Chapter 7. Evolving Connectionist and Fuzzy - Connectionist Systems: Theory and Applications for Adaptive, On-line Intelligent 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. The paper introduces one paradigm of neuro-fuzzy techniques and an approach to building on-line, adaptive intelligent 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. 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. ECOS are three to six orders of magnitude faster than multilayer perceptrons, or fuzzy neural networks, both trained either with the backpropagation algorithm, or with a genetic programming technique. ECOS belong to the new generation of adaptive intelligent systems. This is illustrated on several real world problems for adaptive, on-line classification, prediction, decision making and control: phoneme-based speech recognition; moving person identification; wastewater flow time-series prediction and control; intelligent agents; financial time series prediction and control. The principles of recurrent ECOS and reinforcement learning are outlined.

Key words: evolving neuro-fuzzy systems; fuzzy neural networks; on-line adaptive control; on-line decision making; intelligent agents

1. Introduction: Adaptive, on-line, incremental learning - problems with the conventional neuro-fuzzy techniques and seven requirements of the next generation of intelligent systems

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.

Many developers and practitioners in the area of neural networks (NN), fuzzy systems (FS) and hybrid neuro-fuzzy techniques have enjoyed the power of these now-traditional techniques when solving AI problems. Several difficulties manifested when these techniques were applied to real world problems, such as

speech and image recognition, adaptive prediction, adaptive on-line control, and constructing intelligent agents. These tasks require flexible learning and dynamically adaptive intelligent systems (IS) that have open structures and are able to process both data and knowledge.

Seven major requirements of intelligent systems (that are addressed in the ECOS framework presented later) are listed below:

(1) An IS should be able to learn quickly from large amounts of data therefore using fast training, e.g. one-pass training.

(2) An IS should be able to adapt in a real time and in an on-line mode where new data are accommodated as they come.

(3) An 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, e.g., the system creates on the fly new inputs, new outputs, new connections, and new nodes. An IS should be able to accommodate in an incremental way everything that is, and that will become, known about the problem, i.e. in a supervised, or in an unsupervised mode, using one modality or another, accommodating data, rules, text, image, etc.

(4) An IS should be memory-based, plus possess data and exemplar storage and retrieval capacities.

(5) An IS should be able to learn and improve through active interaction with other ISs and with the environment in a multi-modular, hierarchical fashion.

(6) An IS should adequately represent *space* and *time* in their different scales; it should have parameters to represent short- and long-term memory, age, forgetting, etc.

(7) An IS should be able to analyse itself in terms of behaviour, global error, and success; to explain what it has learned and what it knows about the problem it is trained to solve; to make decisions about its own improvement.

When designing IS that meet fully, or partially, the above seven requirements, one should take into account what is known about the nervous system and the human brain, especially in case the brain is the 'best IS' for the task (e.g. image and speech recognition, object identification, language acquisition). An IS should, if necessary, incorporate in its structure and behaviour principles from living organisms and the human brain.

It is unlikely that unless the above seven problems are addressed in the current and the future theory of IS, there will be significant progress achieved in areas such as adaptive speech recognition and language acquisition, intelligent agent systems, adaptive intelligent prediction and control systems, mobile robots, visual monitoring systems, multi-modal information processing, and many more.

In respect to the above seven issues, the theory and practice of IS development have not gone much farther than other computational and modelling techniques, for example the traditional statistical methods. However some of the above seven issues have been acknowledged and addressed since the early phases of the development of NN, FS, and IS. Several NN theories, models and methods for adaptive learning and dynamical modification of NN structures have been introduced so far (some of them referenced below).

Even though the learning algorithms of NNs strongly relate to the NN structures, the dualism of the learning and the structure still exists and many

connectionist methods deal with the 'learning only' issue, another - with the 'structure only' issue. Some of the references below are given in the respective aspect, i.e. learning, or structure.

Adaptive learning is aiming at solving the well-known *stability/plasticity dilemma* [8]. Methods for adaptive learning fall into three categories, namely incremental learning, lifelong learning, and on-line learning.

Incremental learning is the ability of NNs to learn new data without destroying (or at least fully destroying) the learned patterns from old data and without a need to be both trained on the new data and retrained on the old data. Significant progress in incremental learning has been achieved due to the Adaptive Resonance Theory (ART) [8,9,10] and its various models, which 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. Both growing and pruning are involved in the learning process.

On-line learning is concerned with learning data as the system operates (usually in real time); data might exist only for a short time. Methods for on-line learning in NN are studied in [1,17,20,26,38,64]. These methods unfortunately do not deal with dynamically changing NN structures , neither they deal with dynamically changing environment where the NNs operate.

In the case of the NN *structure*, the *bias/variance dilemma* has been acknowledged by several authors [8,29]. The dilemma is that 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 is too much variance that resulting in over-training, poor generalisation, etc. In order to avoid this problem, a NN (or an IS) structure should change dynamically during the learning process, thus better representing the patterns in the data and the changes in the environment. In terms of dynamically changing IS structures, there are three approaches taken so far: constructivism, selectivism, and a hybrid approach [29].

Constructivism is about developing NNs that have a simple initial structure and grow during its operation. This theory is supported by biological facts [61]. The growth can be controlled by either a similarity measure (similarity between new data and already learned ones), or by the output error measure, or by both. A measure of difference between an input pattern and already stored ones is used to insert new nodes in ART1 and ART2 [8]. There are other methods that insert nodes based on the evaluation of the local error: the Growing Cell Structure and Growing Neural Gas [18], and Dynamic Cell Structures. Other methods insert nodes based on a global error evaluation of the performance of the whole NN. Such method is the Cascade-Correlation [16]. Methods that use both similarity and output error for node insertion are used in Fuzzy ARTMAP [10].

Selectivism is concerned with *pruning* unnecessary connections in a NN that starts its learning with many, in most cases redundant, connections [60,62]. Pruning connections that do not contribute to the performance of the system can be done by using several methods: Optimal-Brain Damage [53], Optimal Brain Surgeon [25], Structural Learning with Forgetting [27,50,51,57], Training-and-

Zeroing [39], and regular pruning [11]. Both growing and pruning are used in [66].

Genetic algorithms (GA) and evolutionary computation have been widely used for optimising the structures of NNs and IS [19,44,59]. GAs are heuristic search techniques that find the optimal or near optimal solution from a solution space [21,58,59]. They utilise ideas from Darwinism [15]. Unfortunately, most of the evolutionary computation methods developed so far assume that the solution space is fixed, i.e. the evolution takes place within a pre-defined problem space and not in a dynamically changing and open one, thus not allowing for real on-line adaptation. The implementations so far have been also very time-consuming, and this also prevents them from being used in real-time applications.

Some of the seven issues outlined above have already been addressed in the so-called knowledge-based neural networks (KBNN) [22,54,67,74]. Knowledge is the essence of what an IS system has learned [58]. KBNN are neural networks pre-structured in such a way that allows for data and knowledge manipulation, which includes learning from data, rule insertion, rule extraction, adaptation and reasoning. KBNN have been developed either as a combination of symbolic AI systems and NN [22,70], or as a combination of fuzzy logic systems [80] and NN [10,24,28,37,40,54]. Rule insertion and rule extraction operations are examples of how a KBNN can accommodate existing knowledge along with data, and how it can explain what it has learned. There are different methods for rule extraction, well tested and broadly applied so far [4,37,40,49,54].

There has been a fast development of hardware systems that support the implementation of adaptive intelligent systems. Such hardware systems are the cellular automata systems, e.g. the evolutionary brain-building systems [14]. These systems grow through connecting new neighbouring cells in a regular cellular structure. Simple rules, embodied in the cells, are used to achieve the growing effect. Unfortunately the rules do not change during the evolution of the hardware systems, thus making the adaptation of the growing structure limited. Field programmable gate arrays (FPGA) provide another methodology and technology for implementing growing, adaptive intelligent systems (see the two chapters at the end of this volume). In order to utilise fully this technology, new methods for building on-line, adaptive, incrementally growing and learning systems are needed.

Despite the successful development and use of NN, FS, GA, hybrid systems, and other IS methods for adaptive training, radically new methods and systems are required both in terms of learning algorithms and structure development in order to address the seven major requirements of the future IS. A model called ECOS (Evolving COnnectionist Systems) that addresses all seven issues is introduced in this chapter, 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 4 to develop an evolving fuzzy neural network model called EFuNN. Several learning strategies of ECOS and EFuNNs are introduced in section 5. In the following sections ECOS and EFuNNs are applied to several benchmark problems as well as to real world tasks such as adaptive phoneme recognition, on-line voice and person identification in a noisy environment, and adaptive learning

of a stock index through intelligent EFuNN-based agents. Some biological motivations for the development of ECOS are given in section 11. Section 12 briefly outlines directions for further development of ECOS.

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). A block diagram of the ECOS framework is given in fig.2 [33].

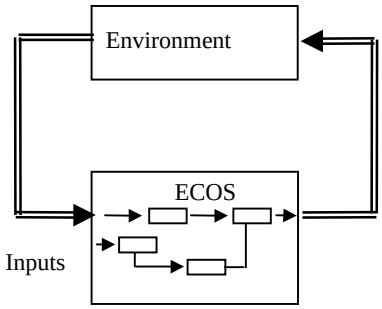

Fig.1. ECOS evolve through interaction with the environment

ECOS are multi-level, multi-modular structures where 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 main parts of an ECOS are described below.
(1) Presentation part . This performs filtering of the input information, feature extraction, and forming the input vectors. The number of inputs (features) can vary from example to example.
(2) Representation and memory part, where information (patterns) is stored. It is a multi-modular, evolving structure of NN modules organised in spatially distributed neural network groups (NNG); for example one group can represent the phonemes in a spoken language (one NN representing one class phoneme).
(3) Higher level decision part . This consists of several modules, each making decision on a particular problem (e.g., word recognition, face identification). The modules receive feedback from the environment and make a decision about the functioning and the adaptation of the whole ECOS.
(4) Action part. The action modules take the output from the decision modules and pass 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 is a mesh of nodes (neurons) with very few connections between them, pre-defined through *prior* knowledge or genetic information. An initial set of rules can be inserted in this structure. Gradually, through self-organisation, the system becomes more and more 'wired'. The network stores different patterns (exemplars) from the training examples. A node 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 dynamically changing parameters).

