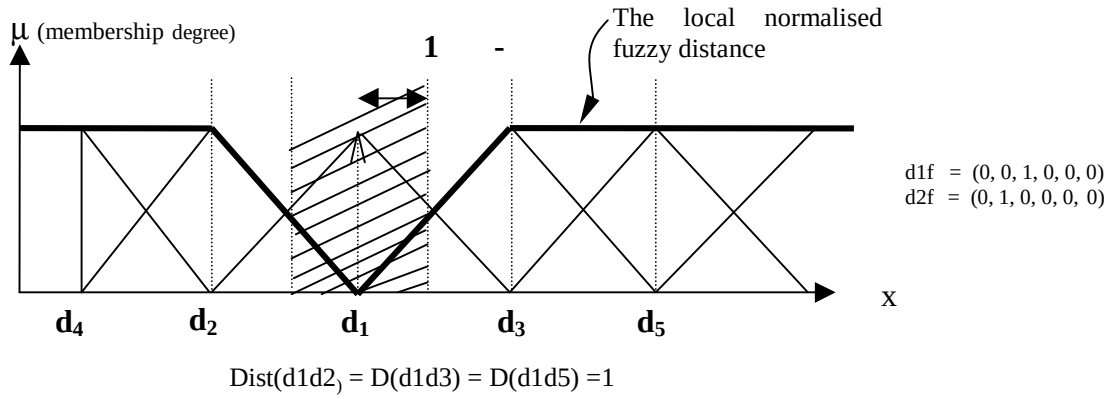


Fig.3. Membership functions (MF) and the local, normalised, fuzzy distance function

The number and the type of MF can be dynamically modified in an EFuNN which is explained later in section 3. New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN (see fig.10a,b and the explanation in section 3). 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/unsupervised learning. The rule nodes represent prototypes (exemplars, clusters) of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and 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. The fourth layer of neurons represents fuzzy quantization for the output variables, similar to the input fuzzy neurons representation. The fifth layer represents the real values for the output variables.

The evolving process can be based on 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 cases.

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.4a), 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 an input hyper-sphere of a rule node is defined as (1- Sthr), where Sthr is the sensitivity threshold parameter defining the minimum activation of a rule node (e.g., r1, previously evolved to represent a data point (Xd1,Yd1)) to an

input vector (e.g., (Xd2,Yd2)) in order for the new input vector to be associated with this rule node. Two pairs of fuzzy input-output data vectors **d1**=(Xd1,Yd1) and **d2**=(Xd2,Yd2) will be allocated to the first rule node r_1 if they fall into the r_1 input sphere and in the r_1 output sphere, i.e. the local normalised fuzzy difference between Xd1 and Xd2 is smaller than the radius *r* and the local normalised fuzzy difference between Yd1 and Yd2 is smaller than an error threshold Errthr. The local normalised fuzzy difference between two fuzzy membership vectors **d1f** and **d2f** that represent the membership degrees to which two real values d1 and d2 data belong to the pre-defined MF, are calculated as D(**d1f**,**d2f**) = sum(abs(**d1f** - **d2f**))/sum(**d1f** + **d2f**)). For example, if **d1f**=(0,0,1,0,0,0) and **d2f**=(0,1,0,0,0,0) (see fig.3), than D(d1,d2) = (1+1)/2=1 which is the maximum value for the local normalised fuzzy difference .

If data example **d1** = (Xd1,Yd1), where Xd1 and Xd2 are correspondingly the input and the output fuzzy membership degree vectors, and the data example is associated with a rule node r_1 with a centre r_1^1, than a new data point **d2**=(Xd2,Yd2), that is within the shaded area as shown in fig.3 and fig.4a, will be associated with this rule node too. Through the process of associating (learning) of new data points to a rule node, the centres of this node hyper-spheres adjust in the fuzzy input space depending on a learning rate lrn1, and in the fuzzy output space depending on a learning rate lr2, as it is shown in fig.4a on the two data points **d1** and **d2**. The adjustment of the centre r_1^1 to its new position r_1^2 can be represented mathematically by the change in the connection weights of the rule node r_1 from $W1(r_1^1)$ and $W2(r_1^1)$ to $W1(r_1^2)$ and $W2(r_1^2)$ according to the following vector operations:

$$W2(r_1^2) = W2(r_1^1) + lr2.\ Err(Yd1,Yd2).\ A1(r_1^1)$$
$$W1(r_1^2) = W1(r_1^1) + lr1.\ Ds(Xd1,Xd2)$$

where: Err(Yd1,Yd2)= Ds(Yd1,Yd2)=Yd1-Yd2 is the signed value rather than the absolute value of the fuzzy difference vector; $A1(r_1^1)$ is the activation of the rule node r_1^1 for the input vector Xd2. The learning process in the fuzzy input space is illustrated in fig.4b on four data points d1,d2,d3 and d4. Fig.4c shows how the centre of the rule node r_1 adjusts after learning each new data point when two-pass learning is applied. If lrn1=lrn2=0, once established, the centres of the rules nodes do not move. The idea of dynamic creation of new rule nodes over time for a time series data is graphically illustrated in fig.4d.

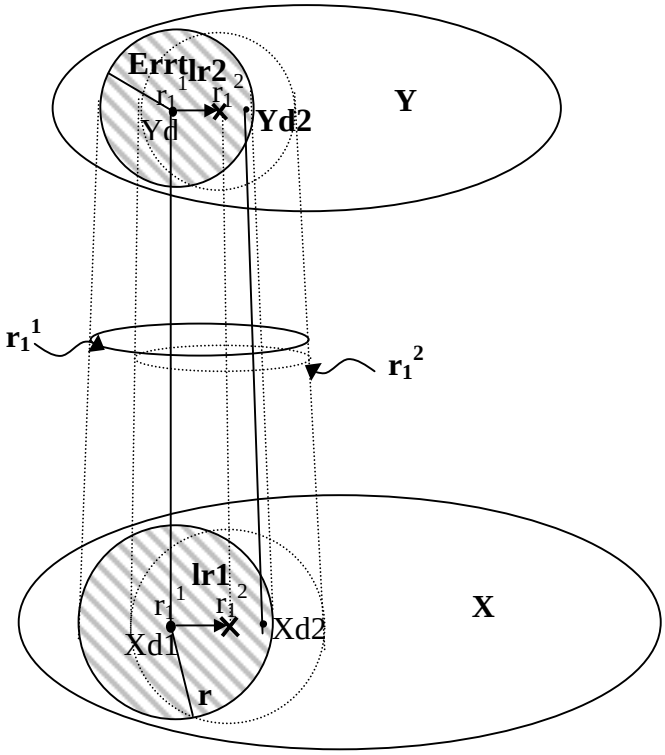


Fig4a Input / Output mapping and learning.

Fig.4b

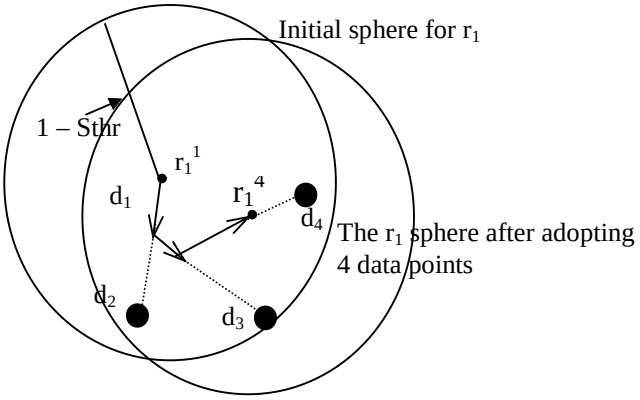
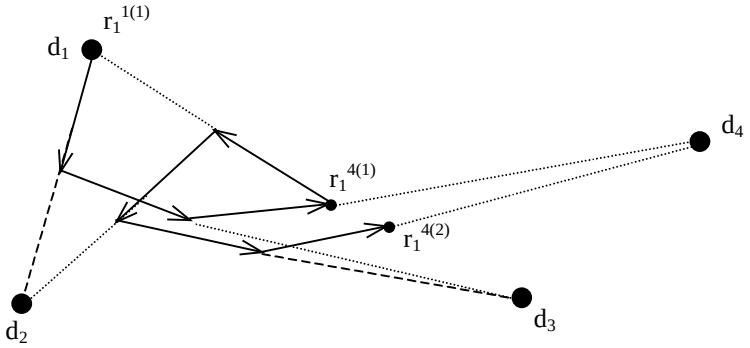


Fig.4c



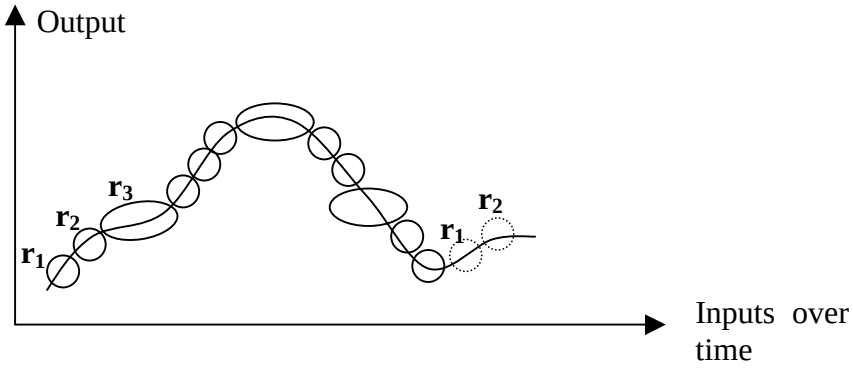


Fig.4d Dynamic Creation of new rule nodes over time.

