Fuzzy Neural Networks and Evolving Connectionist Systems for Intelligent Decision Support

Nikola Kasabov[1] and Mario Fedrizzi[2]

[1] Department of Information science, University of Otago,
P.O.Box, Dunedin, New Zealand, **nkasabov@otago.ac.nz**,

[2] Department of Informatics and Management, University of Trento,
via Inama 5, 38100 Trento, Italy, fedrizzi@unitn.it

Abstract.
The paper presents a general framework of connectionist-based, intelligent decision support systems and its realisation with the use of fuzzy neural networks FuNNs and evolving fuzzy neural networks EFuNNs. FuNNs and EFuNNs facilitate learning from data, fuzzy rule insertion, rule extraction, and adaptation. Several applications of this framework on real problems are presented as case studies, that include classification tasks (e.g., classifying applicants for a bank loan) and on-line prediction tasks (stock index and risk assessment). The latter requires adaptive, incremental, on-line learning when the decision-making system has to work in a real-time mode. This paper suggests that fuzzy neural networks and evolving fuzzy neural networks are suitable tools to use for the implementation of the general framework.

Keywords: fuzzy neural networks, financial data analysis, decision support systems, adaptive learning, on-line learning.

1. Introduction

The complexity of many real-world problems, especially in Economics and Finance, require using sophisticated methods and tools when building intelligent decision support systems (IDESS). The methods and tools used should be able to:

- learn quickly from large amount of historical data;
- learn in an incremental, on-line mode, when the system is used for real-time decision support;
- accommodate both data and *a priori* knowledge, e.g., existing rules;
- produce explanation about the system's functioning and performance (e.g., extract rules);
- adjust to changes in the operating environment, introducing new variables and features if needed without the need to re-train the whole system on both old and new data;

There are different methods and tools that have been used so far to build decision support systems. They include: rule-based systems [4]; neural networks [1,2,4]; fuzzy systems [4]; genetic algorithms [4], fuzzy neural networks, and hybrid systems [8,14]. These systems are usually concerned only with part of the above requirements.

Using neural networks (NN), and especially fuzzy neural networks (FNN), is a promising approach towards building IDESS. What is needed is a framework that facilitates using these techniques in a comprehensive and coherent way to meet all the above requirements.

This paper presents a general framework of connectionist-based intelligent decision making systems (CB-IDESS), that addresses all the above requirements, along with its realisation with the use of fuzzy neural networks FuNN [12,13] and evolving fuzzy neural networks EFuNNs [9,11]. Several applications of this framework on real problems are presented as case studies, that include classification tasks (classifying applicants for a bank loan) and financial assessment tasks (stock index and risk assessment). The paper suggests that fuzzy neural networks and evolving fuzzy neural networks are suitable tools to use when realising the general framework.

2. A general framework for connectionist - based intelligent decision support systems (CB-IDESS)

A block diagram of a CB-IDESS is given in fig.1. It consists of the following blocks:
Pre-processing (filtering) block: data is processed (filtered) in this block (e.g. checking for consistency; feature extraction; calculating moving averages; selecting time-lags from a time-series).
Neural network, multi-modular, learning block: it consists of many NN that are continuously trained on data (both old, historical data, and new incoming data).
(3) Rule-based block for final decision: this block takes the produced by the NN outputs and applies expert rules. The rules take into account some other variables as inputs (e.g. economic situation, political situation) for which there might not be historical data available.
(4) Adaptation block: this block compares what the CB-IDESS 'suggests' as decision with the desired, or the real data obtained over a time period. The error is used to adjust/adapt relevant NN modules.
(5) Rule extraction, explanation block: this block uses both extracted from the NN modules rules and rules from the final block to explain: (1) *what* the CB-IDESS 'knows' about the problem it is designed to solve; (2) *why* a

particular decision for a concrete input vector has been made.

The framework allows for comprehensive information processing and decision making, including: feature selection, modelling, final multivariable decision-making, rule extraction and explanation, and adaptation to changes in the operating environment and to new incoming data.

In order to realise different blocks of the framework for a certain task, different techniques can be used. In the next sections we present the fuzzy neural networks FuNN and demonstrate that FuNNs can realise any of the five functions above and are suitable tools for the overall realisation of the framework from fig.1. That does not exclude of course using other techniques for the realisation of the modules in the framework.