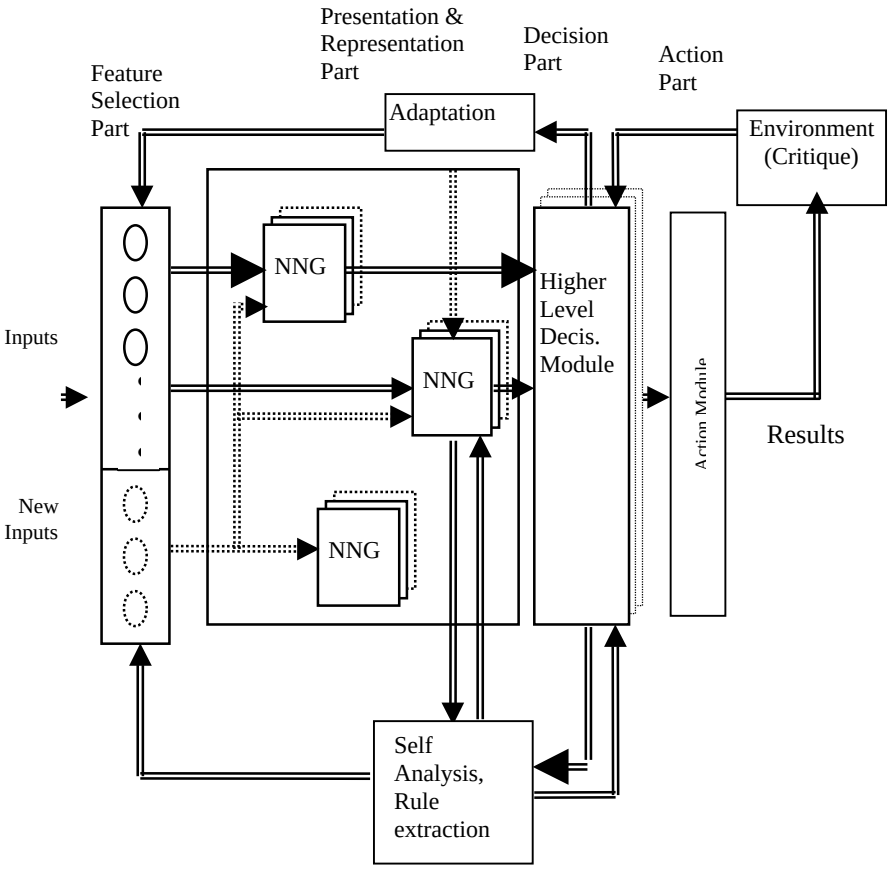

Fig.2. A block diagram of ECOS

The functioning of the ECOS from fig.2 is based on the following general principles.

(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 on e, in a pattern mode, having not necessarily the same input feature sets. After the presentation of each example, the ECOS either associates this example with an already existing rule (case) node, or creates a new one. A NN module, or a neuron is created when needed at any time of the functioning of the whole system.

(3) The representation module evolves in two phases. 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 phase two, activation from either 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-on mode) is passed to the next level of neurons.

Evolving can be achieved in both *supervised and unsupervised* modes. In a supervised mode the final decision on which class (e.g., phoneme) the current vector **x** belongs to, is made at the higher-level decision module that may activate an adaptation process. Then the connections of the representation nodes to the output nodes, and to the input nodes, are updated with the use of *learning rate coefficients lr1 and lr2,* correspondingly. If the activated output neuron (e.g., a class node) is not the desired one, then a new rule (case) node is created. The feedback from the higher-level decision module goes also back to the feature selection and filtering part. If necessary, new features may be introduced in the current adaptation and evolving phase. In an unsupervised mode a new case node is created if there is no existing case node, or existing output node, that are activated above *Sthr* and an *output threshold Othr* respectively. The parameters *Sthr, lr1, lr2, Errthr, Athr and Othr* can change dynamically during learning.

(4) An ECOS has a pruning 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. Pruning can be performed through applying the following fuzzy rule:

IF case node (j) is OLD, and the average activation of (j) is LOW, and the density of the neighbouring area of neurons is HIGH or MODERATE, and the sum of the incoming or outgoing connection weights is LOW, THEN the probability of pruning node (j) is HIGH.

(5) The case neurons are spatially organised and 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, than this node will be allocated closest to the neuron that had the highest activation to the input vector **x,** even though not sufficiently high to accommodate this input vector.

(6) There are two global modes of learning in ECOS:
(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 there is no input pattern 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 eco-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:
(i) *Cascade eco-training*; in *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. 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*; in *sleep eco training mode*, modules are created with part of the data presented (e.g., positive class examples). Then the modules are trained on the stored 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 [37,40,41].

(8) ECOS are biologically inspired. Some biological motivations are given in section 11.

(9) The ECOS framework can be applied to different types of NN (different neurons, activation functions etc.), FS, IS. One realisation of the ECOS framework is the evolving fuzzy neural network EFuNN and the EFuNN algorithm as given in [33,34,35,36] and in section 4. Before the notion of EFuNNs is presented, the notion of FuNNs is presented in the next section [37,41].

3.Fuzzy Neural Networks FuNNs

3.1. The FuNN architecture and its functionality

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist way [24,28,37,41,54,79]. FuNN is a fuzzy neural network introduced in [37,38,39,40] and developed as FuNN/2 in [41]. 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, 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.3. 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 [39,48,49].

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.

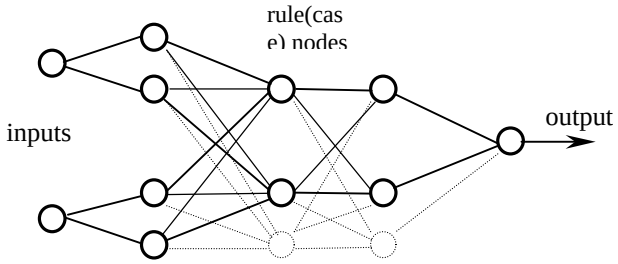

Fig. 3. A FuNN structure of 2 inputs (input variables), 2 fuzzy linguistic terms for each variable (2 membership functions). The number of the rule (case) nodes can vary. Two output membership functions are used for the output variable.

Several training algorithms have been developed for FuNN [41,44,48]:

(a) A modified back-propagation (BP) algorithm that does not change the input and the output connections representing membership functions (MF);
(b) A modified BP algorithm that utilises structural learning with forgetting, i.e. a small forgetting ingredient, e.g. 10^{-5}, is used when the connection weights are updated (see also [27,50]);
(c) A modified BP algorithm that updates both the inner connection layers and the membership layers. This is possible when the derivatives are calculated separately for the two parts of the triangular MF. These are also the non-monotonic activation functions of the neurons in the condition element layer;
(d) A genetic algorithm for training;
(e) A combination of any of the methods above used in a different order.

Several algorithms for rule extraction from FuNNs have been developed and applied [37,40,49]. One of them represents each rule node of a trained FuNN as an IF-THEN fuzzy rule.

FuNNs have several advantages when compared with the traditional connectionist systems, or with the fuzzy systems:

(a) They are both statistical and knowledge engineering tools.

(b) They are robust to catastrophic forgetting, i.e. when they are further trained on new data, they keep a reasonable memory of the old data.
(c) They interpolate and extrapolate well in regions where data is sparse.
(d) They accept both real input data and fuzzy input data represented as singletons (centres of the input membership functions))

3.2. Applications of FuNNs as both statistical and knowledge engineering tools

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: pattern recognition and classification [42,43]; dynamical systems identification and control [34,48]; modelling chaotic time series and extracting the underlying chaos rules [32,48,49], prediction and decision making [34,31].

The functioning of FuNNs is illustrated here on a case study problem of modelling and predicting the NZ SE40 stock index. The NZSE40 index is an aggregated index of the strongest NZ stock indexes. Its analysis shows that the index can be in different states at different time intervals (e.g., random, bullish, chaotic). A good prediction model should perform better than the random walk method even if the index is slightly different from a random fluctuation. A FuNN trained with the structural learning with forgetting algorithm is used for the prediction of SE40 as published in [32]. Ten time-lags have been initially set in the training data. After training with forgetting and a consecutive pruning, only four rule nodes are left, which suggests that the rest of the nodes and connections are not important for the prediction task. The results are better than the obtained by using the random walk method.

Here an experiment is presented with the use of a selected data set from the SE40 data (see http://divcom.otago.ac.nz:800/com/infosci/KEL/home.htm) - fig.4. Three input variables are used to describe the SE40 time series: (1) the change in the current day value, dS(t)= S(t) - S(t-1), (2) the change in the 10 days moving average, dMA10(t)= MA10(t) - MA10(t-1); (3) the change in the 60 days moving average, dMA60(t)= MA60(t) - MA60(t-1). The output variable is the change dS(t+1) of the NZSE40 on the next day. Five MFs for each of the variables are used. The trained FuNN has the following architecture: 3-15-10-5-1; training examples 1500; test examples 49 (taken from the last two months); epochs 1000, lr=0.1, mom=0.8. The obtained root mean square test error RMSE is 0.3, which is lower than the error of 4.32 when the random walk method is applied. After predicting the SE40 daily change, the absolute value of the SE40 can be calculated. Both the desired and the predicted values are shown in fig.4.