While the connection weights from W1 and W2 capture spatial characteristics of the learned data (centres of hyper-spheres), the temporal layer of connection weights W3 from fig.2b 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 r1=inda1(t-1), and the winning node at the moment t is r2=inda1(t), then a link between the two nodes is established as follows:

$$W3(r1,r2)^{(t)} = W3(r1,r2)^{(t-1)} + lr3.\ A1(r1)^{(t-1)}\ A1(r2))^{(t)},$$

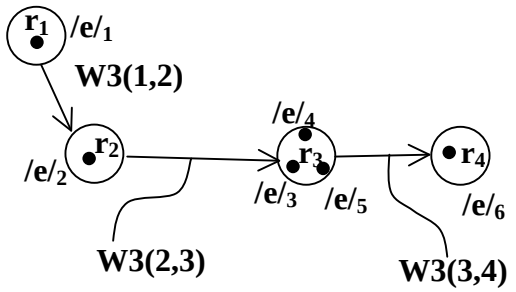
where: $A1(r)^{(t)}$ denotes the activation of a rule node r at a time moment (t); lr3 defines the degree to which the EFuNN associates links between rules (clusters, prototypes) that include consecutive data examples (if lr3=0, no temporal associations are learned in an EFuNN structure and the EFuNN from fig.2b becomes the one from fig.2a).

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 as the following vector operations:

$$A1(r) = f(Ss.\ D(r, d_{new}) + Tc.W3(r^{(t-1)}, r))$$

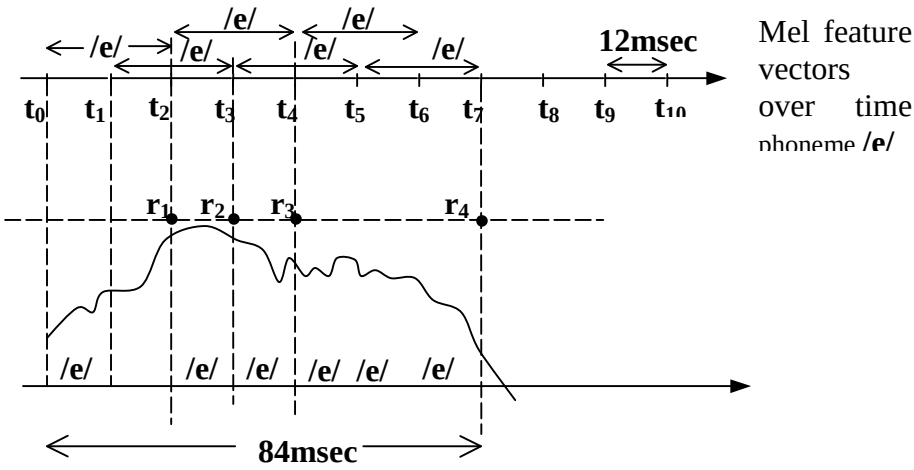
where: f is the activation function of the rule node r, $D(r, d_{new})$ is the normalised fuzzy distance value and $r^{(t-1)}$ is the winning neuron at the previous time moment.

Figures 5a,b show a schematic diagram of the process of evolving of four rule nodes and setting the temporal links between them for data taken from consecutive frames of phoneme /e/ data as discussed in section 4.



Spatial-temporal representation of phoneme /e/ data. W3 accounts for the temporal links.

Fig.5a Consecutive phoneme data frames cause creation of links between the rule nodes.



Mel feature vectors over time phoneme /e/

Fig.5b Schematic diagram of the raw phoneme data and the points in time of rule node creation.

.
Several parameters were introduced so far for the purpose of controlling the functioning of an EFuNN. Some more parameters will be introduced later, that will bring the EFuNN parameters to a comparatively large number. In order to achieve a better control of the functioning of an EFuNN structure, the three-level functional hierarchy is used here as defined in section 2 for the ECOS architecture, namely: genetic level, long-term synaptic level, and short- term activation level.

At the genetic level, all the EFuNN parameters are defined as genes in a chromosome. These are:

(a) structural parameters, e.g.: number of inputs, number of MF for each of the inputs, initial type of rule nodes, maximum number of rule nodes, number of MF for the output variables, number of outputs.

(b) functional parameters, e.g.: activation functions of the rule nodes and the fuzzy output nodes (in the experiments below saturated linear functions are used); mode of rule node activation ("one-of-n", or “many-of-n”, depending on how many activation values of rule nodes are propagated to the next level); learning rates lr1,lr2 and lr3; sensitivity threshold Sthr for the rule layer; error threshold Errthr for the output layer; forgetting rate; various pruning strategies and parameters, as explained in the EFuNN algorithm below.

3.2. The EFuNN learning algorithm

The EFuNN algorithm, to evolve EFuNNs from incoming examples, is based on the principles explained in the previous section. It is given below as a procedure of consecutive steps. Matrix operation expressions are used similar to the expressions in a matrix processing language such as MATLAB.

1. Initialise an EFuNN structure with a maximum number of neurons and no (or zero-value) connections. Initial connections may be set through inserting fuzzy rules in the structure [44]. If initially there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons, then create the first node rn=1 to represent the first example d1 and set its input W1(rn) and output W2(rn) connection weight vectors as follows:

<Create a new rule node rn>: W1(rn)=EX; W2(rn) = TE, where TE is the fuzzy output vector for the current fuzzy input vector EX.

2. WHILE <there are examples in the input stream> DO

Enter the current example (Xdi,Ydi), EX denoting its fuzzy input vector. If new variables appear in this example, which are absent in the previous examples, create new input and/or output nodes with their corresponding membership functions.

3. Find the *local normalised fuzzy distance* between the fuzzy input vector EX and the already stored patterns (prototypes, exemplars) in the rule (case) nodes rj=r1,r2,…,rn

D(EX, rj)= sum (abs (EX - W1(j))) / sum (W1(j)+EX)

4. Find the activation A1 (rj) of the rule (case) nodes rj, rj=r1:rn. Here radial basis activation function, or a saturated linear one, can be used, i.e. A1 (rj) = radbas (D(EX, rj)), or A1(rj) = satlin (1 – D(EX, rj)). The former may be appropriate for function approximation tasks, while the latter may be preferred for classification tasks. In case of the feedback variant of an EFuNN, the activation A1(rj) is calculated as:

A1 (rj) = radbas (Ss. D(EX, rj) - Tc.W3), or A1(j) = satlin (1 – Ss. D(EX, rj) + Tc.W3) .

5. Update the pruning parameter values for the rule nodes, e.g. *age*, *average activation*, as pre-defined in the EFuNN chromosome.

6. Find all case nodes rj with an activation value A1(rj) above a sensitivity threshold Sthr.

7. If there is no such case node, then <Create a new rule node> using the procedure from step 1 in an unsupervised learning mode

ELSE

8. Find the rule node inda1 that has the maximum activation value (e.g., maxa1).

9. (a) in case of "one-of-n" EFuNNs (as it is in [9,27,47]) propagate the activation maxa1 of the rule node inda1 to the fuzzy output neurons:

A2 = satlin (A1(inda1) . W2(inda1)

(b) in case of "many-of-n" mode, the activation values of all rule nodes that are above an activation threshold of Athr are propagated to the next neuronal layer (this case is not discussed in details here; it has been further developed into a new EFuNN architecture called dynamic, 'many-of-n' EFuNN, or DEFuNN [42]) .

10. Find the winning fuzzy output neuron *inda2* and its activation *maxa2*.
11. Find the desired winning fuzzy output neuron *indt2* and its value *maxt2*.
12. Calculate the fuzzy output error vector: Err=A2 - TE.
13. IF (*inda2* is different from *indt2*) or (D(A2,TE) > Errthr) <Create a new rule node>
ELSE

14. Update: (a) the input, (b) the output, an (c) the temporal connection vectors (if such exist) of the rule node *k=inda1* as follows:

(a) Ds(EX,W1(k)) =EX-W1(k); W1(k)=W1(k) + lr1.Ds(EX,W1(k)), where *lr1* is the learning rate for the first layer;

(b) W2(k) = W2 (k) + lr2. Err. maxa1, where *lr2* is the learning rate for the second layer;

(c) W3(l,k)=W3(l,k)+lr3. A1(k).A1(l) $^{(t-1)}$, here l is the winning rule neron at the previous time moment (t-1), and A1(l) $^{(t-1)}$ is its activation value kept in the short term memory.

15. Prune rule nodes j and their connections that satisfy the following fuzzy pruning rule to a pre-defined level:

IF (a rule node rj is OLD) AND (average activation A1av(rj) is LOW) and (the density of the neighbouring area of neurons is HIGH or MODERATE (i.e. there are other prototypical nodes that overlap with j in the input-output space; this condition apply only for some strategies of inseting rule nodes as explained in a sub-section below)

THEN the probability of pruning node (rj) is HIGH

The above pruning rule is fuzzy and it requires that the fuzzy concepts of OLD, HIGH, etc., are defined in advance (as part of the EFuNN's chromosome). 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 1000 examples. The use of a pruning strategy and the way the values for the pruning parameters are defined, depends on the application task.

16. Aggregate rule nodes, if necessary, into a smaller number of nodes (see the explanation in the following subsection).
17. END of the while loop and the algorithm
18. Repeat steps 2-17 for a second presentation of the same input data or for an ECO training if needed.

3.3. Strategies for locating rule nodes in the rule node space

There are different ways to locate rule nodes in an EFuNN rule node space as it is explained here. The type selected depends on the type of the problem the EFuNN is designed to solve. Here some possible strategies are explained as illustrated in fig.6:

(a) Simple consecutive allocation strategy, i.e. each newly created rule (case) node is allocated next to the previous and the following ones in a linear fashion. That represents a time order. The following statement is valid if no pruning technique is applied, but aggregation technique instead, to optimise the size of the rule layer: at least one example that was associated with rule node r_j was presented to the EFuNN before at least one example that was associated to the rule node (r_j+1) (see fig.6a).