3. Fuzzy neural networks FuNNs

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist manner [7,16,12,13]. They have the following properties:

- They implement fuzzy rules and fuzzy inference in a connectionist way.
- They have all the properties of neural networks (e.g., learning, generalisation).
- They have several modes of operation: training; rule insertion; rule extraction; adaptation to new data.

The fuzzy neural network FuNN is a connectionist feed-forward architecture that has five layers of neurons and four layers of connections (see fig.2). 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, or large. The MF can be kept fixed, or can change during training. 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 values for the output variables. A FuNN has both the features of a neural network and a fuzzy inference machine. Several training algorithms have been developed for FuNNs [12,13]:

(a) A modified back-propagation (BP) algorithm that does not change the input and the output connections representing the membership functions.

(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.

(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 membership functions that form a non-monotonic activation function of the neurons in the condition element layer.

(d) a genetic algorithm;

(e) a combination of methods from above.

Several algorithms for fuzzy rule extraction from FuNN have been developed and applied [12,13,8,14]. They are based on two different approaches:

(1) a rule represents a combination of input variables with their fuzzy values that can trigger activation of output variables. Rules can be either weighted rules (they have weights attached to the condition and conclusion parts representing their relative importance within the rule), or simple rules (they do not have any weights attached);

(2) a rule represents the connections of a rule node in a trained FuNN. Such rules are called aggregated rules.

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

(a) They can be trained to approximate data as the NNs can;

(b) They can be used to deal with knowledge in the form of fuzzy rules (insert, extract, modify rules);

(c) They are robust to catastrophic forgetting, i.e. when further trained only on new data, they keep a reasonable memory of the old data;

(d) They can be used as replicators, where same input data is used as output data during training; in this case the rule nodes perform an optimal encoding of the input space;

(e) They can work on both real input data and fuzzy input data represented as singletons (centres of gravity of the input membership functions);

(f) When structural learning with forgetting and a consecutive pruning is applied, the FuNN structure becomes a skeleton structure that contains only the important input, rule and output nodes and the important connections between them. In this respect FuNNs are tools that can be used for feature selection.

Generally speaking, FuNNs are both statistical and knowledge engineering tools. They can be used to realise each of the functions of the framework from fig.1: feature selection - features (e), (f) and (b) from above; modeling - (a); final multivariable rule-based decision-making - rule insertion as pointed in (b); rule extraction and explanation - (b); adaptation to changes in the operating environment and to new incoming data - feature (c).

The issue of on-line adaptation is crucial when the system has to learn high-frequency data 'on the fly' (e.g., to accommodate new data every minute). The already trained FuNN systems can either be trained for a few iterations on the new data, in which case forgetting of old data is inevitable, or the new data is added to the last part of the old data (e.g., the last 6 months of a stock index data) and then the FuNN is trained on the new data set. In this case the oldest part of the data is removed and the new data is added before

the FuNN is further trained with the new training data set.

Here, growing FuNNs (the rule nodes grow with the number of the data examples), along with shrinking ones (FuNNs trained with the method of structural learning with forgetting [6,15]) are appropriate to use. In section 6 the principles of evolving connectionist systems (ECOS) and evolving FuNNs (EFuNNs) are presented as introduced in [9,10,11]. In section 7 EFuNNs are used for adaptive learning and assessment of a stock index data.

4. Fuzzy neural networks and hybrid systems for classification and prediction: two case studies on a bank loan approval task

FuNNs are suitable tools for solving classification or prediction tasks for the following reasons:
1) FuNNs, being multi-layer perceptrons, learn posterior class probability from data;
2) FuNNs can be initialised through rule insertion with initial (a priory) classification rules if such rules are available prior to (or instead of) using data;
3) FuNNs can be used to extract fuzzy rules from classification data that explain the process of classification.

The three points above are illustrated on two case studies of the bank loan approval task. The first one classifies applicants for a loan in two groups (loan approved, loan rejected). The second one predicts the amount that an applicant will be granted.

Case study 1.