Nine rules are extracted from the trained FuNN using the aggregated rule extraction method. The rules are shown below where A ,B,C,D and E are the labels used to denote the five MF (very small, small, medium, large, very large) respectively, for both the input and the output variables. The fuzzy propositions have degrees of importance attached:

R1) if <Input1 is B 2.8><Input2 is B 3.5> <Input3 is C 1.1 > then <Output1 is A 1.6>; R2) if <Input1 is E 4.5> <Input2 is A 4.6><Input3 is E 2.5> then <Output1 is A 5.3> and <Output1 is B 2.5>; R3) if <Input1 is A 2.4> <Input2 is A 3.4> or <Input2 is B 4.9> <Input3 is A 3.8>then <Output1 is B 5.9> and <Output1 is C 9.6>; R4) if <Input1 is B 3.6> or <Input1 is C 2.6> <Input3 is B 1.6> or <Input3 is C 3.5> then <Output1 is D 6.9>; R5)if <Input1 is B 3.4> or <Input1 is D 3.3> <Input2 is E 6.5> <Input3 is E 1.3> then <Output1 is D 2.6>; R6) if <Input1 is B 1.7> or <Input1 is D 8.1> <Input2 is E 4.1> <Input3 is D 1.5> then <Output1 is E 4.1>; R7) if <Input1 is E 1.5> and <Input2 is A 3.9> and <Input3 is B 1.1> then <Output1 is C 4.4>; R8) if <Input1 is E 2.4> and <Input2 is C 4.4> and (<Input3 is A 1.1> or <Input3 is E 1.3>) then <Output1 is D 2.6>; R9) if <Input1 is C 9.1> and <Input2 is D 5.5 or E 2.2> and <Input3 is C 4.2 or E 3.2> then <Output1 is D 4.2> and <Output1 is E 2.7>.

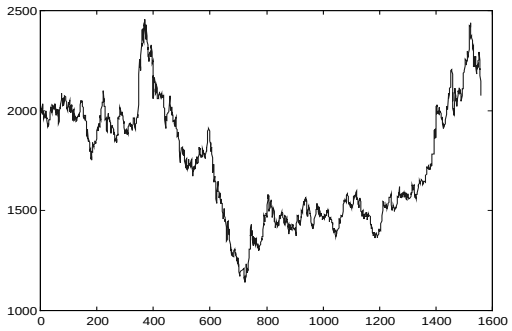

Fig.4. Using FuNN to predict the NZ SE40 index - the desired and the predicted by the FuNN values.

The average training time for FuNN p er example is 10^7 operations. FuNN is an excellent technique when used on static data, but the modified BP algorithm could be unacceptably slow when FuNNs have to be trained on very large data sets or have to be regularly re-trained to accommodate new data. This is especially true when learning with forgetting is applied. In section 10 an EFuNN-based intelligent agent is used to predict in an on-line (evolving) mode the same time series data. The learning process is much faster than the one when traditional NN techniques are used without compromising with the accuracy of the prediction results.

4. Evolving Fuzzy Neural Networks EFuNNs

4.1. A general description

EFuNNs are FuNN structures that evolve according to the ECOS principles. EfuNNs adopt some known techniques from [10,46,47,54] and from other known NN techniques, but here all nodes in an EFuNN are created during (possibly one-pass) learning. The nodes representing 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 membership functions (MF) 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 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.

The EFuNN algorithm, for evolving EfuNNs, has been first presented in [36]. A new rule node rn is connected (created) and its input and output connection weights are set as follows: W1(rn)=EX; W2(rn) = TE, where TE is the fuzzy output vector for the current fuzzy input vector EX. In case of "one-of-n" EFuNNs, the maximum activation of a rule node is propagated to the next level. Saturated linear functions are used as activation functions of the fuzzy output neurons. In case of "many-of-n" mode, all the activation values of rule (case) nodes, that are above an activation threshold of *Ahtr*, are propagated further in the connectionist structure.

4.2. The EFuNN learning algorithm

Here, the EFuNN evolving algorithm is given as a procedure of consecutive steps:

1.Initialise an EFuNN structure with a maximum number of neurons and zero-value connections. Initial connections may be set through inserting fuzzy rules in a FuNN structure. FuNNs allow for insertion of fuzzy rules as an initialisation procedure thus allowing for prior information to be used prior to the evolving process (the rule insertion procedure for FuNNs can be applied [37,41]). If initially there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons with non-zero connections, then *connect* the first node rn=1 to represent the first example EX=x_1 and set its input W1(rn) and output W2 (rn) connection weights as follows:

<*Connect a new rule node rn to represent an example EX*>: W1(rn)=EX;W2(rn) = TE, where TE is the fuzzy output vector for the (fuzzy) example EX.

2. WHILE <there are examples> DO

Enter the current, example xi, EX being the fuzzy input vector (the vector of the degrees to which the input values belong to the input membership functions). If there are new variables that appear in this example and have not been used in previous examples, create new input and/or output nodes with their corresponding membership functions.

3. Find the normalised fuzzy similarity between the new example EX (fuzzy input vector) and the already stored patterns in the case nodes j=1,2,…,rn:

Dj= sum (abs (EX - W1(j))/ 2) / sum (W1(j))

4. Find the activation of the rule (case) nodes j, j=1:rn. Here radial basis activation function, or a saturated linear one, can be used on the Dj input values i.e. A1 (j) = radbas (Dj), or A1(j) = satlin (1 – Dj).

5. Update the local parameters defined for the rule nodes, e.g. age, average activation as pre-defined.

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

7. If there is no such case node, then < *Connect a new rule node* > using the procedure from step 1.

ELSE

8. Find the rule node *inda1* that has the maximum activation value (ma*xa1).*

9. (a) in case of one-of-n EFuNNs, propagate the activation *maxa1* of the rule node *inda1* to the fuzzy output neurons. Saturated linear functions are used as activation functions of the fuzzy output neurons:

A2 = satlin (A1(inda1) * W2)

(b) in case of many-of-n mode, only the activation values of case nodes that are above an activation threshold of *Athr* ar*e* propagate to the next neuronal layer.

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 (abs(Err (inda2)) > *Errthr*) <*Connect a new rule node*>

ELSE

14. Update: (a) the input, and (b) the output connections of rule node k=inda1 as follows:

(a) Dist=EX-W1(k); W1(k)=W1(k) + lr1. Dist, 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.

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

IF *(node (j) is OLD) and (average activation A1av(j) is LOW) and (the density of the neighbouring area of neurons is HIGH or MODERATE) and (the sum of the incoming or outgoing connection weights is LOW) and (the neuron is NOT associated with the corresponding "yes" class output nodes (for classification tasks only))* THEN *the probability of pruning node (j) is HIGH*

The above pruning rule is fuzzy and it requires that the fuzzy concepts as OLD, HIGH, etc. are defined in advance. As a partial case, a fixed value can be used , e.g. a node is old if it has existed during the evolving of a FuNN from more than 60 examples.

16. END of the while loop and the algorithm

17. Repeat steps 2-16 for a second presentation of the same input data or for ECO training if needed.

5. Learning strategies for ECOS. The ECO-learning paradigm

ECOS, and EFuNN in particular, allow for different learning strategies to be experimented with, depending on the type of data available and on the requirements to the learning system. Several of them are introduced and illustrated

in this section. The EFuNN realisation of ECOS has been used in the experiments below on the benchmark Iris data set (150 instances; 3 classes - setosa, versicolour and virginica; four attributes - sepal length, sepal width, petal length, petal width). Three EFuNNs are evolved, one for each class, in each of the experiments that illustrate different learning strategies.

5.1. Incremental, one-pass learning

A FuNN is evolved with the use of both positive and negative class data. Data is propagated only once through the EuNN. The following are the characteristics of the experimentally evolved EFuNN system: "one-of-n" mode; no pruning; radial basis activation function for the rule neurons; sensitivity threshold Sthr=0.85; error threshold Errthr= 0.1; learning rate for both the first and the second layer lr=0.1; number of rule nodes evolved in each of the three modules: rn(setosa)=25; rn(versicolor)= 29; rn(virginica)=27. Overall correct classification rate: Setosa - 50 instances (100%); Versicolor - 48(96%); Virginica - 48 (96%).

5. 2. Incremental, multiple-pass learning.

A second pass on the evolved, in the experiment from 5.1 EfuNNs, is performed with the following characteristics: SThr=0.9; Errthr=0.05; rn(setosa) = 28; rn(versicolor) =37; rn(virginica)=37. Overall correct classification: Setosa - 50(100%); Versicolor - 50 (100%); Virginica - 50 (100%).

This experiment illustrates the ability of ECOS and EFuNNs to improve if more learning epochs are applied, but the number of epochs needed to achieve a certain accuracy would be orders of magnitude less than the number of epochs needed in traditional learning techniques such as the BP.

5.3. Using positive examples only.

In many practical applications only positive examples are available for a certain class and no negative examples; or the negative examples may be too many that creates problems of statistical incorrectness of the training procedure. Should 10,000 class objects (each of them represented by, say 1,000 examples) be used as negative examples to train a module to recognise just one of these objects? What if we have already trained a NN module on both positive and negative examples, but currently data has become available about a new class object and the examples about the other class objects (negative example) are lost.