(b) Pre-clustered location, i.e. for each output fuzzy node (e.g. NO, YES) there is a predefined location where the rule nodes supporting this predefined concept are located. At the center of this area the nodes that fully support this concept (error 0) are placed; every new rule node's location is defined based on the fuzzy output error and the similarity with other nodes (fig.6b);

(c) Nearest activated node insertion strategy, i.e. a new rule node is placed nearest to the highly activated node which activation is still less than the Sthr. A connection between the neighbouring nodes can be established similar to the temporary connections from W3.
(d) As in (c) but temporal feedback connections are set as well (see fig.2b and fig.6c). New connections are set that link consecutively activated rule nodes through using the short term memory and the links established through the W3 weight matrix; that will allow for the evolving system to repeat a sequence of data points starting from a certain point and not necessarily from the beginning.
(e) The same as above, but in addition, new connections are established between rule nodes from different EFuNN modules that become activated simultaneously (at the same time moment) (fig.6d). This would make it possible for an ECOS to learn a correlation between conceptually different variables, e.g. correlation between speech sound and lip movement.

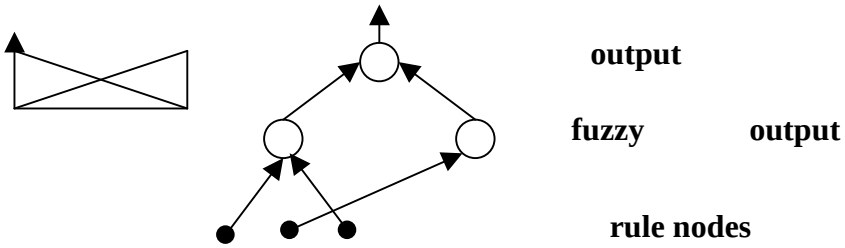


Fig.6a.

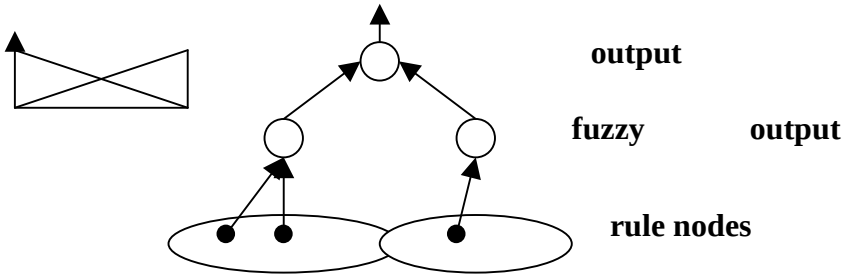


Fig.6b

Fig.6.c

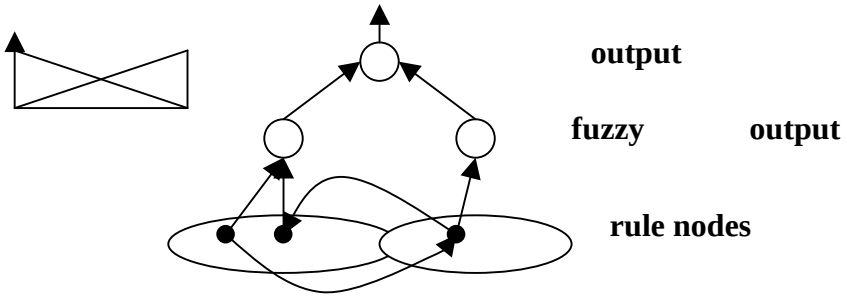


Fig.6d.

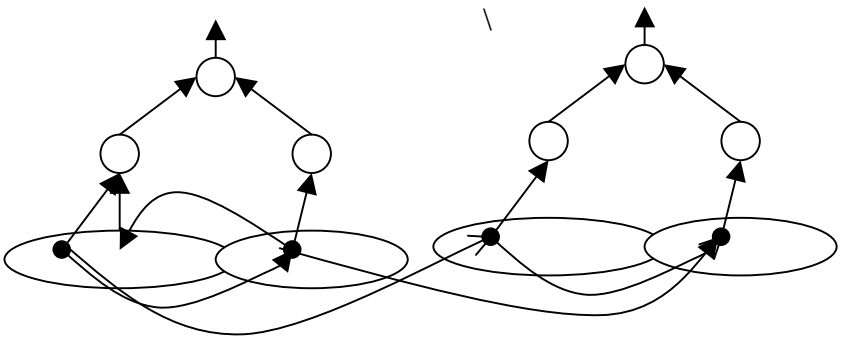


Fig.6. Different strategies for rule node insertion and connection creation

3.4 An example of using the EFuNN algorithm in an EFuNN simulator

Here, a small speech data set of 400 phoneme data examples is used to illustrate the EFuNN learning algorithm. 100 examples of each of the four phonemes /I/ (from 'sit'), /e/ (from 'get'), /ae/ (from 'cat'), and /i/ (from 'see'), which are phonemes 25,26,27 and 31 from the Otago Speech Corpus available from the WWW http://kel.otago.ac.nz/, are extracted from the speech data of two speakers of NZ English (one male and one female, numbers 17 and 21 from the Corpus). Each data example used in the experiment described below consists of 3 time lags of 26-element mel-scale vectors, each representing the speech signal within a time frame of 11.6msec, and an output label giving the phoneme class. The speech data is segmented and processed with the use of a 256-point FFT, Hamming window, overlapping of 50% between the consecutive time frames, each of them being 11.6msec long (see fig.5b).
An EFuNN with 78 inputs and 4 outputs was evolved on the 400 data examples and tested on another set. Fig. 7a shows the growth of the number of the rule nodes with the progress of entering data examples for one pass of training and the root mean square error RMSE. Fig.7b shows the activation of the /I/ output of the evolved EFuNN for the phoneme /I/ test data (the first 100 examples belong to /I/ and the rest do not belong to it). The parameter values for the EFuNN parameters (e.g. number of evolved rule nodes rn, learning rates lr1,lr2 and lr3, pruning parameters) are shown on the display of the EFuNN simulator which is available from the WWW:
http://divcom.otago.ac.nz/infosci/kel/projects/CBIIS/).

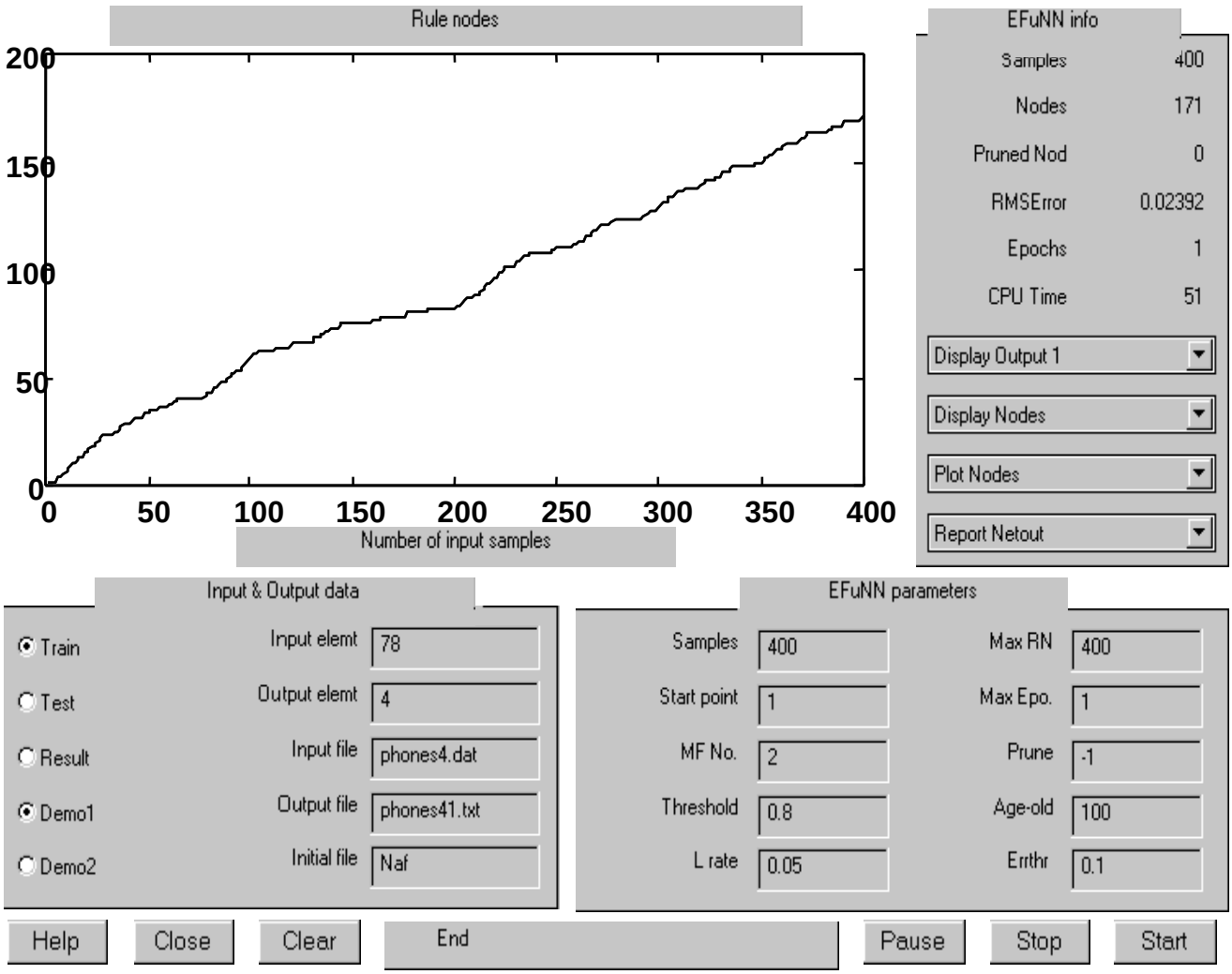


Fig. 7a shows the growth of the number of the rule nodes with the progress of entering data examples for one pass of training and the root mean square error RMSE.

