Evolving Fuzzy Neural Networks for Adaptive, On-line Intelligent Agents and Systems

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. This paper discusses and illustrates one paradigm of neuro-fuzzy techniques for building on-line, adaptive intelligent agents and systems. This approach is called evolving connectionist systems (ECOS). ECOS evolve through incremental, on-line learning, both supervised and unsupervised. They can accommodate new input data including new features, new classes, etc. The ECOS framework is presented and illustrated on a particular type of evolving neural networks - evolving fuzzy neural networks. ECOS are orders of magnitude faster than multilayer perceptrons, or fuzzy neural networks and they belong to the new generation of adaptive intelligent systems. ECOS are suitable techniques for building intelligent agents on the WWW, intelligent mobile robots and embedded systems. An ECOS based structure of an intelligent agent is proposed and discussed.

1. Introduction: Adaptive, On-Line, Incremental Learning

The complexity and the dynamics of many real-world problems, particularly in engineering and manufacturing, requires sophisticated methods and tools for building on-line, adaptive decision making and control systems. Such systems should grow as they operate, increase their knowledge, and refine themselves through interaction with the environment [14].

Many developers and practitioners in the area of neural networks (NN) [3], fuzzy systems (FS) [299] and hybrid neuro-fuzzy techniques [10, 13, 18, 20, 21, 23, 29] have enjoyed the power of these now traditional techniques when solving AI problems. Despite of their power, using these techniques for on-line, incremental, life-long learning through adaptation in a changing environment may create difficulties. There are some methods that have already been developed and implemented, such as: ART and Fuzzy ARTMAP [3] for incremental learning; several methods for on-line learning [1, 6, 9, 11, 25]; methods for dynamically changing NN structures during learning process, that include growing and pruning of unnecessary connections and nodes of an NN[7, 12, 24, 26, 27].

The paper continues the list of the techniques from above by introducing a new framework called evolving connectionist systems (ECOS) and a new hybrid model called evolving fuzzy neural network. It discusses their potential for building intelligent agents and intelligent robotic systems.

2. ECOS – Evolving Connectionist and Fuzzy – Connectionist Systems

ECOS are systems that evolve in time through interaction with the environment, i.e. an ECOS adjusts its structure with a reference to the environment (fig.1) as explained in [13 -18].

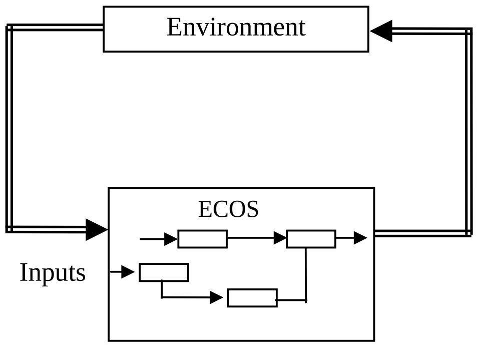


Fig.1. ECOS evolve through interaction with the environment

ECOS are multi-level, multi-modular structures in which many modules have inter- and intra- connections. The evolving connectionist system does not have a clear multi-layer structure. It has a modular open structure. The functioning of the ECOS is based on the following general principles [14 -17].

(1) There are *three levels* of functionality of an ECOS, defined by :
   (a) Genetically specified parameters, such as size of the system, types of inputs, learning rate, forgetting.
   (b) Synaptic connection weights
   (c) Activation of neurons.

(2) Input patterns are presented one by one, in a *pattern mode*, not necessarily having the same input feature sets. After the presentation of each input vector (example), the ECOS associates this example through a *local tuning* of units with either an already existing node (called rule or case node), or it creates a new one. In this respect, ECOS are similar to the case-based learning and reasoning systems. An NN module or a neuron is created when needed at any time of the functioning of the whole system.

(3) The ECOS structure evolves in two phases in a hybrid *unsupervised- supervised mode* of one pass data propagation. In phase one, an input vector **x** is passed through the representation module and the case (rule) nodes become activated based on the similarity between the input vector and the input connection weights. If there is no node activated above a certain *sensitivity threshold (Sthr)*, a new rule neuron (*rn*) is connected ('created') and its input weights are set equal to the values of the input vector **x;** the output weights are set to the desired output vector. In this respect, the ECOS paradigm is similar to ART [3]. Activation either from the winning case neuron (one-out of-n mode), or from all case neurons that have activation values above an *activation threshold (Athr)* (many-of-n mode) is passed to the next level of neurons. In phase two, if there is no need to create a new node, the output error is found and the output connections are adjusted accordingly in a supervised mode. The input connections are also adjusted but

according to similarity between the existing nodes and the input vectors (unsupervised mode).