Here, three EFuNNs are evolved on the Iris data by using positive examples only. The following are their characteristics: SThr=0.85; Errthr=0.05; rn(setosa) = 6; rn(versicolor) =16; rn(virginica)=20. Overall correct classification rate: Setosa - 50(100%); Versicolor - 48 (96%); Virginica - 46 (92%).

In this case only rule nodes that support the "yes" fuzzy output node have been created. That results in less number of rule nodes created (evolved), but also in a higher false positive activation of the EFuNNs when similar data, but from different classes, are presented. As a general rule, the true positive activation is higher than the false positive one. By taking the maximum activated module as the

winning class, when an input vector is presented, even better classification than the one achieved in 5.1, where both positive and negative examples were used, can be achieved.

5.4. Cascade eco-learning.

This strategy was explained in section 2. Here we assume that data, once used, is lost forever. When a new class data arrives, a new class EFuNN module is created. It starts to evolve on the positive data of this class, as well as on following negative data about other classes, and on the negative exemplars stored in the W1 connections of the already evolved EFuNNs (before this module was created).

The following are the parameters of the three evolved EFuNNs from the Iris data: SThr=0.85, Errthr=0.1; rn(setosa) = 19 (4 positive); rn(versicolor) =32 (9 positive); rn(virginica)=27 (12 positive). Overall classification: Setosa - 50(100%); Versicolor - 50 (100%); Virginica - 50 (100%).

5. 5. Sleep eco-training.

This strategy was explained in section 2. The main idea is that different modules evolve quickly to capture the most important information concerning their specialised function (e.g., class). The modules store exemplars of relevant for their functioning examples during the active training mode - when the examples are presented at the ECOS' inputs. 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 (e.g., recognition rate). During the sleep-eco training new rule nodes are created and the same evolving algorithm is used on examples (exemplars) that are not presented but rather stored in the already evolved modules. During the sleep- eco training the ECOS parameters can have different values from the values used in the active training phase, e.g., different sensitivity threshold and different learning rates.

The following are the parameters of the evolved through sleep eco-training EFuNNs for the three Iris classes: SThr=0.95; Errthr=0.05; rn (setosa) = 9; rn (versicolor) =18; rn (virginica)=22. Overall classification: Setosa - 50(100%); Versicolor - 50 (100%); Virginica - 50 (100%). The results of the sleep eco-training are better than the results after training with positive data only (see 5.3), but the significant difference is that here the false positive activation is strongly depressed and in some EFuNNs it is completely eliminated.

5.6. 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. There are two cases in the evolving procedure:
(a) There is an output node activated (by the current input vector **x)** above a pre-set threshold *Outhr*. In this case the example **x** is accommodated in the connection

weights of the most highly activated case neuron according to the learning rules of ECOS (e.g. as it is in the EFuNN algorithm).
(b) Otherwise, there will be a new rule node created and new output neuron (or new module) created to accommodate this example. The new rule node is then connected to the fuzzy input nodes and to a new output node as it is the case in the supervised evolving (e.g., as it is in the EFuNN algorithm).
Reinforcement learning uses similar procedures as case (a) and case (b) above. Case (a) is applied only when the output from the evolving system is confirmed (approved) by the 'critique' and case (b) is applied otherwise.

5.7. Evolving fuzzy systems. Rule insertions and rule extractions. On-line, adaptive learning of fuzzy rules and membership functions in EFuNNs

A fuzzy system consists of fuzzy rules (where fuzzy variables, fuzzy predicates and fuzzy sets are used) and fuzzy inference method defined. *Evolving fuzzy systems* are fuzzy systems in which the fuzzy rules evolve and change as the fuzzy system operates, thus adding new rules, modifying existing rules and deleting rules from the rule set according to changes in the environment (e.g., new data arrive regularly). Evolving fuzzy systems have also their fuzzy variables, fuzzy membership functions and predicates varying as the system operates.

A FuNN and an EFuNN in particular can be represented by a set of fuzzy rules through rule extraction techniques [37,41]. The fuzzy inference is embodied in the connectionist structure. In this respect an EFuNN can be considered as an evolving fuzzy system. The rules that represent the rule nodes need to be aggregated in clusters of rules. The degree of aggregation can vary depending on the level of *granularity* needed. Sometimes, for explanation purposes, the number of rules needed, could be as many as the number of the fuzzy output values (e.g., a rule for "No" class and a rule for "Yes" class for a classification task that uses just two output fuzzy values denoting "yes" and "no").

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 for each new rule, such as the connection weights W1 and W2 of the rule node represent the fuzzy or the exact rule.

Example1: 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 connection weights of a new rule node to the fuzzy condition nodes x1- Small and x2- Small to 0.5 each, and the connection weights to the output fuzzy node y-Small to a value of 1.

Example 2: The exact rule IF x1 is 3.4 and x2 is 6.7 THEN y is 9.5 can be inserted in the same way as in example 1, but here the membership degrees to which the input values x1=3.4 and x2=6.7 belong to the corresponding fuzzy values are calculated and attached to the connection weights instead of values of 0.5. The same procedure is applied for the fuzzy output connection weight.

Changing MF during operation may be needed for a refined performance after certain time of the system operation, for example instead of three MF the system has to change to five MF. In traditional fuzzy neural networks this change is not possible, but in EFuNNs it is possible, because an EFuNN stores in its W1 and

W2 connections fuzzy exemplars. These exemplars, if necessary, can be defuzzifyied at any time of the operation of the whole system, and than used to evolve a new EFuNN structure that has, for example, five MF rather than three MF for the input variables, and three rather than two, MF for the output variable. The idea is illustrated on fig. 5.

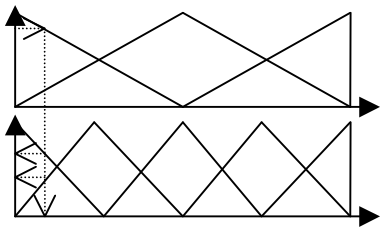

Fig.5. Changing the number of MF in EFuNN from 3 MF to 5 MF is achieved through defuzzifying the membership degrees stored as connection weights in the first EFuNN structure, and finding the membership degrees of the obtained real values to 5 MF for the new EFuNN structure, before this structure is evolved.

6. ECOS and EFuNNs for adaptive, on-line, phoneme-based spoken language recognition

Here the EFuNN algorithm is applied to the problem of phoneme recognition and phoneme adaptation.

6.1. The problem of adaptive speech recognition

Adaptive speech recognition is concerned with the development of speech recognition systems that can adapt to new speakers (of the same, or a new accent); that can enlarge their vocabulary of words in an on-line mode; that can acquire new languages [2,12,30]. There are several methods that have been experimented for adaptive phoneme recognition. One of them [42] uses phoneme FuNN modules for each class phoneme. The adaptation to a new speaker is achieved through additional training of a phoneme FuNN on new speaker's data for a few epochs. This approach to adaptive speech recognition assumes that at the higher, word recognition level, a decision is made about which phoneme module should be adapted in order to accommodate the new speaker's data and to achieve a correct word recognition. The BP algorithm was used. This method assumes a fixed number of rule nodes in the FuNNs. There are some difficulties when applying this method for on-line adaptation on continuous speech: (1) even few epochs of additional training with the use of the BP algorithm may not be fast enough for real time application; (2) in spite of the robustness of the FuNN architecture to catastrophic forgetting, a trained FuNN tends to forget old speech data if the new data differs significantly from the old one; (3) limited potential for accommodating new speech data because of the fixed size of the FuNN networks.

Here, the EFuNN algorithm is used for the purpose of phoneme adaptation of already trained EFuNNs on new accent data. In the experiments below, four EFuNNs are evolved to learn existing data on four NZ English phonemes.

Recognition results are compared with the results when ordinary FuNNs or GFuNNs (FuNNs optimised by a genetic algorithm [74]) are used. After the four phoneme EFuNNs are evolved, one of them - the phoneme /I/ module, is further evolved (adapted) to accommodate new data of the phoneme /I/ taken from a speaker of a different accent.

6.2. EFuNNS for phoneme recognition

The following phoneme data on four phonemes of New Zealand English spoken by two male and two female speakers are used in the experiment: / **I/** (taken from the pronounced word "sit" (see [69] and also the WWW page: http://) : 100 mel scale vectors, each of them consisting of 26 mel coefficients); / **e**/ (taken from "get", 170 mel scale vectors); / **ae**/ (taken from "cat", 170 vectors), and / **i**/ (taken from "see", 270 vectors). Three membership functions are used to represent "small", "medium" and "high" values for each mel-coefficient. The number of examples selected for each phoneme corresponds to the relative frequency of the appearance of these phonemes in spoken NZ English. Phonemes /e/ and / i/ have similar average mel values which makes their differentiation more difficult.

Experiment 1. **EFuNNs trained on both positive and negative data** . Four EFuNNs are evolved from the 710 input vectors. The EFuNNs have the following characteristics: linear activation function for the case (rule) nodes; saturated linear functions for the fuzzy outputs and a linear function for the class output neurons; Sthr=0.9; Errthr=0.2; no pruning; lr=0; rn(phoneme /I/) = 361 (90 for the class phoneme - positive); rn(phoneme /e/) = 395 (90 positive); rn(phoneme /ae/) = 362 (110 positive); rn(phoneme /i/) = 393 (101 positive). The following mean sum-square error is evaluated for the four phoneme modules correspondingly: 0.0085; 0.055; 0.025; 0.145. The overall correct classification rate is : /I/ - 94 examples (94%); /e/ - 131 examples (77%); / ae/ - 152 examples (90%); and / i/ - 167examples (62%). The examples that have not been classified correctly have not been miss-classified either. They did not activate any of the four EFuNNs (for them all EFuNNs had zero output values). This is a better result than in case of having misclassification (false positive activation). Here the negative examples (that do not belong to a phoneme module) are rejected with 100% accuracy in all EFuNN modules.