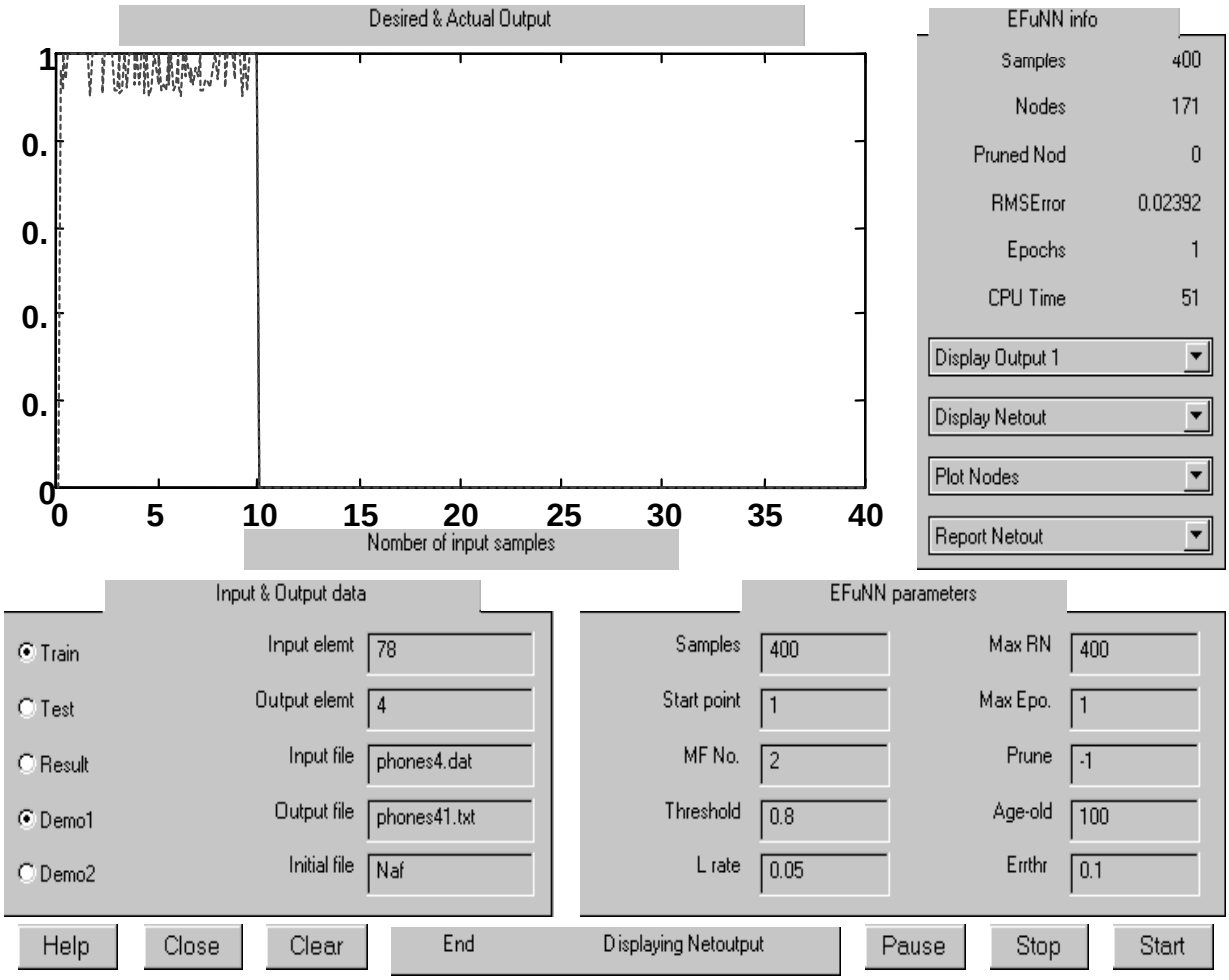


Fig.7b shows the activation of the /I/ output of the evolved EFuNN for the phoneme /I/ test data

3.5. Learning modes in EFuNN. Rule insertion, rule extraction and aggregation.

Different learning, adaptation and optimisation strategies and algorithms can be applied on an EFuNN structure for the purpose of its evolving. These include:

- *Active learning* , e.g. the EFuNN algorithm;
- *Passive learning* (i.e., *cascade-eco*, and *sleep-eco* learning) as explained in section 2;
- *Rule insertion* into EFuNNs [44]. 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 rj for each new rule R, such that the connection weights W1(rj) and W2 (rj) of the rule node represent the rule R. For example, the fuzzy rule *(IF x1 is Small and x2 is Small THEN y is Small*) can be inserted into an EFuNN structure by setting the connections of a new rule node to the fuzzy condition nodes x1- Small and x2- Small and to the fuzzy output node y-Small to a value of 1 each. 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 x1 is 3.4 and x2 is 6.7 THEN y is 9.5*, but here the membership degrees to which the input values x1=3.4 and x2=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 extraction and aggregation.* Each rule node *r*, which represents a prototype, rule, exemplar from the problem space, can be described by its connection weights W1(r) and W2 (r) that define the

association of the two corresponding hyper-spheres from the fuzzy input and the fuzzy output problem spaces. The association is expressed as a fuzzy rule, for example:

IF x1 is Small 0.85 and x1 is Medium 0.15 and x2 is Small 0.7 and x2 is Medium 0.3
THEN y is Small 0.2 and y is Large 0.8

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 MF.

The process of rule extraction can be performed as aggregation of several rule nodes into a larger hyper-spheres as it is shown in fig.8a and fig.8b on an example of three rule nodes r1, r2 and r3 (only the input space is shown there). For the aggregation of two rule nodes r1 and r2, the following aggregation rule is used [44]:

IF (D(W1(r1),W1(r2)) < = Thr1) AND (D(W2(r1),W2(r2)) <= Thr2)
THEN aggregate r1 and r2 into r_{agg} and calculate the centres of the new rule node as:
$W1(r_{agg})$ = average (W1(r1),W1(r2)), $W2(r_{agg})$ = average (W2(r1),W2(r2))

Here the geometrical center between two points in a fuzzy problem space is calculated with the use of an average vector operation over the two fuzzy vectors. This is based on a presumed piece-wise linear function between two points from the defined through the parameters Sthr and Errthr input and output fuzzy hyper-spheres.

Example: The following two rules (rule nodes) r1 and r2 can be aggregated for Thr1=0.15 and Thr2=0.05 into a new rule r_{agg} as it is shown below:

r1: IF x1 is Small 0.85 and x1 is Medium 0.15 and x2 is Small 0.7 and x2 is Medium 0.3
THEN y is Small 0.1 and y is Medium 0.9

r2: IF x1 is Small 0.80 and x1 is Medium 0.2 and x2 is Small 0.8 and x2 is Medium 0.2
THEN y is Small 0.12 and y is Medium 0.88
D(W1(r1),W1(r2))=(0.05+0.05+0.1+0.1)/2/2=0.075 < Thr1=0.15;
D(W2(r1),W2(r2))=(0.02+0.02)/2/1=0.005 < 0.02 < Thr2=0.05;

r_{agg}: IF x1 is Small 0.825 and x1 is Medium 0.175 and x2 is Small 0.75 and x2 is Medium 0.25
THEN y is Small 0.11 and y is Medium 0.89

Through node creation and consecutive aggregation an EFuNN systems can adjust over time to changes in the data stream. Fig.8c shows a hypothetical case of how a rule node rj, which represents a phoneme data cluster, would shift in the phoneme data space with new speakers of different accents talking to the system over time and the system adapts to them.

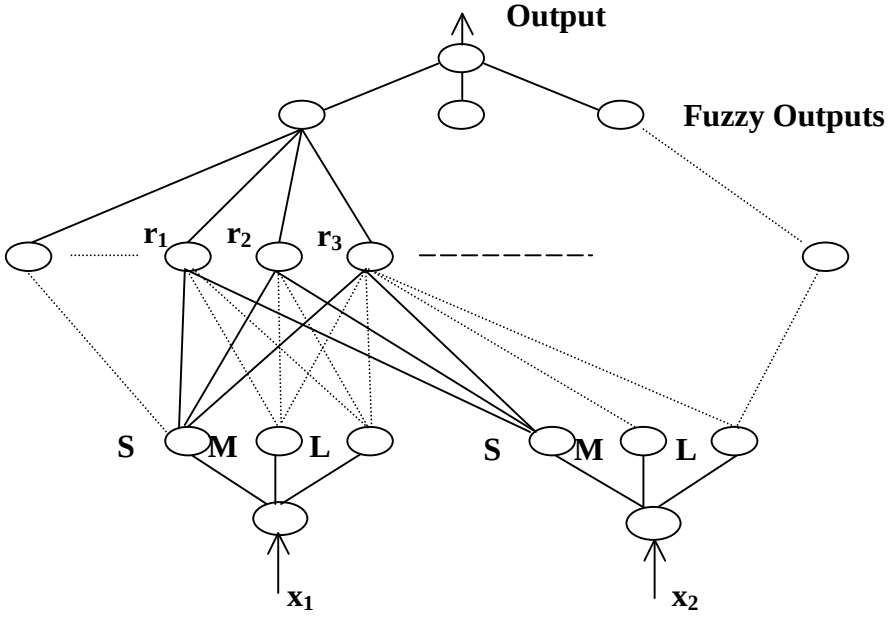


Fig.8a An evolved EFuNN structure;

• *Aggregation and abstraction through ECO-learning*: Aggregation of rule nodes to represent association of larger hyper-spheres from the input and the output space can be achieved through the use of the ECO learning method, when the connection weights $W1^{(1)}$ and $W2^{(1)}$ of an evolved EFuNN1 are used as fuzzy exemplars to evolve an EFuNN2 for smaller values of the sensitivity threshold Sthr and the error threshold Errthr (see fig.9). This process can be continued further to evolve a new EFuNN3 with smaller number of rule nodes, therefore smaller number of rules, and so on. In case of function approximation tasks, the accuracy of the generalisation in this case may decrease depending on the chosen thresholds Thr1 and Thr2 as aggregation means creation of larger prototypes that accommodate more examples having similar input vectors and similar output vectors. For classification tasks where the output value is a symbolic (e.g., 'yes'/'no' class label) the aggregation may not affect the accuracy.