(4) ECOS have a pruning and aggregation procedure defined. It allows for removing neurons and their corresponding connections that are not actively involved in the functioning of the ECOS (thus making space for new input patterns). Pruning is based on local information kept in the neurons. Each neuron in ECOS keeps a 'track' of its 'age', its average activation over the whole life span, the global error it contributes to, and the density of the surrounding area of neurons. Through aggregation many neurons are merged together based on their similarity.

(5) The case neurons are spatially and temporarily organized. Each neuron has its relative spatial dimensions in regards to the rest of the neurons based on their reaction to the input patterns. If a new rule node is to be created when an input vector **x** is presented, then this node will be positioned closest to the neuron that had the highest activation to the input vector **x,** even though not sufficiently high to accommodate this input vector. Temporal connections between neurons can be established thus reflecting the temporal correlation of input patters.

(6) There are two global modes of learning in ECOS as explained in [13- 18]:

  (a) Active learning mode - learning is performed when a stimulus (input pattern) is presented and kept active.

  (b) ECO training mode - learning is performed when no input pattern is presented at the input of 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 previous input patterns are used as training examples. The connection weights that represent stored input patterns are now used as exemplar input patterns for training other modules in ECOS. This type of learning with the use of 'echo' data is called here ECO training.

  There are two types of ECO training [14, 16]:

    (i) *Cascade eco-training:* a new NN module is created in an on-line mode when conceptually new data (e.g., a new class data) is presented. This mode is similar to the one from [3]. The module is trained on the positive examples of this class, plus the negative examples of the following different class data, and on the negative examples of previously stored patterns in previously created modules taken from the connection weights of these modules.

    (ii) *Sleep eco-training:* modules are created with part of the data presented (e.g., positive class examples). Then the modules are trained on the stored data in the other modules' patterns as negative examples (exemplars).

(7) ECOS provide explanation information extracted from the structure of the NN modules. Each case (rule) node can be interpreted as an IF-THEN rule as it is in the FuNN fuzzy neural network [19-20]. As ECOS are connectionist knowledge-based systems, important part of their functionality is inserting and extracting rules.

(8) ECOS are biologically inspired. Some biological motivations are given in [18].

(9) The ECOS framework can be applied to different types of NN (different neurons, activation functions etc.) and FS. One realisation of the ECOS framework is the

evolving fuzzy neural network EFuNN and the EFuNN algorithm as given in [18] and in section 4. Before the notion of EFuNNs is presented, the notion of the fuzzy neural networks FuNNs is presented in the next section.

3. Fuzzy Neural Networks FuNNs

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist way [10, 13, 19, 23, 29]. FuNN is a particular fuzzy neural network introduced in [19] and developed in [21]. FuNN is part of a comprehensive environment called FuzzyCOPE/3 available free on the WWW:
http://divcom.otago.ac.nz/infosci/kel/software/FuzzyCOPE3/main.htm
It is a connectionist feed-forward architecture with five layers of neurons and four layers of connections. The first layer of neurons receives the input information. The second layer calculates the fuzzy membership degrees to which the input values belong to predefined fuzzy membership functions, e.g. small, medium, and large. The third layer of neurons represents associations between the input and the output variables, fuzzy rules. The fourth layer calculates the degrees to which output membership functions are matched by the input data, and the fifth layer does defuzzification and calculates exact values for the output variables. A FuNN has features of both a neural network and a fuzzy inference machine. A simple FuNN structure is shown in fig.2. The number of neurons in each of the layers can potentially change during operation through growing or shrinking. The number of connections is also modifiable through learning with forgetting, zeroing, pruning and other operations [19, 21]. The membership functions (MF) used in FuNN to represent fuzzy values are of triangular type, the centres of the triangles being attached as weights to the corresponding connections. The MF can be modified through learning that involves changing the centres and the widths of the triangles.