Experiment 2. **Using positive phoneme data only.** The same experimental setting is used as in experiment 1, but four phoneme EFuNNs are evolved with positive data only. The EFuNNs have the following characteristics: rn(phoneme /I/) = 89; rn(phoneme /e/) = 89; rn(phoneme /ae/) = 108; rn(phoneme /i/) = 101. The overall classification rate is : /I/ - 94 (94%); /e/ - 149 (87.6%); / ae/ - 154 (90.6%); / i/ - 190 (70.3%). In contrast with experiment 1, some examples that have not been classified correctly have been miss-classified , i.e. the correct classification of negative examples by the phoneme modules is different from 100%, opposite to the case in experiment 1.

Experiment 3. **Sleep eco training.** The trained in experiment 2 EFuNNs on positive data, are further trained on negative data as stored in the other EFuNN

modules (sleep eco training). The same accuracy is achieved as in EFuNNp on positive data, but here 100% accuracy is achieved on the negative data.

6.3. Comparative analysis of FuNNs, GFuNNs and EFuNNs on the phoneme recognition task

Tables 1 and 2 show the results from the above experiments and also the results when: (1) four FuNNs are 'manually' designed and trained with a BP algorithm; (2) four FuNNs are optimised with a GA algorithm and trained again with the BP as published in [74].

For the FuNN experiment, four FuNNs were 'manually' created each having the following architecture: 78 inputs (3 time lags of 26 element mel vectors each), 234 condition nodes (three fuzzy membership functions per input), 10 rule nodes, two action nodes, and one output. This architecture is identical to that used for the speech recognition system described in [42]. Nine networks were created and trained for 1000 epochs for each phoneme, the final result being the average classification result of them. A bootstrap method is used for selecting statistically appropriate data sets at every 10 epochs of training. Each trained FuNN was recalled over the same data set, and the recall accuracy calculated. For these calculations an output activation of 0.8, or greater, is taken to be a positive result, while an activation of less than 0.8 is considered as negative classification result. The mean classification accuracy of the manually designed FuNNs is presented in Table l. The manually designed networks have great difficulty in correctly identifying the target phonemes, tending instead to classify all of the phonemes presented, as negative examples (for the chosen classification threshold of 0.8).

For the GFuNN experiment a population size of fifty FuNNs was used, with tournament selection, one point crossover, and a mutation rate of one in one thousand. Each FuNN was trained with the BP algorithm for five epochs on the training data set with the learning rate and momentum set to 0.5 each. The GA was run for fifty generations, at the end of which the fittest individual was extracted and decoded. The resulting FuNN was then trained on the entire data set using the bootstrapped BP training algorithm. Each resultant network was trained for one thousand epochs, with the learning rate and momentum again set to 0.5 each, and the training data set being rebuilt every ten epochs. The GA was run nine times over each of the phonemes. The mean classification accuracy of the GA designed FuNNs is displayed in Table 1.

Overall, the best results have been obtained with the use of EFuNNs. The large number of rule nodes in the EFuNNs shows the variation between the different pronunciations of the same words by the four reference speakers. EFuNNs require 5 to 20 times more rule nodes, but at the same time they require four to six order of magnitude less time for training per example (Table 2).

6.4. On-line adaptation of phoneme EFuNNs on new accent (speaker) data

Adaptation of a phoneme module to a new speaker's data takes place when this module is identified for on-line adaptation by the higher-level decision module

according to the ECOS framework and the framework presented in [30]. Adaptation in EFuNNs is not different from its usual training (evolving) procedure. This is illustrated in the following experiment.

Table.1. True positive and true negative (in brackets) classification accuracy

	FuNN	GfuNN	EFuNN	EFuNNp
/I/	32%(98)	57% (97)	94% (100)	94% (98)
/e/	80%(94)	81% (95)	77% (100)	87% (87)
ae	52%(96)	72% (96)	90% (100)	90% (97)
/i/	5% (99)	18% (98)	62% (100)	70% (94)

Table.2 The size and the time complexity of the FuNNs, GFuNNs and EFuNNs in number of connections and in approximate time for training per example (in relative units, representing the number and the complexity of the operations)

	FuNN	GfuNN	EFuNN	EFuNNps	EfuNNsl/eco
I	$2596/18.10^7$	$616/10.10^{10}$	$28960/60.10^3$	$7200/14.10^3$	$14000/30.10^3$
e	$2596/18.10^7$	$1045/14.10^{10}$	$31680/62.10^3$	$7200/14.10^3$	$14000/30.10^3$
ae	$2596/18.10^7$	$847/11.10^{10}$	$29040/61.10^3$	$8720/15.10^3$	$17000/35.10^3$
i	$2596/18.10^7$	$946/12.10^{10}$	$31520/62.10^3$	$8160/15.10^3$	$16000/34.10^3$

Example. Adaptation of the /I/ phoneme EFuNN. The /I/ phoneme EFuNN that evolved in experiment 1, was tested on a new speaker's phoneme /I/ data taken from the pronounced by the new speaker word "sit". The new pronunciation of /I/ was significantly different from the pronunciation of the reference data used to evolve the phoneme /I/ EFuNN. Fig.6a shows the average values of each of the 26 mel scale coefficients of the reference data and the new speaker data. The /I/ EFuNN did not recognise any of the 10 new input vectors. The /I/ EFuNN was further evolved for just one pass with the use of the 10 new positive input vectors of the phoneme /I/. After that, the EFuNN increased its rule nodes from 361 to 369 and recognised 9 out of 10 new input vectors (fig.6b).

6.5. ECOS for evolving language systems

The 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 [2,52,56]. 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 trough an active interaction with a teaching environment.

It can be assumed that in a modular spoken language evolving system, the language (languages) modules evolve through using both domain text data and spoken information data fed from the speech recognition part. The language module produces final results as well as a feedback for the adaptation in the previous modules. This idea is currently being elaborated with the use of ECOS.

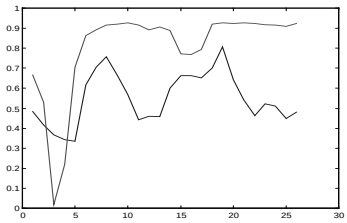

Fig.6a. The average normalised values of each of the 26 mel scale coefficients of the reference data and the new speaker data on the phoneme /I/.

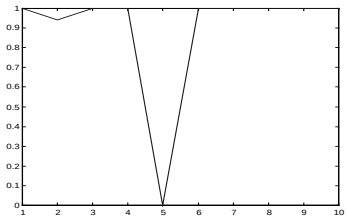

Fig. 6b. Adaptation of an already evolved EFuNN from NZEnglish phoneme /I/ data to a new accent data on the same phoneme. After a single pass of and additional evolving (adaptation) of the /I/ EFuNN on the new accent data, 9 out of 10 frames from the new accent data were correctly recognised (none of the 10 speech frames from the new accent were recognised before the adaptation took place). The y-axis shows the output activation value of the adapted /I/ phoneme EFuNN to the new accent data (10 vectors).

7. ECOS and EFuNNs for on-line, adaptive, multi-modal (speech, image, text) information processing

Several methods for multi-modal information processing that involve images (e.g., lip movement) to enhance speech recognition have been developed [23,55,73]. Other methods use speech to enhance image recognition. But when the multimodal -based recognition (or identification) process has to be performed in a real time, on-line, adaptive mode, most of the above methods would fail to achieve satisfactory results. That is because of the speed of the processing needed and a method of adaptation needed that can deal with fast adaptation to new data, some of them presented only for a very short period of time, in a noisy environment. Here, a brief reference to a framework AVIS for integrated auditory and visual information processing published in [43], is made. In sub-section two the use of ECOS and EFuNNs for the implementation of AVIS is discussed and directions for further implementations are given.

7.1. The AVIS framework for integrated auditory and visual information processing (Kasabov, Postma and Herik)

A block diagram of the AVIS framework is shown in fig.7. It consists of three main parts: auditory subsystem that processes auditory information; visual subsystem that processes visual information; and higher level decision subsystem that combines the outputs from the two subsystems. Each of the auditory and the visual sub-systems consists of five modules as described in [43]. The auditory subsystem and the visual subsystem can each operate as separate subsystems and have their results combined at a higher-level. Single-modality input streams, or multimodal input streams, can be used by each of the subsystems. Audio and visual inputs can be used by either of the subsystems, by the two subsystems simultaneously, and by the higher-level conceptual subsystem. There are several different modes of operation allowing for a comparative study of how to use visual and/or audio information in a different combination [43].

7.2. Using ECOS and EFuNNs for the implementation of AVIS

In an on-line, adaptive mode of operation each of the sub-systems and the modules of them would develop as the system operates. ECOS and EFuNNs are suitable techniques to be used for such implementation. Here a preliminary EFuNN implementation of some of the modules from the auditory sub-system are presented and discussed.