- *Extracting rules for learning temporal pattern correlation:* 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 $(W1(r1),W2(r1))^{(t-1)}$

  THEN $(W1(r1),W2(r2))^{(t)}$ (0.3)

The meaning of the above rule is that examples that belong to the rule (prototype) r1 follow in time examples from the rule prototype r2 with a relative conditional probability of 0.3.

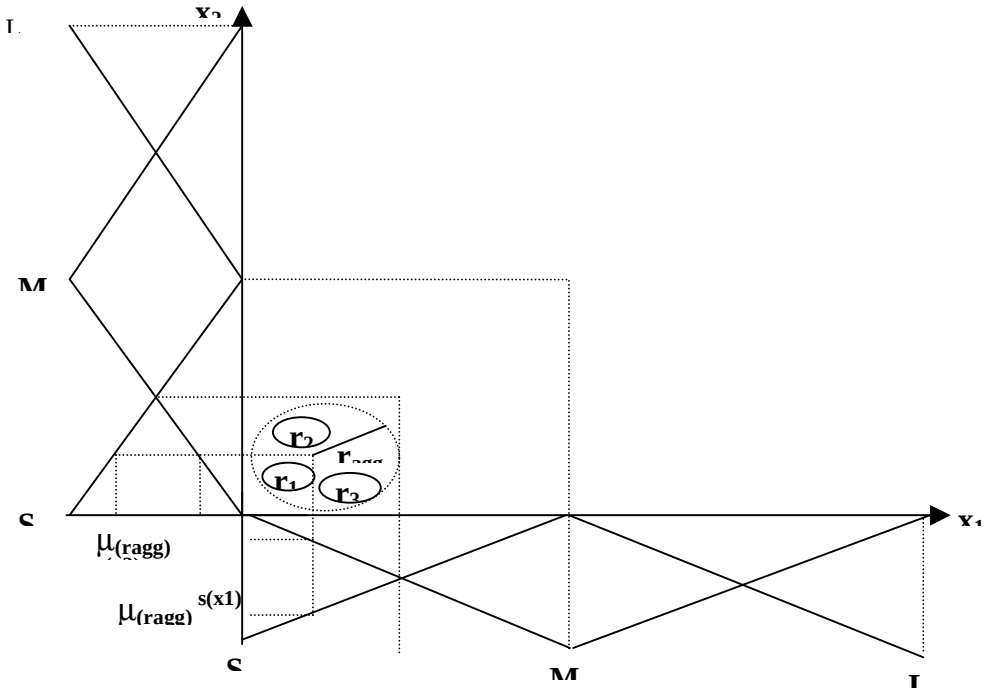


Fig8b The process of aggregation of three rule nodes r1,r2 and r3 into one cluster node r_{agg};

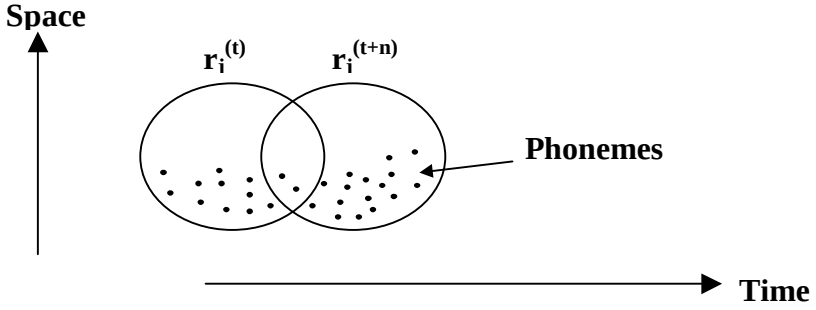


Fig.8c The process of rule node adaptation over time;

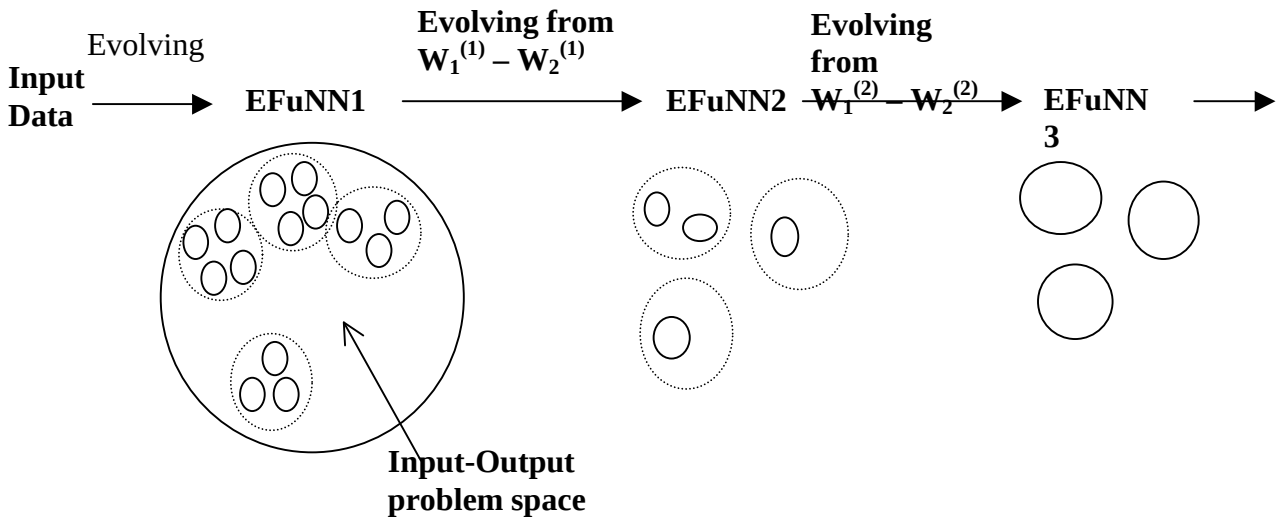


Fig.9. Aggregating rule nodes through ECO learning

• *Changing MF during operation*. This operation may be needed for a refined performance after certain time of the system operation. For example, instead of three MF, the system may perform better if it had five MF for some of the variables. In traditional fuzzy neural networks this change is either not allowed, or is extremely difficult to implement. In EFuNNs there are several possibilities to implement such dynamical changes of MF as it is graphically illustrated on fig.10a,b,c. These are: (a) The stored fuzzy exemplars in W1 and W2 that have three MF are defuzzifyied (e.g., through the center of gravity deffuzification technique) and than used to evolve a new EFuNN structure that has, for example, five MF (fig.10a); (b) New MF can be created (inserted) without a need for the old ones to be changed (fig.10b).The degree to which each cluster centre (each rule node) belongs to the new MF can be calculated through defuzzifying the centres as in case (a); (c) When aggregation of rule nodes is applied after many epochs, it is possible that input or output MF become fuzzy as the centers of the rule hyper-spheres move, so that there is no one-to-one defuzzification procedure from the connection weights back to the real input values as it is the case in fig.10a and 10b. (see an illustration in fig.10c).

• *On-line parameter optimisation*. Once set, the values for the EFuNN parameters will need to be optimised during the learning process. Optimisation can be done through analysis of the behaviour of the system and through a feedback connection from the higher level modules. Genetic algorithms (GA) can also be applied to optimise the EFuNNs structural and functional parameters based on either standard GA algorithms, or on their possible modifications for dynamic, on-line application. The latter case is concerned with an optimisation of parameters to adjust to a continuously incoming stream of data with changing dynamics and changing probability distribution. In this case a segment of the most recent data is stored regularly into an additional memory and a GA is applied on this data to optimise the EFuNN.

With the learning and pruning operations as part of the EFuNN learning algorithm, and with some additional adaptation techniques, an EFuNN can dynamically organise its structure to learn from data in an adaptive, continuous, incremental, life-long learning mode.

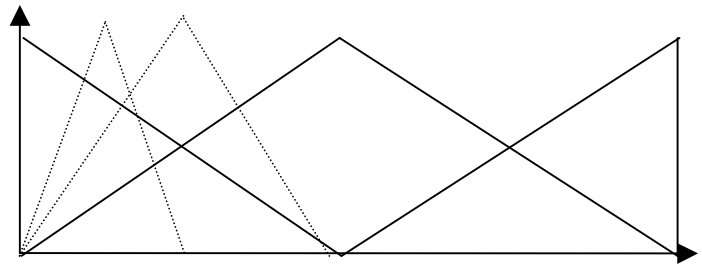


Fig. 10a New MF are inserted without modifying the existing ones.

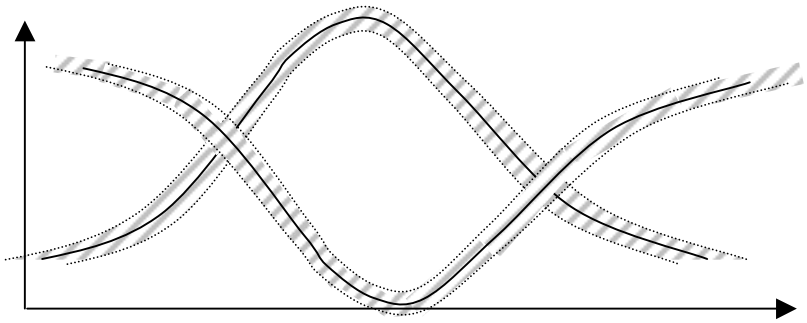


Fig. 10b Fuzzy MF.