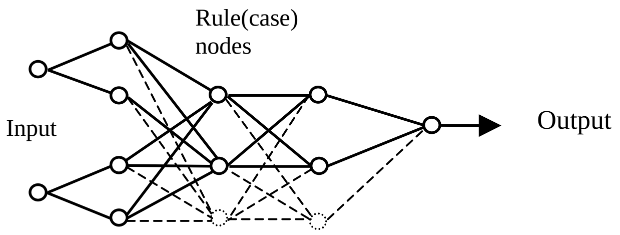


Fig. 2. A FuNN structure of two inputs (input variables), two fuzzy linguistic terms for each variable (two membership functions). The number of the rule (case) nodes can vary. Two output membership functions are used for the output variable.

Several algorithms for training, rule insertion, rule extraction and adaptation have been developed for FuNN [19, 21]. FuNNs have several advantages when compared with the traditional connectionist systems, or with the traditional fuzzy systems: they are statistical and knowledge engineering tools; they are relatively robust to catastrophic forgetting, i.e. when they are further trained on new data, they keep a reasonable memory of the old data; they interpolate and extrapolate well in regions where data is sparse; they accept both real input data and fuzzy input data represented as singletons (centres of the input membership functions). The above listed features of FuNNs make them universal statistical and knowledge engineering tools. Many applications of FuNNs have been developed and explored so far [19, 21].

4. Evolving Fuzzy Neural Networks EFuNNs

4.1. The EFuNN Principles and Structures

EFuNNs are FuNN structures that evolve according to the ECOS principles. EFuNNs adopt some known techniques from [3,19, 22], but here all nodes in an EFuNN are created/connected during (possibly one-pass) learning. The nodes representing membership functions (MF) (fuzzy label neurons) can be modified during learning. As in FuNN, each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantisation of this variable. For example, three neurons can be used to represent "small", "medium" and "large" fuzzy values of the variable. Different MFs can be attached to these neurons (triangular, Gaussian, etc.). 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 MFs to a degree greater than a set threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN. An optional short-term memory layer can be used through a feedback connection from the rule (also called, case) node layer (see fig.3). The layer of feedback connections could be used if temporal relationships between input data are to be memorized structurally.

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, that either no rule nodes exist prior to learning and all of them are created (generated) during the evolving process, or 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. The EFuNN evolving algorithm presented in the next section does not make a difference between these two cases.

Each rule node (e.g., r1) represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space (see fig.4), 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) to an input vector (e.g., (Xd2,Yd2)) in order for the new input vector to be associated to this rule node.