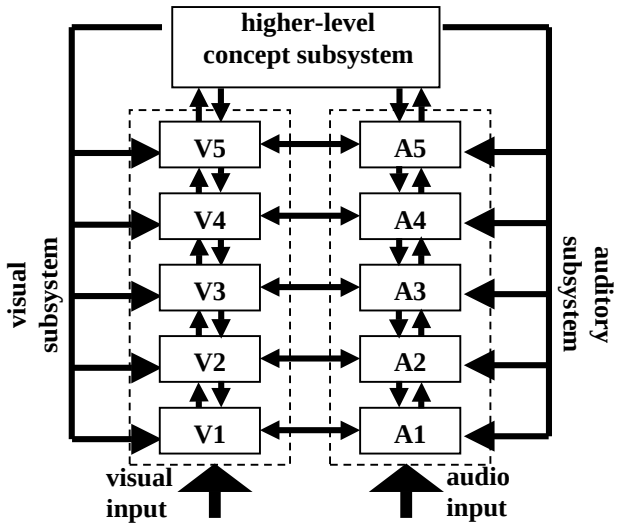

Fig. 7. A block diagram of the ***AVIS*** framework for integrated ***A***uditory - ***V***isual ***I***nformation processing ***S***ystems (adapted from [43])

The task of person identification from short movie files is implemented in [43] as a system called PIAVI that uses the AVIS framework and FuNNs for the implementation of the modules in the AVIS framework. Here the auditory data from [43] is used and four EFuNNs are evolved for the identification of four persons. 2.5 msec voice data is used as reference data for each of four speakers (news presenters of the CNN). Voce data taken from sections of 1.5 msec are used for testing. The voice data is transformed every 11.8 msec (with 50% overlap between two consecutive windows) into 26-element mel scale (MS) vectors. The 26-element MS vectors are averaged over a time frame of 125 msec thus producing 20 examples for training and 10 examples for testing for each person. The evolved EFuNNs require four to six order of magnitude less time for training per input vector than the reported in [43] experiments.

Experiment 1 . **Incremental on-line learning .** Four EFuNNs are evolved with both positive and negative data with the following parameter values: Sthr=0.9; Errthr=0.2; Person 1 EFuNN: rn=31 (8 positive); Person 2 EFuNN: rn= 35 (16 positive); Person 3 Efu NN: rn=35 (14 positive); Person 4 EFuNN: rn=29 (15 positive). Overall recognition rate: on training data - 11,16,17 and 20 examples of the corresponding person's data (80% recognition rate); on test data: 2,2,6 and 7 (43%).

Experiment2**. Changing the number of the input variables** . Two time lags of 26-element MS vectors are added to the inputs, and the EFuNNs from experiment 1 are further trained with the new 78 element input vectors. Person 1 EFuNN: rn=56; Person 2 EFuNN: rn= 60; Person 3 EFuNN: rn=56; Person 4 EFuNN: rn=59; Overall recognition: on training data - 17, 20,20 and 20 (96.25% recognition rate); on test data: 7,2,2 and 8 (48%).

Experiment 3.**Sleep eco training.** First, four EFuNNs are evolved with positive data only. Sthr=0.9; Errthr=0.2; Person 1 EFuNN: rn=15; Person 2 EFuNN: rn= 20; Person 3 EFuNN: rn=15; Person 4 EFuNN: rn=10. Overall recognition: on training data - 17,20,18,16 (89%); on test data: 7,2,2 and 6 (43%). After this initial training, the eco training is applied. The recognition rate has improved to 96% on the training data and 53% on the test data.

Further experiments on using EFuNNs for the implementation of the visual subsystem and the higher-level decision system from AVIS, are to be performed and the feasibility of using ECOS for the total AVIS implementation is to be discussed on different case studies.

8. ECOS and EFuNNs for adaptive, on-line time-series prediction, decision making and control

Here application of ECOS for on-line, adaptive time series prediction, decision making and control is discussed.

8.1. A general scheme of using ECOS and EFuNNs for on-line, adaptive prediction, decision making and control

A general block diagram of an adaptive, on-line decision making system is given in fig. 8 [31]. It consists of the following blocks:

- Pre-processing (filtering) block (e.g., checking for consistency; feature extraction, calculating moving averages, selecting time-lags for a time-series).
- ECOS block; it consists of modules that are continuously trained with data (both old, historical data, and new incoming data).
- A rule-based block for final decision - this block takes the produced by the ECOS outputs and applies expert rules. The rules may take some other input variables.
- Adaptation block - this block compares the output of the system with the desired-, or the real data, obtained over certain period of time. The error is used to adjust/adapt the evolving modules in a continuous mode.
- Rule extraction, explanation block - this block uses both extracted from the evolved modules rules, and rules from the final decision making (DM) block to explain: (1) *what* the system currently 'knows' about the problem it is solving; (2) *why* a particular decision for a concrete input vector has been made.

8.2. Applying ECOS for adaptive, on-line time series prediction

A case study problem is taken here to illustrate the potential of the ECOS and the EFuNNs for on-line adaptive prediction and control. The problem is to predict a waste water flow coming from three pumps into a sewage plant (see [37] for a description of the problem and the WWW: http://divcom.otago.ac.nz:800/com/infosci/KEL/home.htm for the data set). The flow is measured every hour. It is important to be able to predict the volume of the flow as the collecting tank has a limited capacity (in this case it is 650 cubic meters) and a sudden overflow will cause bacteria, that clean the water, to be thrown away. As there is very little data available before the control system is installed and put in operation, the control system has to be adaptive and learn the dynamics of the flow as it operates.

Here one EFuNN, that has 4 inputs, namely F(t), F(t-1), MA12h(t) and MA24h(t), and one output, F(t+1), is evolved from the time series data that consists of 500 data points. The evolved EFuNN has 397 rule nodes (Sthr=0.9; Errthr=0.05; lr=0; no pruning applied). The MSSE over a test set of the last 100 data points is 0.0068 (normalised data is used) - see fig.9. The longer the EFuNN evolves over time (more time series data points are used) the better the predicted value for next hour water flow is.

It is seen from fig.9 that in the beginning the EFuNN could not generalise well on the next hour flow, but after learning (accommodating) about 400 data points, it produces a generalisation that is much better than the generalisation on the same test data when a MLP is used that had 5 inputs, two hidden layers with 6 nodes in each of them, and one output, trained with the BP algorithm for 20,000 epochs (MSSE=0.044, see [37]). The EFuNN required 4 orders of magnitude less time for training per example at average, than the MLP.

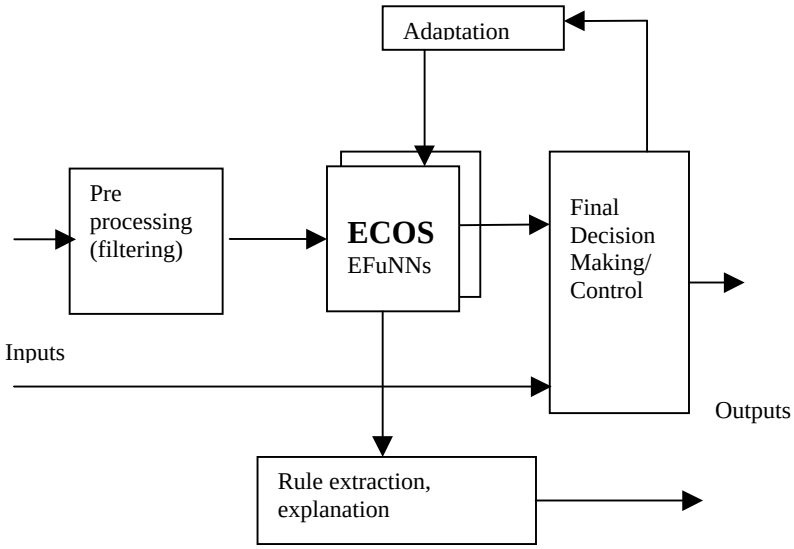

Fig.8. A block diagram of an intelligent, adaptive, on-line system for prediction, decision making and control

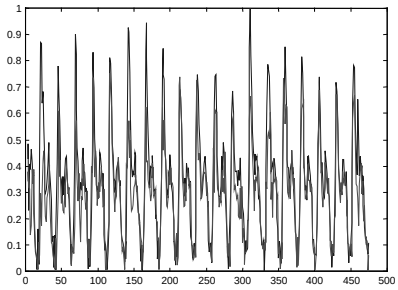

Fig.9. An EFuNN is evolved from the flow F(t), the previous hour flow F(t-1), the moving average 12 hours MA12h(t) and the moving average 24 hours MA24h(t) data. Here the real values F(t+1) and the predicted by the evolving EFuNN for one day ahead are plotted as the EFuNN evolves step by step through the time-series data.

9. ECOS and EFuNNs for evolving intelligent agents

9.1. The evolving, intelligent agents paradigm

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 [77]. 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. Such adaptation is crucial for a mobile robot navigation, or for an adequate decision making on operations with a dynamically changing stock index [6,7,13,31]. Here the latter example is used to illustrate the potential for using ECOS and EFuNNs for building intelligent agents.

9.2. EFuNN-based intelligent agents for adaptive, on-line prediction of the NZ SE40 stock index

Here an intelligent agent that learns and predicts in an on-line, adaptive mode the SE40 data is realised as an EFuNN [31]. The same input and output variables are used as in the experiment with the NZSE40 data in section 3. The following evolving parameter values are used: sensitivity threshold Sthr=0.92, error threshold Errthr=0.05, number of rule nodes rn=910; after pruning this number is 730; learning rate lr=0. The SE40 daily change is predicted on-line based on the evolving of the EFuNN on the previous data. The root mean square error is RMSE= 0.22 (on the last 49 test data points) while the random walk gives RMSE=4.32.