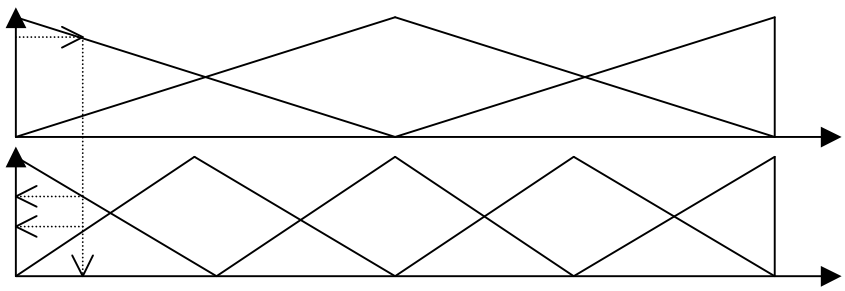


Fig. 10c The number of the MF increases from 3 to 5;

3.6. EFuNNs as universal learning machines. Local and global generalisation

EFuNNs are designed to work in an on-line mode, with a continuous input data stream. An EFuNN is trained (evolved) on input-output vectors of data available over time. Then it is used to generalise on new incoming data Xd for which the output is not known. Once the output vector Yd for the new input data becomes known, the input-output pair (Xd,Yd) is accommodated in the EFuNN structure, which is then used on the next input data, and so on. EFuNNs are memory-based systems, i.e. they store the incoming information as associated input-output clusters (fuzzy rules, prototypes) organised in hyper-spherical forms. The clusters (their centres) are adjustable through the learning parameters lr1 and lr2, so they can 'move' in the problem space in order to accommodate new examples as such become available from the input stream. This continuous, learning process depends very much on the values set for the learning and pruning parameters. The optimal performance of EFuNNs in terms of learning error, generalisation, forgetting and convergence can be achieved through varying their structural and functional parameters. The generalisation ability of EFuNNs depends on the learning and pruning coefficients which can be dynamically adjusted in an ECOS architecture through a feedback connection from the higher level decision module or through optimisation techniques (see fig.1). It will be shown here that EFuNNs are universal learning machines that can learn, subject to a chosen degree of accuracy, any data set D, regardless of the class of problems (function approximation, time series prediction, classification, etc.).

In an on-line learning an EFuNN is evolved incrementally on different segments of data from the input stream (as a partial case this is just one data item). Off-line learning can also be applied on an EFuNN, when the system is evolved on part of the data and then tested on another part from the problem space, which completes the training and testing procedure as it is the case in many traditional NN models.

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

(a) The incoming data is 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 (or an extraction of it). After an EFuNN is evolved on some examples for the problem space, its g*lobal generalisation error* can be evaluated on a set of p new examples from the problem space as follows:

$$GErr = \text{sum}\ \{Err_i\}_{i=1,2,\ldots p},$$

where: Err_i is the error for a vector $\mathbf{x}_i$ from the input space X, which vector has not been and will not be used for training the EFuNN before the value GErr is calculated. After having evolved an EFuNN on a small, but representative part of the whole problem space, its global generalisation error can become sufficiently small. This is valid for both off-line learning mode and on-line learning (when an EFuNN is evolved on k examples and then used to generalise on the next p examples, as it is the case in section 4 when EFuNNs are trained on one articulation data and then tested and adapted on another articulation data of same speakers).

For an on-line learning mode in which the EFuNN is adjusted incrementally on each example from the data stream the generalisation error on the next new input vector (for which the output vector is not known) is called *local generalisation error*. The local generalisation error at the moment t, for example, when the input vector is Xdt, and the calculated by the evolved EFuNN output vector is Ydt', is expressed as Err_t. The cumulative local generalisation error can be estimated as:

$$TErr_t = \text{sum}\ \{Err_t\}_{,\ t=1,2,\ldots i}.$$

In contrast to the global generalisation error, here the error Err_t is calculated after the EFuNN has learned the previous example (Xd(t-1), Yd(t-1)). Each example is propagated only once through the EFuNN, both for testing the error and learning (after the output vector becomes known).

The root mean square error can be calculated for each data point i from the input data stream as:

$$RMSE(i) = sqrt\ (sum\{Err_t\}_{t=1,2,..,i}) / i\ ),$$

where: $Err_t = (d_t - o_t)^2$, d_t is the desired output value and o_t is the EFuNN output value produced for the t_{th} input vector. The non-dimensional error index NDEI(i) can also be calculated (as shown in section 4 and fig.14b,c):

$$NDEI(i) = RMSE\ (i)\ /std\ (D(1:i)),$$

where: std (D(1:i) is the standard deviation of the data points from 1 to i.

(b) Open problem space, where the data dynamics and data probability distribution can change over time in a continuous way. Here, local generalisation error only can be evaluated.

For the two cases (a) and (b) above the following two theorems are valid.

Theorem 1. For any stream of input-output data from a compact and bounded problem space, there is an EFuNN system that can approximate the data to any desired degree of accuracy ξ after a certain time moment T defined by the distribution of the incoming data if the data represents a continuous function in the problem space.

Proof. The proof of the theorem, which is outlined here, is based on the following assumptions. After a time moment T, each of the fuzzy input and the fuzzy output spaces (they are compact and bounded) will be covered by the fuzzy hyper-spheres of the rule nodes generated over time, with a resolution accuracy of r=1-Sthr and Errthr respectively. After a sufficient number of examples from the stream presented by a certain time moment T, both the global generalisation error and the total local generalisation error will saturate to a value E proportional to the chosen value for the error threshold Errthr, therefore each of them will become less than the desired accuracy ξ. This is valid in case of the data stream approximating a continuous function, so that any two data points from a sufficiently small fuzzy input neighbourhood will have sufficiently small difference in the fuzzy output space. It can be precisely proved that any two associated compact and bounded fuzzy spaces X and Y can be fully covered by associated (possibly, overlapping) fuzzy hyper-spheres [38]. A similar theorem for multi-layer perceptrons with sigmoidal activation functions was proved in [10,21]. But here, the on-line learning mode is covered too.

The EFuNNs can also be used to learn sequences from open spaces (case (b)), where the probability distribution and the dynamics of the data sequence can change over time. In this case the system will learn rules and prototypes and the generalisation accuracy will depend on the closeness of the new input data to already evolved prototypes both in space and time.

Theorem 2. For any continuous stream of input-output data from an open problem space, used to evolve an EFuNN, the local generalisation error at a time moment (t+1) will be less than a predefined value ξ if at the time moment t there is a rule node rj = (W1(rj), W2(rj)), such that $D(W2(rj).(1-Dx), Ydt) < \xi$, when $Dx = D(W1(rj,Xdt)) = \min \{D(W1(ri), Xdt))\}$, for i= 1,2,…,rn (rn is the number of the rule nodes evolved in the EFuNN structure until the time moment t).

The proof of this theorem uses the definition of local generalisation and the operations from the EFuNN learning algorithm.

4. Case Studies of Evolving Systems for On-line Incremental Learning

4.1. ECOS and EFuNNs for adaptive speech recognition

Building adaptive speech recognition systems is an important task in the area of spoken language processing [72,34,41,61]. Adaptive speech recognition is concerned with the development of speech recognition systems that: (1) can adapt to new pronunciation (of the same, or a new speaker); (2) can enlarge their vocabulary of words in an on-line mode; (3) can acquire new languages. Here, EFuNNs are illustrated on the problem of phoneme adaptation.

It is well known that, there are a lot of variations in the pronunciation of phones of the same phonemes, and at the same time there are similarities in the pronunciation of phones of different phonemes. These make the recognition of phonemes a very difficult task. Four phoneme data is used here (the same four phonemes as in the example from section 3, but here taken from the words 'pit', 'pet', 'pat' and 'bean' from the same data base, same two speakers [75]). While fig.7a illustrates the "spatial" ambiguity of the phoneme data in the first two-formant space, fig.11a,b illustrates the temporal variability of the /I/ phoneme data (new speaker, not in the database, pronouncing the word 'sit') and the /e/ phoneme data (from the word 'get', speaker 17 from the database) within a small time interval. Fig.11a shows the values of the 26 mel-scale coefficients of the phoneme /I/ data for each of ten consecutive time frames (each of them 11.6 msec long). It can be seen that while there is similarity in the mel-scale vector patterns, there is a significant difference in the values of the main mel coefficients. Fig.11b shows the membership degree to which the second mel-scale coefficient (which is the main one for the phoneme /e/) belongs to a triangular MF denoting "high" value for each of ten consecutive time frames.

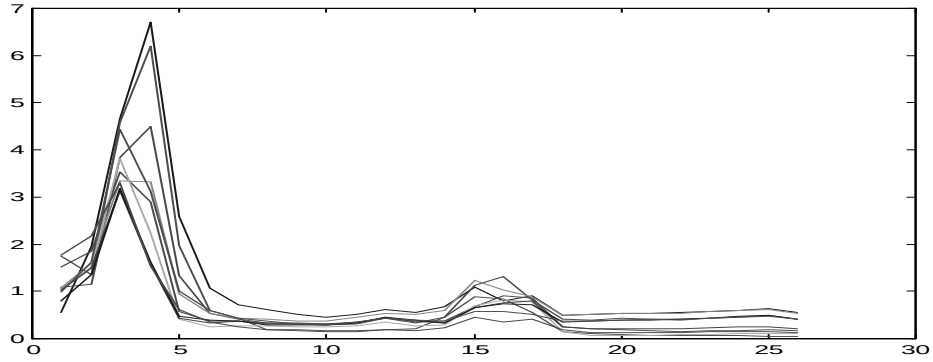


Fig. 11a Phoneme /I/ from 'sit'- 10 consecutive mel scale vectors, each of 26 elements;

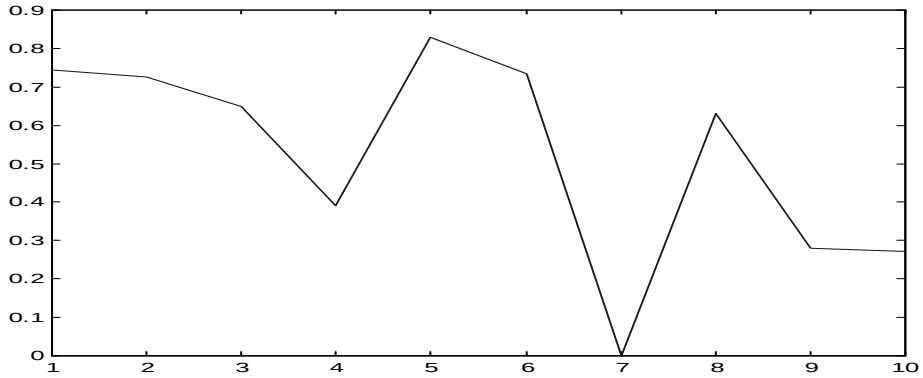


Fig. 11b Phoneme /e/ data – the membership degrees to which 10 consecutive in time values of the second mel scale coefficient belong to the MF of "high" value.