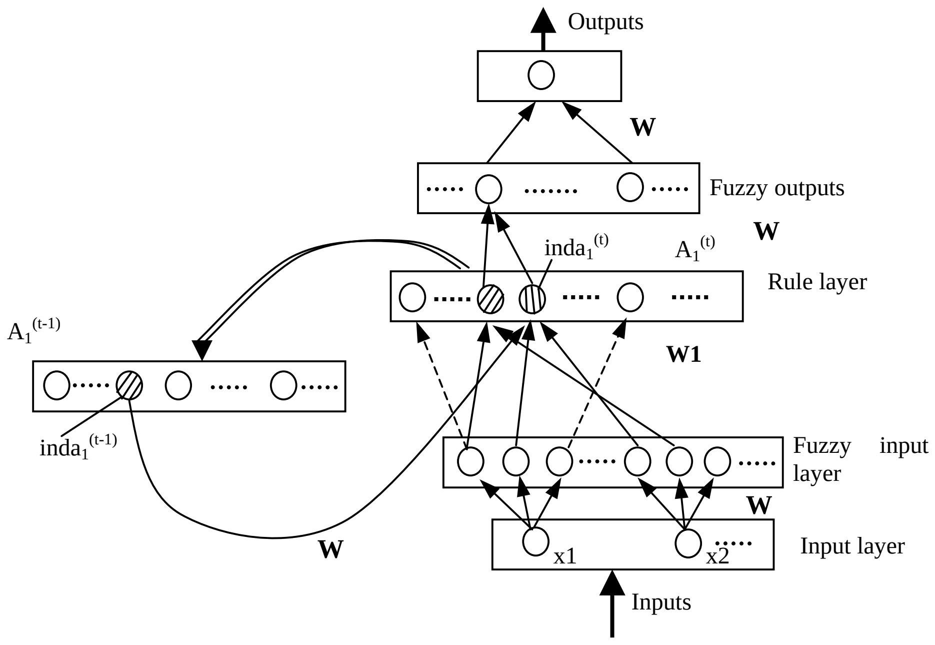


Fig.3. An EFuNN architecture with a short term memory and feedback connections

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), 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 r1 with a centre r_1^1, than a new data point d2=(Xd2,Yd2), that is within the shaded area as shown in fig.4, 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 two data points. The adjustment of the center 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 r1 from $W1(r_1^1)$ and $W2(r_1^1)$ to $W1(r_1^2)$ and $W2(r_1^2)$ as it is presented in 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 difference vector; $A1(r_1^1)$ is the activation of the rule node r_1^1 for the input vector Xd2.

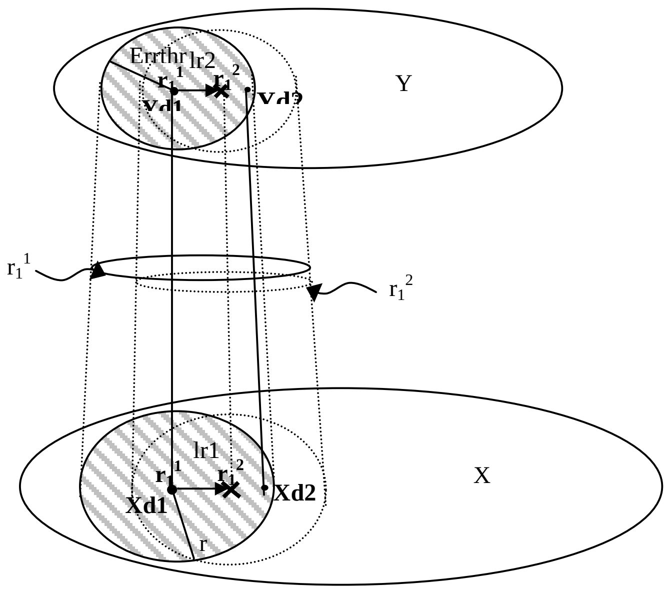


Fig.4. Each rule node created during the evolving process associates a hyper-sphere from the fuzzy input space to a hyper-sphere from the fuzzy output space. Throuh accommodating new nodes the center of the rule node moves slightly.

The idea of dynamic creation of new rule nodes over time for a time series data is graphically illustrated in fig.5

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.3 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).

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 difference value and $r^{(t-1)}$ is the winning neuron at time moment (t-1).

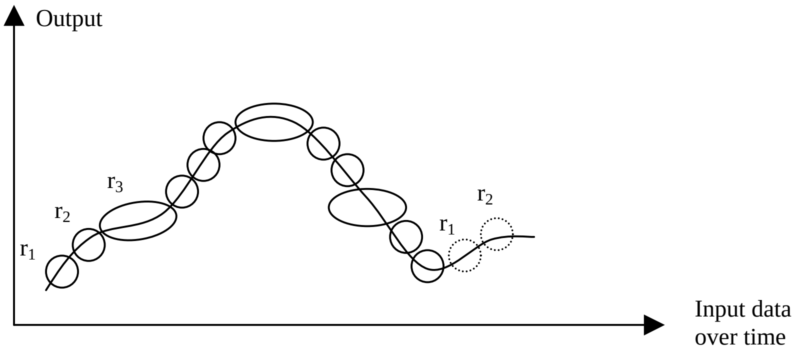


Fig.5. The rule nodes in an EFuNN evolve in time depending on the similarity in the input data

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.

4.2. The EFuNN algorithm

The EFuNN algorithm has been first presented in [18]. A new rule node rn is connected (created) and its input and output connection weights are set 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. Vector and matrix operation expressions are used for the sake of simplicity of presentation.

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. 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 normalised fuzzy local 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))/ 2) / sum (W1(j))

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 is calculated as explained above:

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.
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, 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 EFuNN, or DEFuNN) .

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.