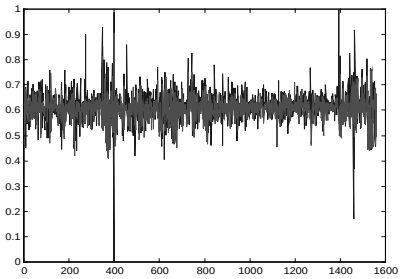

Fig.10. An EFuNN evolved as an intelligent agent and tested incrementally on the NZ SE40 difference data

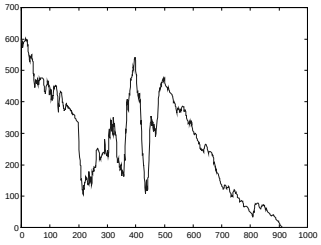

Fig.11. The total activation of the rule nodes of the EFuNN evolved as an intelligent agent from the NZ SE40 data

Fore more EFuNNs were evolved to predict two, three, four and five days ahead. The test error RMSE for them is correspondingly 0.25, 0.28, 0.45, 1.26, 2.78. It can be seen that even 5 days prediction will give a better result than the random walk one-day prediction. That justifies the use of EFuNNs for this particular task. As EFuNNs have principally the same structure as FuNNs, fuzzy rules can be extracted as explained in section 3. Fig.11 shows the total activation of the rule (case) nodes of the evolved EfuNN before pruning.

10. Recurrent EFuNNs

A recurrent EFuNN (REFuNN) structure has feedback connections from its outputs back to its inputs. In the EFuNN-based ECOS, adaptation take place when a signal from the higher level decision making is passed to the lower level modules (e.g., EFuNNs). A block diagram of a REFuNN structure is shown in fig. 12. It consists of the same input-, fuzzy condition element-, and rule(case) layers as the feed-forward EFuNN, but it has also a state layer and an action layer. There are feedback connections from the state layer to the rule layer, so which rule node will be the highest activated for a certain input, depends not only on the input vector but on the state the REFuNN is in. The connection weights from the state to the action (output) nodes can be learned through reinforcement learning where the awards are attached as positive connection weights and the punishments - as negative connection weights.

REFuNNs can be used in mobile robots that learn and evolve as they operate. They are suitable techniques for the realisation of intelligent agents when supervised, unsupervised, or reinforcement learning is applied at different stages of the system's operation.

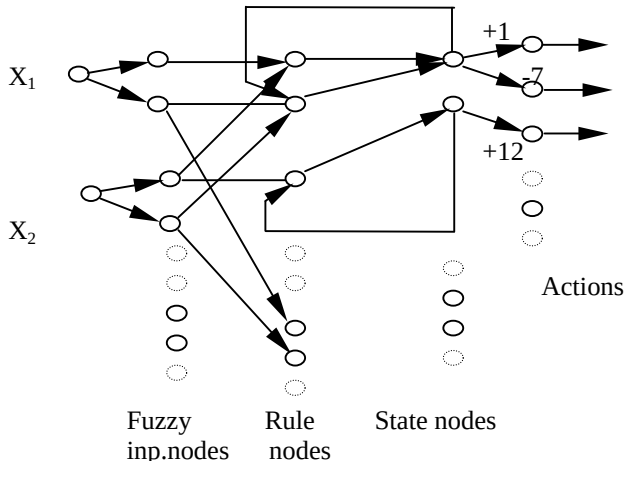

Fig. 12. Recurrent EFuNN

11. Biological motivations for the ECOS development: evolving brains

It is known that the human brain develops even before the child is born. During learning the brain allocates neurons to respond to certain stimuli and develops their connections [5,45,61,68]. The process of evolving is based on several principles, some of them listed here: (a) evolving is achieved through both genetically defined information and learning; (b) the evolved neurons have a spatial-temporal representation where similar stimuli activate close neurons; (c) the evolving process leads to a large number of neurons involved in each task similar to the activation of the brain where many neurons are allocated to respond to a single stimulus, or to perform a single task; e.g. when a word is heard, there are hundreds of thousands of neurons that get immediately activated; (d) memory-based learning, i.e. the brain stores exemplars of facts that can be recalled at a later stage; (e) evolving through interaction with the environment and with other brains; (f) inner processes take place; (g) the evolving process is continuous, lifelong; (h) through evolving brain structures (neurons, connections), higher-level concepts emerge that are embodied in the structure, but can also be represented as a level of abstraction (e.g., acquisition and the development of speech and language, especially in multilingual subjects).

The learning and the struc tural evolution coexist in ECOS. That is plausible with the co-evolution of structure and learning in the brain. The neuronal structures eventually implement a long- term memory. Biological facts about growing neural network structures through learning and adaptation are presented in [3,5,29,45,61,68, 71,75,78].

The observation that humans (and animals) learn through memorising sensory information and then remembering it when interpreting it in a context-driven way belongs to Helmholtz (1866) [72]. This is demonstrated in the consolidation principle that is widely accepted in physiology. It states that what has happened in the first 5 or so hours after presenting input stimulus the brain is learning to 'cement' what has been learned. This has been used to explain retrograde amnesia (a trauma of the brain that results in loss of memory about events that occurred several hours before the event of the trauma).

The above biological principles are presented in ECOS in the form of eco-training mode. During the ECOS learning process, one exemplar (or pattern) is stored in a long-term memory (a pathway from the presentation part to the higher-level decision part). Using stored patterns in the eco-training mode is similar to the Task Rehearsal Mechanism (TRM). The TRM assumes that there are long term and short term centers for learning [56]. "The TRM relies on long-term memory for the production of virtual examples of previously learned task knowledge (background knowledge). A functional transfer method is then used to selectively bias the learning of a new task that is developed in short-term memory. The representation of this short-term memory is then transferred to long-term memory where it can be used for learning yet another new task in the future. Notice, that explicit examples of a new task need not be stored in long-term memory, only the

representation of the task which can be later used to generate virtual examples. These virtual examples can be used to rehearse previously learned tasks in a concert with a new 'related' task". But if a system is working in a real-time mode, it may not be able to adapt to new data if its speed of processing is 'too, when compared to the speed of the continuously incoming information. This phenomenon is known in psychology as "loss of skills". The brain has a limited amount of working or short term memory. And when encountering important new information, the brain stores it simply by erasing some old information from the working memory. The prior information gets erased from the working memory before the brain has time to transfer it to a more permanent or semi-permanent location for actual learning. These issues are also discussed in [63,76].

12. Problems with ECOS and EFuNNs and directions for further research

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. These include finding the optimal values for the evolving parameters, such as the sensitivity threshold Sthr, the error threshold Ethr, learning rate lr1 and lr2, forgetting rate. Also, pruning of fuzzy rule needs to be made specific for every application, thus depending on the definition of age and the other fuzzy variables in the pruning rule. One way to overcome the above problems is to regularly apply genetic algorithms and evolutionary computing as optimisation procedures to the ECOS and EFuNN structures. Introducing DNA computing as part of the evolving process may also be beneficial [65].

Evolving connecti onist systems could be considered as a *new AI paradigm.* They incorporate the following AI features: learning; reasoning; knowledge manipulation; knowledge acquisition ; adaptation. This sets new tasks for the ECOS future development that are relevant to the current AI methods and systems, such as implementing in ECOS non-monotonic and temporal reasoning, optimal dimensionality spatial representation, meta-knowledge and meta-reasoning. More theoretical investigations on the limitations of ECOS are needed.

13. Conclusions

This chapter presents a framework ECOS for evolving connectionist and fuzzy connectionist systems, and evolving fuzzy neural networks EFuNN, in particular, for building on-line, adaptive learning systems. Real problems have been used to illustrate the potential of this approach. Several applications of ECOS and EFuNNs have been demonstrated in the chapter, namely: adaptive speech recognition; financial applications; multi-modal information processing and integrated audio and video information processing; intelligent agents. ECOS have features that address the seven major requirements to the next generation neuro-fuzzy techniques presented in section one.

Acknowledgements

This work was done as part of my sabbatical leave from the University of Otago in 1998. I developed the ECOS paradigm and conducted experiments while I was visiting the following universities: University of Trento (Italy), University of Maastricht (The Netherlands), University of Twente (The Netherlands), University of Essex (UK). I would like to thank my colleagues Prof. Mario Fedrizzi, Prof. Jaap van den Herik, Dr. Eric Postma, Dr. Leo Pluhar, Prof. Anton Nijholt, Dr Marc Drossaers, Dr. Mannes Poel, Dr. Mehmet Aksit, Dr Anne De Roeck, Dr Jeoff Reinolds, Prof. Ray Turnur, Mr Grahame Clarke, and other colleagues, for the research conditions they offered me during my stay in their respective departments and for the fruitful discussions I had with them. I hope the introduced here ECOS techniques will be useful to apply to research problems, currently under investigation in these departments. This research is also part of a research programme funded by the New Zealand Foundation for Research Science and Technology.