In the experiment below the data is grouped into two data sets – a set A, that constitutes a first pronunciation of the four phonemes, and a set B – a second pronunciation of the same words by the same speakers. Once evolved on set A the system will be tested on set B and if it does not perfom well it will be adapted to set B. The level of forgetting on the set A will be tested.

The following numbers of 78–element frame vectors are used as positive examples (and negative examples in brackets): /I/ - 174 (85); /e/ - 253 (124); /ae/ - 285(138); /i/ - 325 (159). The data is taken from the Otago Speech Corpus (see the example in section 3). Initially four EFuNNs were evolved from the set A through one pass of training for the following parameter values: linear activation functions; SThr=0.5; lr1=lr2=0.5; lr3=0; no pruning; Errthr=0.01. The classification rate was evaluated on set A (to evaluate the training error), and on set B (to evaluate the generalisation of the EFuNNs over a new articulation data of the same speakers (see fig.12). Then all EFuNNs were further trained for one pass on the set B to adapt to the new articulation data. After the additional training the EFuNNs were tested again on set A and set B. The classification rate significantly improved on both set A and set B. This experiment shows that EFuNNs can successfully adapt to new pronuncition without forgetting previous ones. When temporal links were evolved, for a small learning rate of lr3=0.01, the classification accuracy further improved which was expected after having seen the temporal variations within the phones of same phonemes from fig.11a,b.

	4 EFuNNs are evolved for one pass on A and tested on A and on B (in%) on A on B	The evolved on A EFuNNs are adjusted for one pass on B and tested on A and B on A on B	Temporal EFuNNs are evolved for one pass on A and tested on A and on B (lr3=0.001) on A on B	The temporal EFuNNs that were evolved on A are adjusted for one pass on B and tested on A and B (lr3=0.001) on A on B
/I/	95(99) 71(99)	96(99) 97(99)	94(99) 74(99)	96(99) 98(99)
/e/	95(97) 74(96)	97(98) 91(98)	95(97) 80(96)	97(98) 93(98)
ae	98(99) 81(93)	99(99) 94(98)	97(99) 81(94)	99(99) 93(98)
/i/	93(95) 76(82)	92(98) 94(95)	95(96) 75(82)	94(98) 96(95)

Fig.12. True positive and true negative (in brackets) classification accuracy in % for the four- vowel experiment

Fig.13 shows a general framework of an adaptive phoneme-based speech recognition system that adapts its phoneme modules after every unsuccessful recognition attempt. This framework constitutes an ECOS and an EFuNN-based system for the task of phoneme recognition. Classification and adaptation results for the 43 phonemes in NZ English and also a comparative analysis of using MLP, fuzzy neural networks, GAs and EFuNNs for the task of adaptive phoneme recognition are given in [43]. The analysis shows that EFuNNs are superior when used for on-line adaptive phoneme recognition.

Further development in this area includes building ECOS for evolving spoken languages and building multi-modal spoken language processing systems [54,78,37]. The cortical areas of the human brain that are responsible for the speech and the language abilities of humans evolve through the whole development of an individual [72,73,82]. Computer modelling of this process, before its biological, physiological and psychological aspects are made completely known, is an extremely difficult task. It requires flexible techniques for adaptive learning through active interaction with a teaching environment. ECOS and EFuNNs are appropriate models to use for the task.

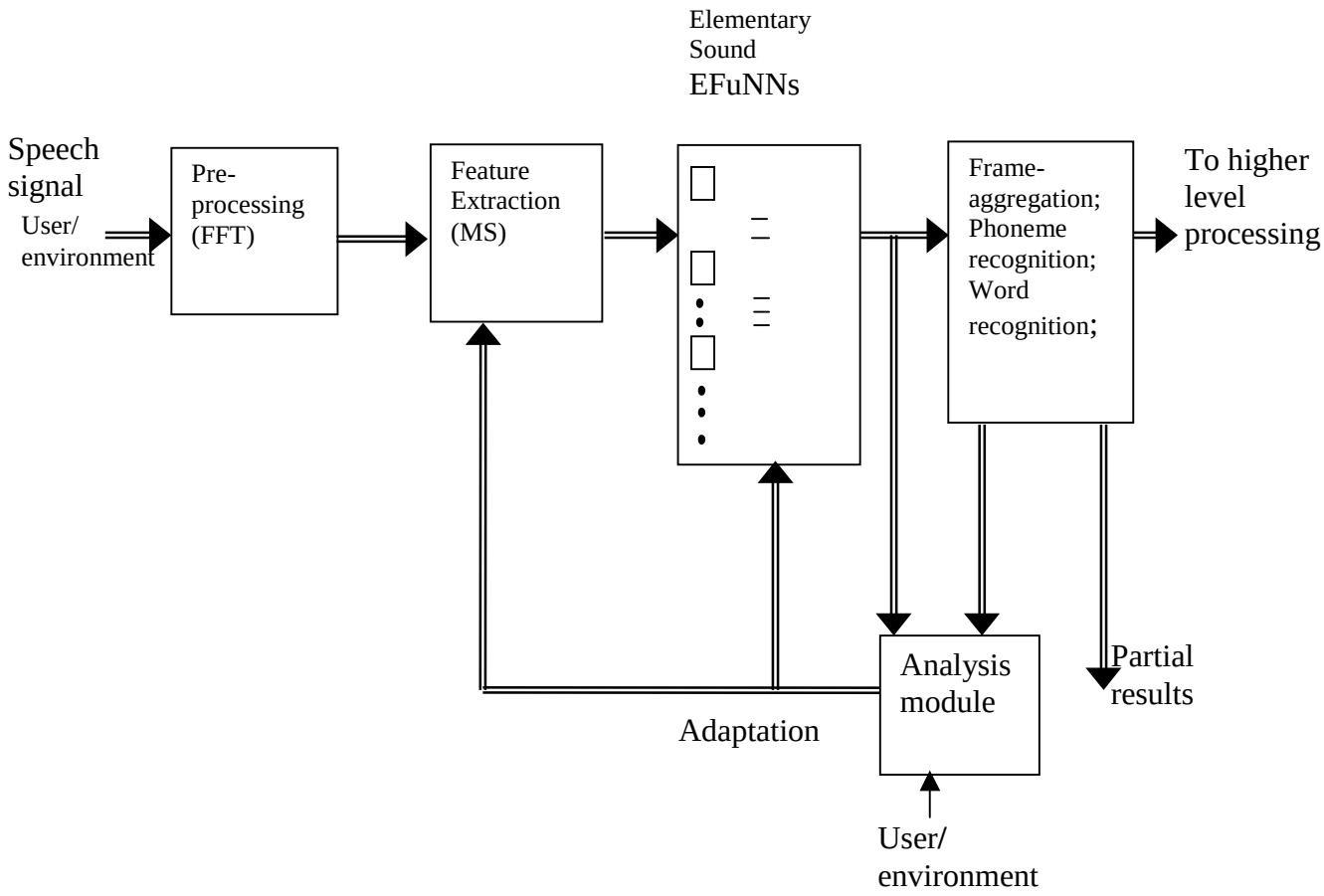


Fig.13. A block diagram of an ECOS and EFuNN-based adaptive speech recognition system where the phoneme modules adapt to new data in an on-line mode during the operation of the system

4.2. EFuNNs for on-line, dynamic time series prediction

EFuNNs, being universal learning machines, can be used for different on-line learning tasks, such as classification, decision making, dynamic time-series approximation and prediction [60,16]. Here the latter is illustrated on the gas-furnace bench-mark time-series data set. The gas-furnace data has been used by many researchers in the area of neuro-fuzzy engineering for control, prediction and adaptive learning [16]. The data set consists of 292 consecutive values 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.

The following steps were taken in the experiments illustrated in fig.14a,b,c:

1) an EFuNN is trained on each data item in an on-line mode and tested immediately to predict the following data item before it is accommodated in the system (fig.14a);
2) an EFuNN is trained on each data item in an on-line mode and tested immediately to predict tree steps ahead data item (fig.14b);
3) an EFuNN is trained on the first half of the data and tested on the whole data set (either in an on-line or in an off-line mode) (fig.14c);
4) the EFuNN from (3) is additionally trained for one pass on the second half of the data.

In the above experiments the EFuNNs were set up with 5 MF for the following parameter values: sensitivity threshold Sthr=0.9; error threshold Errthr=0.05; learning rate for both the first and second layer lr=0.5.

The results shown in fig.14 confirm that an EFuNN can adapt to new data in an on-line mode with just one pass of training on any new data item without forgetting the old data if that is required for the functioning of the system. It can also be seen that after certain time moment T the RMSE and the NDEI converge to a constant value subject to small number. In the case of compact and bounded problem space the error can be made sufficiently small subject to appropriate selection of the parameters of the EFuNN (mainly sensitivity threshold, error threshold, learning and forgetting rates).