ECOS, and EFuNNs in particular, allow for different learning strategies to be experimented, depending on the type of data available and on the requirements of the learning system. Several of these are introduced and illustrated in [17, 18]. They are:

- *Incremental, one-pass learning:* Data is propagated only once through the EFuNN.
- *Incremental, multiple-pass learning:* Consecutive passes of data on evolved EfuNNs, are performed
- *Using positive examples only*
- *Cascade eco-learning*
- *Sleep eco-training:* Different modules evolve quickly to capture the most important information concerning their specialised functions (e.g., class information). The modules store exemplars of relevant for their functioning examples during the active training mode. After that, the modules begin to exchange exemplars that are stored in their W1 connections as negative examples for other modules to improve their performance.
- *Unsupervised and reinforcement learning:* Unsupervised learning in ECOS systems is based on the same principles as the supervised learning, but there is no desired output and no calculated output error
- *Rule insertion and rule extraction*: these algorithms allow for insertion of fuzzy rules in an EFuNN structure at any time of its operation, or extraction of a minimal set of fuzzy rules (aggregated), that is in general much smaller in size than the number of the rule nodes.

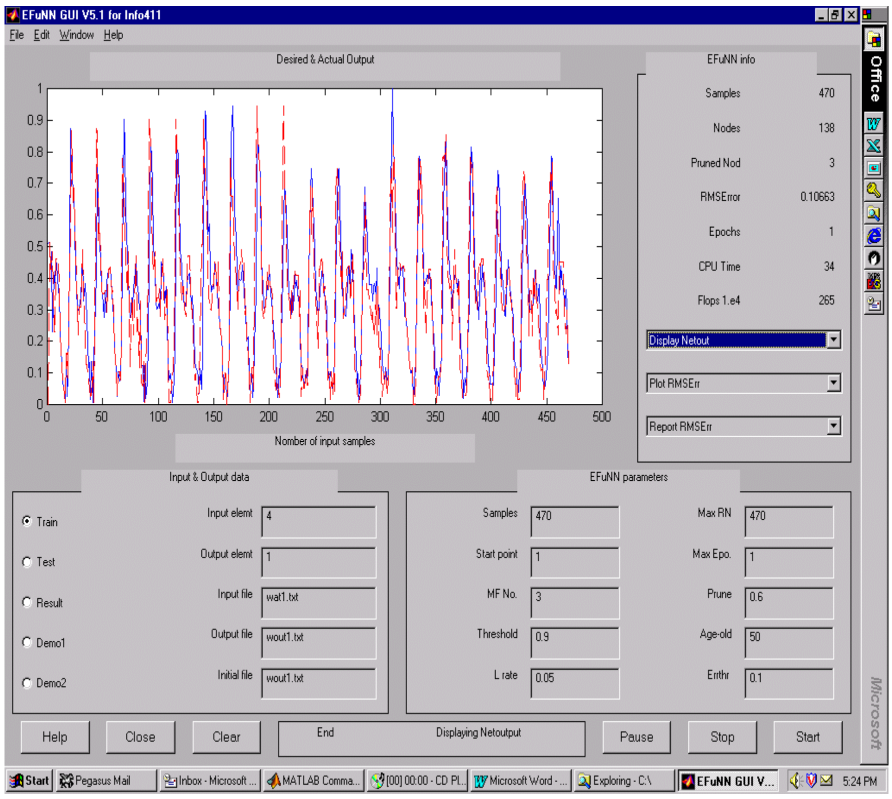
4.3. EFuNN Simulators

Figure 6 shows the GUI of an EFuNN simulator. EFuNN simulators and MATLAB functions are available from the WWW site:
http://divcom.otago.ac.nz/infosci/kel/software/FuzzyCOPE3/main.htm.

It is possible to set values for the following parameters: sensitivity threshold SThr, error threshold Ethr, learning rates lr1 and lr2, number of inputs, number of membership functions, number of outputs, passes of learning.

The simulation and the testing can be done either in an on-line, or in an off-line mode. In an on-line mode after learning each input example, the system is tested on the following input data to locally predict the corresponding output value. After the output value becomes known this example is learned by the system and the next output is predicted, etc. This is illustrated in fig.7 on a waste-water flow data hourly collected (see site http://divcom.otago.ac.nz/infosci/kel/software/datasets/). The task of the evolved EFuNN is to learn in an-on-line mode and predict the next hour flow volume, which is shown in the figure.