References

1. Almeida,L., T. Langlois, J. Amaral, J. On-line Step Size Adaptation, Technical R eport, INESC RT07/97, 1997
2. Altman, G., *Cognitive Models of Speech Pro cessing*, MIT Press, 1990
3. Amari, S. and Kasabov, N. eds (1997) Brain-like computing and intelligent information systems, Springer Verlag
4. Andrews, R., J. Diederich, A.B.Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks", Knowledge-Based Systems, 8, 373-389 (1995).
5. Arbib, M. (ed) (1995) *The Handbook of Brain Theory and Neural Networks.* The MIT Press
6. Baestalus, Dik-Emma, van den Bergh, W.M.,Wood, D. Neural network solutions for trading financial market,Pitman Publications, 1994
7. Beltraffi, A., Margarita, S., Terna, P. Neural networks for economics and financial modelling, Int. Thomson Computer Press, 1996
8. Carpenter, G. and Grossberg, S., Pattern recognition by self- organizing neural networks , The MIT Press, Cambridge, Massachusetts (1991)
9. Carpenter, G.A. and Grossberg, S., ART3: Hierarchical search using chemical transmitters in self-organising pattern-recognition architectures, Neural Networks, 3(2) (1990) 129-152.
10. 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
11. Chauvin, L, A backpropagation algorithm with optimal use of hidden units, Advances in Neuro Information Processing Syste, 1 (1989) 519-526.
12. Cole, R., et al. The Challenge of Spoken Language Systems: Research Directions for the Nineties, IEEE Transactions on Speech and Audio Processing, vol.3, No.1, 1-21, 1995
13. DeBoeck, L. Trading on the edge. Kluwer Academics, 1994

14. DeGaris, H. , Circuits of Production Rule - GenNets – The genetic programming of nervous systems, in: Albrecht, R., Reeves, C. and N. Steele (eds*) Artifical Neural Networks and Genetic Algorithms*, Springer Verlag (1993)
15. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992)
16. 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).
17. Freeman, J.A.S., Saad, D., On-line learning in radial basis function networks, Neural Computation vol. 9, No.7 (1997)
18. Fritzke, B., A growing neural gas network learns topologies, Advances in Neural Information Processing Systems, vol.7 (1995)
19. 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)
20. 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
21. Goldberg, D.E., *Genetic Algorithms in Search, Optimisation and Machine Learning* , Addison-Wesley (1989)
22. Goodman, R.M., C.M. Higgins, J.W. Miller, P.Smyth, "Rule-based neural networks for classification and probability estimation", Neural Computation, 14, 781-804 (1992)
23. Gray, M.S., J.R.Movellan,, and T.J.Sejnowski, Dynamic features for visual speech reading: A systematic comparison. In M.C. Mozer, M.I. Jordan, and T. Petsche (Eds.), Advances in Neural Inform. Proc. Systems, Vol.9, pp.751-757. Morgan- Kaufmann: San Fransisco, CA, 1997.
24. 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.
25. Hassibi and Stork, Second order derivatives for network pruning: Optimal Brain Surgeon, in: Advances in Neural Information Processing Systems, 4, (1992) 164-171
26. Heskes, T.M., Kappen, B. (1993) On-line learning processes in artificial neural networks, in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233
27. Ishikawa, M. (1996) "Structural Learning with Forgetting", Neural Networks 9, 501-521.
28. Jang, R. (1993) ANFIS: adaptive network-based fuzzy inference system, *IEEE Trans. on Syst.,Man, Cybernetics*, 23(3), May-June 1993, 665-685
29. Joseph, S.R.H. Theories of adaptive neural growth, PhD Thesis, University of Edinburgh, 1998
30. Kasabov, N. A framework for intell igent conscious machines utilising fuzzy neural ne tworks and spatial temporal maps and a case study of multilingual speech recognition", in: Amari, S. and K asabov, N. (eds) *Brain-like computing and intelligent information systems* , Springer, 106-126 (1997)
31. Kasabov, N. and Fedrizzi, M. (1999) Fuzzy Neural Networks and Evolving Connectionist Systems for Intelligent Decision Making, Proc. of IFSA'99, Taiwan , 1999, submitted.
32. Kasabov, N**.** and Kozma, R. Multi-scale analysis of time series based on neuro-fuzzy-chaos methodology applied to financial data. In: Refenes, A., Burges, A. and Moody, B. eds. *Computational Finance 1997*, Kluwer Academic, 1998, accepted
33. Kasabov, N. ECOS: A framework for evolving connectionist systems and the eco learning paradigm,Proc. of ICONIP'98, Kitakyushu, Oct. 1998

34. Kasabov, N. Evolving connectionist and fuzzy connectionist system for on-line decision making and control, Proc. of the 3rd On-line WWW World Congress on Soft Computing in Engineering Design, June 1998, Springer Verlag, to appear.
35. Kasabov, N. Evolving connectionist and fuzzy connectionist systems, *IEEE Transactions on Man, Machine and Cybernetics*, submitted
36. Kasabov, N. Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation, in Proc. of Iizuka'98, Iizuka, Japan, Oct.1998
37. Kasabov, N.(1996) *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, The MIT Press, CA, MA.
38. Kasabov, N., "Adaptable connectionist production systems". Neurocomputing, 13 (2-4) 95-117 (1996).
39. 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).
40. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2-20 (1996).
41. 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)
42. Kasabov, N., Kozma, R., Kilgour, R., Laws, M., Taylor, J., Watts, M. and Gray, A. "A Methodology for Speech Data Analysis and a Framework for Adaptive Speech Recognition Using Fuzzy Neural Networks and Self Organising Maps, in the same volume.
43. Kasabov, N., Postma, E., and Van den Herik, J AVIS: A Connectionist-based Framework for Integrated Audio and Visual Information Processing, in Proc. of Iizuka'98, Iizuka, Japan, Oct.1998
44. Kasabov, N., Watts, M. 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)
45. Kater, S.B., Mattson, N.P., Cohan, C. and Connor, J. Calcium regulation of the neuronal cone growth, Trends in Neuroscience, 11 (1988) 315-321.
46. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, vol.78, N-9, pp.1464-1497.
47. Kohonen, T. ,. *Self-Organizing Maps*, second edition, Springer Verlag, 1997
48. Kozma, R. and Kasabov, N Generic neuro-fuzzy-chaos methodologies and techniques for intelligent time-series analysis. In: *Soft Computing in Financial Engineering.* R. Ribeiro, R.Yager, H. J. Zimmermann and J. Kacprzyk eds. Heidelberg, Physica-Verlag (1998)
49. Kozma, R. and N.Kasabov, Rules of chaotic behaviour extracted from the fuzzy neural network FuNN, Proc. of the WCCI'98 FUZZ-IEEE International Conference on Fuzzy Systems, Anchorage, May (1998).
50. Kozma, R., M. Sakuma, Y. Yokoyama, M. Kitamura, On the Accuracy of Mapping by Neural Networks Trained by Backpropagation with Forgetting, Neurocomputing, 13 (2-4) (1996).
51. Krogh, A. and Hertz, J.A., A simple weight decay can improve generalisation. Advances in Neural Information Processing Systems, 4 (1992) 951-957
52. Lawrence, S., Fong, S., Giles, L. natural language grammatical inference: A comparison of recurrent neural networks and machine learning methods, in: S.Wermtner, E.Riloff and G.Scheler (eds) Symbolic, Connectionist and Statistical;

Approaches to Learning for Natural language processing, Lecture Notes in AI, (1996) 33-47

53. Le Cun, Y., J.S. Denker and S.A. Solla, Optimal Brain Damage, in: D.S. Touretzky, ed., *Advances in Neural Information Processing Systems* , Morgan Kaufmann, 2, 598-605 (1990).
54. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).
55. 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.
56. McClelland, J., B.L. McNaughton, and R.C. Reilly (1994) "Why there are Complementary Learning Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memeory", CMU Technical Report PDP.CNS.94.1, March
57. 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).
58. Mitchell, M.T. "Machine Learning", MacGraw-Hill (1997)
59. Mitchell, Melanie, *An Introduction to Genetic Algorithms* , MIT Press, Cambridge, Massachusetts (1996).
60. 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).
61. Quartz, S.R., and Sejnowski, T.J. The neural basis of cognitive development: a constructivist manifesto, Behavioral and Brain Science, to appear
62. Reed, R. (1993) Pruning algorithms - a survey, IEEE Trans. Neural Networks, 4 (5) 740-747.
63. Robins, A. Consolidation in neural networks and the sleeping brain *, Connection Science*, 8, 2 (1996) 259-275
64. Rummery, G.A. and Niranjan, M., On-line Q-learning using connectionist systems, Cambridge University Engineering Department, CUED/F-INENG/TR 166 (1994)
65. Sanchez, E. , DNA Biosoft Computing, in: Methodology for the Conception, design, and Appication of Intelligent Systems, Proc. Iizuka'96, 30-37
66. Sankar, A. and R.J. Mammone, Growing and Pruning Neural Tree Networks, IEEE Trans. Comput. 42(3) 291-299 (1993).
67. Schiffman, W., Joost, M. and Werner. R., Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons. In: Albrecht, R.F., Reeves, C. R., Steele, N. C. (Eds.), *Artificial Neural Nets and Genetic Algorithms* , Spring-Verlag Wien, New York (1993)
68. Segev, R. and E.Ben-Jacob, from neurons to brain: Adaptive self-wiring of neurons, TR, Faculty of Exact Sciences, Tel-Aviv University (1998)
69. Sinclair, S., and Watson, C., 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).
70. Towel, G., J. Shavlik, J. and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks", Proc. of the 8 th National Conf. on Artificial Intelligence AAAI'90, Morgan Kaufmann, 861-866 (1990).
71. Van Ooyen, A. Activity-dependent neural network development, Network: Computation in Neural Systems, 5 (1994) 401-423.
72. von Helmholtz, H., Handbuch der Physiologishe Optic, Hamburg and Leipzig: Voss, 1866, 1896
73. Waibel, A., M.Vo, P.Duchnovski, S.Manke, " Multimodal Interfaces", Artificial Intelligence Review, 1997

74. Watts, M., and Kasabov, N. Genetic algorithms for the design of fuzzy neural networks, in Proc. of ICONIP'98, Kitakyushu, Oct. 1998
75. 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.
76. Winson, J. The meaning of dreams, Scientific American, November (1990) 42-48
77. Woldrige, M. and Jennings, N. Intelligent agents: Theory and practice, The Knowledge Engineering review (10) 1995
78. Wong, R.O.L. Use, disuse, and growth of the brain, Proc. Nat. Acad. Sci. USA, 92 (6) (1995) 1797-99.
79. 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
80. Zadeh, L. 1965. Fuzzy Sets, *Information, and Control*, vol.8, 338-353.