4.3. EFuNNs for on-line, intelligent agents

Agent-based techniques allow for implementing modular systems that consist of independent software modules that can communicate with each other and with the user using a standard protocol, can "navigate" in a new software environment searching for relevant data, processing the data and passing results [81]. Intelligent agents can perform intelligent information processing, such as reasoning with uncertainties, rule extraction, generalisation, adaptation. Intelligent agents should be able to adapt to a possibly changing environment as they work. Such adaptation is crucial for a mobile robot navigation, for an adequate decision making on operations with a dynamically changing stock index, or for on-line search on the WWW [5]. ECOS and EFuNNs are well suited to the above requirements and some preliminary results show a good performance of them.

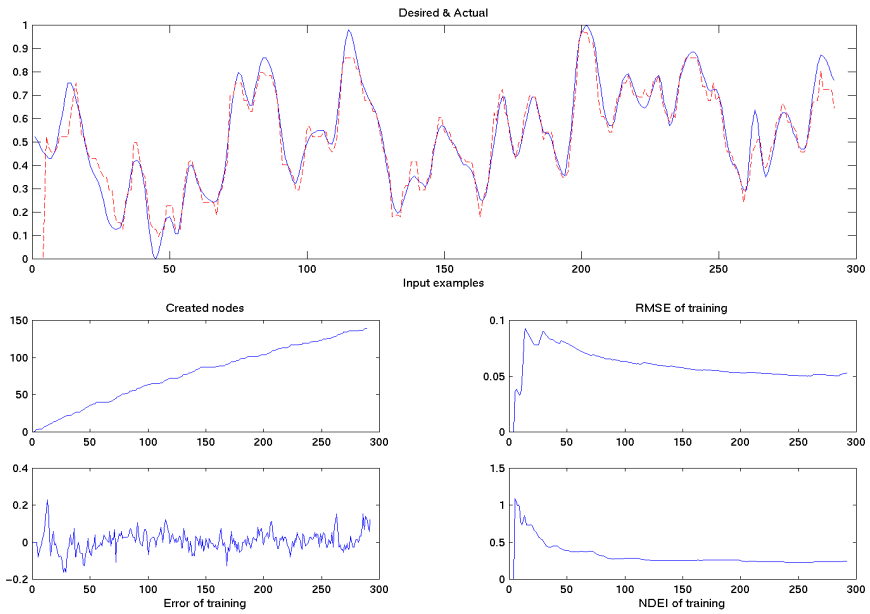


Fig.14a EFuNN is trained on each data item in an on-line mode and tested immediately to predict the following data item before it is accommodated in the system.

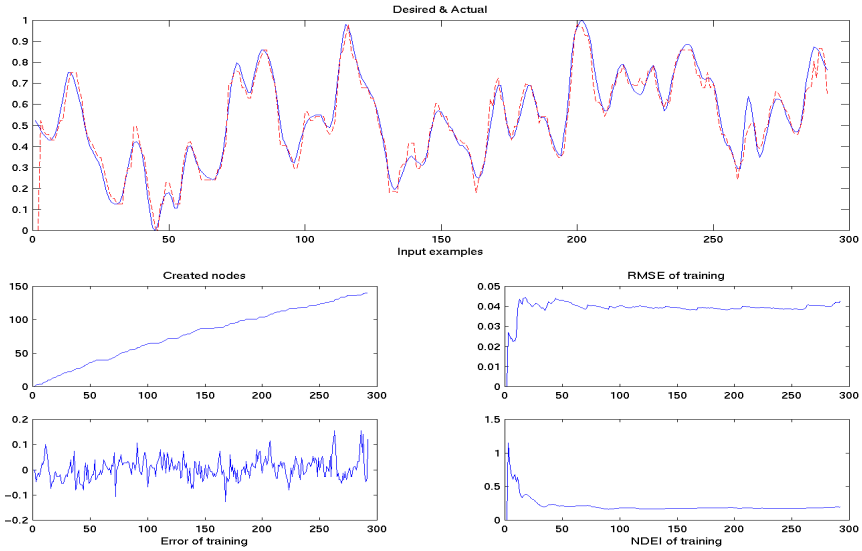


Fig.14b EFuNN is trained on each data item in an on-line mode and tested immediately to predict tree steps ahead data item

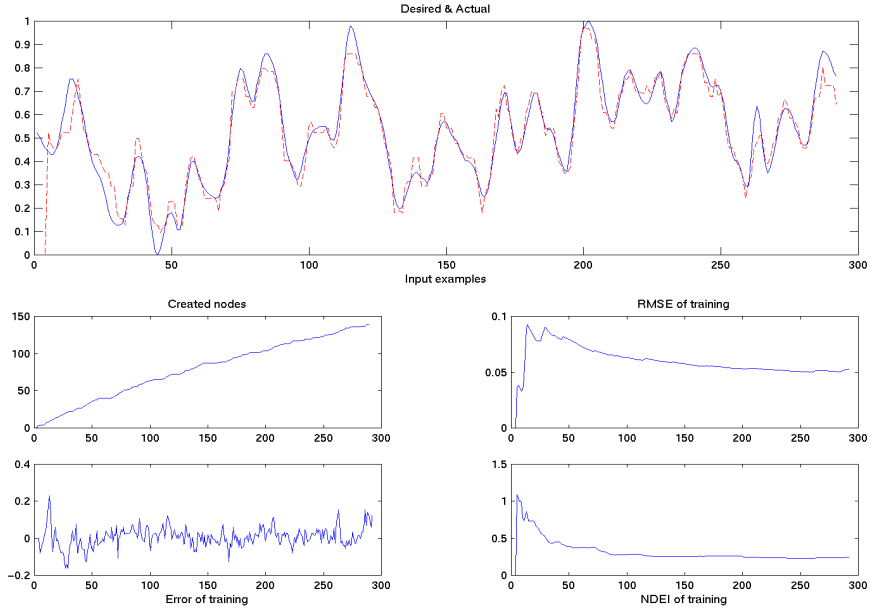


Fig. 14c EFuNN is trained on the first half of the data and tested on the whole data set (either in an on-line or in an off-line mode)

5. Conclusions and directions for further research

This paper presents a framework ECOS for evolving connectionist systems, and evolving fuzzy neural networks EFuNN, in particular, for building on-line, knowledge-based, adaptive learning systems. ECOS have features that address the seven major requirements to the next generation of intelligent information systems as presented in section one. A significant advantage of ECOS and EFuNNs is the local learning procedure which allows for a fast learning (possibly – one pass) after every new data item is entered and only few connections and nodes are changed. 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.

In spite of the advantages of ECOS and EFuNNs when applied for on-line, adaptive learning, there are some difficulties that should be addressed in the future research. These include finding the optimal values for the evolving parameters, such as the sensitivity threshold Sthr, the error threshold Errthr, learning rates lr1, lr2 and lr3, forgetting rate, pruning, etc. For example, pruning of rule nodes has to be made specific for every application, thus depending on the definition of age and the other fuzzy variables in the pruning rule. One solution is to regularly apply genetic algorithms and evolutionary computation as optimisation procedures to the ECOS and EFuNN structures.

Evolving connectionist systems could be viewed as a new AI paradigm. They 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 are knowledge-based systems, logic systems, case-based reasoning systems and adaptive connectionist-based systems, all together. Through self-organisation and self-improvement during its learning process, they allow for simulations of emerging, evolving intelligence to be attempted.

At present more theoretical investigations on the limitations of ECOS and EFuNNs are needed and also more analysis on their biological plausibility.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, contract UOO808. I would like to thank several graduate students for their useful contributions to this final version of the paper and for helping me with some experimental results, above all Mike Watts, Brendon Woodford and Qun Song.

References

1. Albus, J.S., A new approach to manipulator control: The cerebellar model articulation controller (CMAC), Tarns. 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. 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
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. Duda and Hart, “Pattern classification and scene analysis”, New York: Willey (1973)
13. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992).
14. Elman, J., E.Bates, M.Johnson, A.Karmiloff-Smith, D.Parisi and K.Plunkett, Rethinking Innateness (A Connectionist Perspective of Development), The MIT Press, 1997
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 Sidorowitch, 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. Funuhashi, K., On the approximate realization of continuous mappings by neural networks, Neural Networks, 2, 183-192 (1989)
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: 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 Proc. of Iizuka'98, Iizuka, Japan, Oct.1998.
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 - Applications, 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 fuzzy neural networks with 'm-out-of- n' activation nodes for on-line adaptive systems" , TR 99/04, Department of Information Science, University of Otago (1999)
43. Kasabov, N., Watts, M. Spatial-temporal evolving fuzzy neural networks STEFuNNs and applications for adaptive phoneme recognition, TR 99/03 Department of Information Science, University of Otago (1999)
44. 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 Int. Conf. FUZZ-IEEE, Seoul, August 1999 (1999)
45. Kasabov,N. "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2-20 (1996).
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 organizing neural network with creatinga nd deleting methods, Nonlinear 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, 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. Port, R., and T.van Gelder (eds) Mind as motion (Explorations in the Dynamics of Cognition) , The MIT Press, 1995
62. Quartz, S.R., and T.J. Sejnowski, “The neural basis of cognitive development: a constructivist manifesto”, Behavioral and Brain Science, to appear.
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. Robins, A., “Consolidation in neural networks and the sleeping brain, Connection Science”, 8, 2, 259-275, (1996).
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. Schiffman, W., M. Joost, and R. Werner, “Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons” In: Albrecht, R.F., Reeves,
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. Van Ooyen, and J. Van Pelt, “Activity-dependent outgrowth of neurons and overshoot phenomena in developing neural networks”, Journal Theoretical Biology, 167, 27-43 (1994).

78. Waibel, A., M.Vo, P.Duchnovski, S.Manke, "Multimodal Interfaces", Artificial Intelligence Review, 1997.
79. Watts, M., and N. Kasabov, “Genetic algorithms for the design of fuzzy neural networks”, in Proc. of ICONIP'98, Kitakyushu, Oct. 1998.
80. Whitley, D., and C. Bogart, The evolution of connectivity: Pruning neural networks using genetic algorithms. Proc. Int. Joint Conf. Neural Networks, No. 1, 17-22. (1990).
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).