The off-line mode is similar to the way multilayer perceptrons and other neural network types for supervised learning are trained and tested.



EFuNN info

Samples	470
Nodes	138
Pruned Nod	3
RMSError	0.10663
Epochs	1
CPU Time	34
Flops 1.e4	265

Display Netout
Plot RMSErr
Report RMSErr

Input & Output data

[✓ Train] Test Result Demo1 Demo2

Input elemt	4
Output elemt	1
Input file	wat1.txt
Output file	wout1.txt
Initial file	wout1.txt

EFuNN parameters

Samples	470	Max RN	470
Start point	1	Max Epo.	1
MF No.	3	Prune	0.6
Threshold	0.9	Age-old	50
L rate	0.05	Errthr	0.1

End   Displaying Netoutput

Fig.6. The GUI of an EFuNN simulator

5. ECOS and EFuNNs for Adaptive, On-Line, Intelligent Agents and Systems

Agent-based techniques allow the implementing of modular systems that consist of independent software modules. These modules can communicate with each other and with the user by using a standard protocol and they can ʼnavigate' in a new software environment by searching for relevant data, then process the data and pass the results [28]. Intelligent agents can perform intelligent information processing, such as reasoning with uncertainties and generalisation. Intelligent agents should be able to adapt to a possibly changing environment as they work in an on-line mode. Such adaptation is crucial for mobile robot navigation, or for an adequate decision making on operations with a dynamically changing data.

A general block diagram of the architecture of an ECOS and EFuNN-based adaptive, on-line agent is given in fig. 7. It consists of the following blocks:

- Pre-processing (filtering) block for input data (this block checks input data for consistency; selects appropriate input features and vectors)

- EFuNN modules that are continuously trained with data in one of the explained in section 4 modes. Rules can be are inserted at any time of the operation of the agent.
- A block for decision making/control and communication - this block communicates with other agents and the environment and sends the results produced by the EFuNN modules.
- Adaptation block - this block compares the behaviour of the agent with a desired behaviour over regular periods of time. The error is used to adjust/adapt the evolving EFuNN modules in a continuous mode. Genetic algorithms can be applied here or other optimisation procedures [5, 8].
- Rule extraction, explanation block - this block extracts rules from the evolved EFuNN modules for explanation purposes

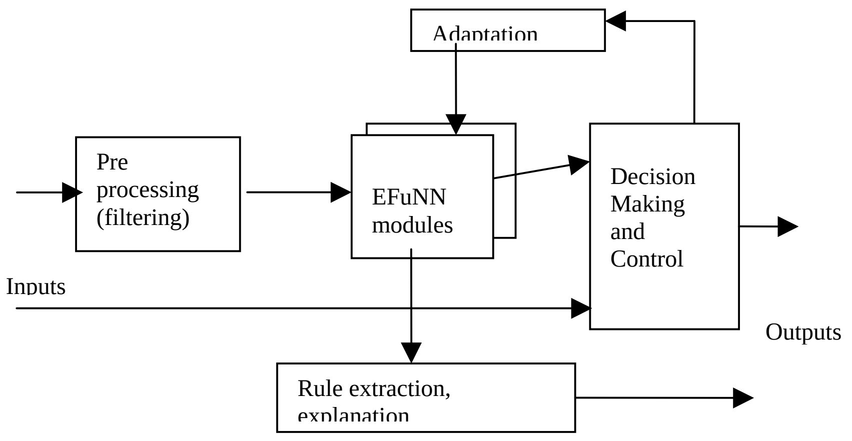

Adaptation
Pre processing (filtering)
EFuNN modules
Decision Making and Control
Inputs
Outputs
Rule extraction, explanation

Fig.7. A block diagram of an ECOS-based agent for on-line, intelligent decision and control

The above general scheme of an intelligent evolving agent is currently being applied in the Knowledge Engineering Laboratory at the University of Otago to two classes of problems:

- Intelligent agents on the WWW, for the purpose of: learning from data repositories; climate prediction; financial decision making;
- Mobile robot control.

For solving specific problems from the two generic classes of problems, a repository of different connectionist, AI and data processing techniques (including different types of EFuNNs) are organised in a Repository for Intelligent Connectionist Based InformationSystemsRICBIS (http://divcom.otago.ac.nz/infosci/kel/software/RICBIS/. For solving problems from the first class the Voyager environmnet is currently being experimented for building intelligent agents for ditributed processing on the WWW. For the second class of applications both sensory and visual information are enabled to collec and process in an on-line mode with the use of EFuNNs. The experiments deal with speech, image and text input. The evolving agents, as part of the robot's software, evolve in real time for different purposes of recognition of object classes, for navigation in a new environment, and for adaptation to new input features. The experiments are in their initial phase and more results are expected in the near future.

6. Conclusion

This paper discusses the ECOS framework for evolving connectionist systems and for evolving fuzzy neural networks (EFuNNs) and introduces a framework for building on-line, adaptive learning agents and agent-based systems. ECOS have features that address the major requirements for the next generation of intelligent and adaptive information systems, such as fast learning (possibly, one pass learning); continuous on-line incremental learning and adaptation; the possibility of working in a life-long learning mode; storing information and data for a consecutive improvement and rule extraction.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, New Zealand, project UOO808.

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 (1997) Brain-like computing and intelligent information systems, Springer Verlag
3. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B., FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps, IEEE Transactions of Neural Networks , vol.3, No.5 (1991), 698-713
4. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992)
5. Freeman, J.A.S., Saad, D., On-line learning in radial basis function networks, Neural Computation vol. 9, No.7 (1997)
6. Fritzke, B., A growing neural gas network learns topologies, Advances in Neural Information Processing Systems, vol.7 (1995)
7. Fukuda, T., Komata, Y., and Arakawa, T. "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)
8. Gaussier, P. and Zrehen, S., A topological neural map for on-line learning: Emergence of obstacle avoidance in a mobile robot, In: From Animals to Animats No.3, (1994) 282-290
9. Hashiyama, T., Furuhashi, T., Uchikawa, Y.(1992) 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.
10. Heskes, T.M., Kappen, B. (1993) On-line learning processes in artificial neural networks, in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233

11. Ishikawa, M. (1996) "Structural Learning with Forgetting", Neural Networks 9, 501-521.
12. Jang, R. (1993) ANFIS: adaptive network-based fuzzy inference system, IEEE Trans. on Syst.,Man, Cybernetics, 23(3), May-June 1993, 665-685
13. Kasabov, N. ECOS: A framework for evolving connectionist systems and the eco learning paradigm, Proc. of ICONIP'98, Kitakyushu, Oct. 1998, IOS Press, 1222-1235
14. Kasabov, N. Evolving connectionist and fuzzy connectionist system for on-line decision making and control, in: Soft Computing in Engineering Design and Manufacturing, Springer Verlag, 1999
15. Kasabov, N. The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems, Journal of Advanced Computational Intelligence, 2 (6) 1998, 1-8
16. Kasabov, N. Evolving connectionist and fuzzy connectionist systems. IEEE Transactions on Man, Machine and Cybernetics, submitted
17. 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
18. Kasabov, N.(1996) Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA.
19. Kasabov, N., "Adaptable connectionist production systems". Neurocomputing, 13 (2-4) 95-117 (1996).
20. Kasabov, N., Kim J S, Watts, M., Gray, A (1997) FuNN/2- A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition, Information Sciences - Applications, 101(3-4): 155-175 (1997)
21. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, vol.78, N-9, pp.1464-1497.
22. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).
23. 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).
24. Rummery, G.A. and Niranjan, M., On-line Q-learning using connectionist systems, Cambridge University Engineering Department, CUED/F-INENG/TR 166 (1994)
25. Sankar, A. and R.J. Mammone, Growing and Pruning Neural Tree Networks, IEEE Trans. Comput. 42(3) 291-299 (1993).
26. Whitley, D. and Bogart, C., The evolution of connectivity: Pruning neural networks using genetic algorithms. Proc. Int. Joint Conf. Neural Networks, No.1 (1990) 17-22.
27. Woldrige, M. and Jennings, N. Intelligent agents: Theory and practice, The Knowledge Engineering review (10) 1995
28. Yamakawa, T., H. Kusanagi, E. Uchino and T.Miki, (1993) "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017-1020
29. Zadeh, L. 1965. Fuzzy Sets, Information, and Control, vol.8, 338-353.