A FuNN is trained on 90 data examples for decision making on mortgage approval (bank loan) extracted from a database available from the WWW site: http://divcom.otago.ac.nz:800/com/infosci/KEL/. The following attributes are used: Input1: character (0- doubtful; 1 - good); Input2: total asset; Input3: equity; Input4: mortgage loan; Input5: budget surplus; Input6: gross income; Input7: debt servicing ratio;Input8: term of loan; Output: decision (disapprove; approve). Two MFs are used both for the input and for the output variables. Ten FuNNs are trained with the modified BP algorithm, fixed MFs, for 3500 epochs each, on 10 different sets of data, each of them containing 10 positive (applications are approved) and 10 negative (applications are rejected) examples, and tested on another data of 5 positive and 5 negative examples. The learning rate is lrate=0.1 and the momentum - 0.8. The average root mean square error RMSE across the ten FuNNs is 0.02. The average classification rate on the test data is 100% for the reject class and 95% for the accept class when a threshold of 0.6 is used to distinguish the reject from the accept class activation of the output of the FuNN.

Weighted rules are extracted from the trained FuNN (after a threshold of 1 is applied) that are equivalent to the aggregated rules extracted from the same FuNN:

r1) if <Input1 is A 2.2> and <Input2 is B 1.0> and <Input3 is A 1.2> and <Input5 is A 2.7> and <Input8 is B 1.0> then <Output1 is A 1.9> ;
r2) if <Input1 is B 1.8> and <Input2 is B 2.0> and <Input4 is B 1.4> and <Input5 is B 2.1> then <Output1 is A 2.8>;
r3) if <Input1 is B 4.3> and <Input2 is A 2.0> and <Input3 is B 2.7> and <Input4 is B 3.6> and <Input5 is B 6.2> and <Input7 is B 1.8> then <Output1 is B 5.0>.

The rule extraction part of this experiment is a very important part of the modelling process. The rules above point to some conclusions:
(1) Input 6 is not present in the rules, therefore it is not significant for decision support process.
(2) The most important variables are inputs 1 and 5;
(3) In order to better differentiate some cases, it is necessary to use more MFs for the representation of some of the input variables, especially input 5. In this case the FuNN classifier works in an off-line mode with static data, its further adaptation can be achieved through adding new examples to the old ones and retraining the FuNN on the whole data, or through additional training of the trained FuNN on the new data for a few epochs accounting for the forgetting phenomenon.

Case study 2.

This is a similar case study to the case study 1, but here different set of attributes is used on the loan approval task and the task is the one of prediction - to predict the amount that will be granted. The data used is owned by a bank in Italy (Trento). The following attributes are used to make decision on loan applications for each of the destinations car, furniture, maintenance, health, travelling, household, repair of a car, sport equipment, clothes, hobby equipment: 1) credit asked; 2) income spendable by the applicant 3) patrimony of the applicant; 4) married or not; 5) job situation; 6) bank account ;7) experience; 8) previous rejection record; 9) risk of the application; 10) income spendable by the guarantor; 11) patrimony of the warrantor; 12) type of warranty; 13) risk of warrantor. Output: amount granted.

In this experiment a FuNN 13-39-3-3-1 is trained and tested. The following rules are extracted that explain the decision on loan applications for buying cars (here *A* denotes "small", *B*- "medium", *C*- "high"):

r1) if <Input4 is B 0.6> <Input5 is B 0.6> <Input6 is A 0.9> or <Input6 is B 0.9> <Input7 is A 3.3> <Input8 is A 1.1> <Input9 is not A 1.3> or <Input9 is B 0.6> <Input10 is A 1.9> <Input11 is A 1.2> <Input13 is A 1.4> then <Output1 is A 4.5>;

r2) if <Input1 is A 0.8> <Input1 is B 0.7> <Input2 is 1.3> <Input3 is A 1.4> <Input5 is A 1.1> <Input6 is A 0.8>> <Input6 is B 1.4> <Input7 is A 0.7> or <Input7 is B 0.7> <Input8 is A 0.5> <Input9 is C 0.7> <Input10 is A 0.7> or <Input10 is B 1.5><Input11 is A 1.3> <Input13 is B 1.1> then <Output1 is A 3.1> or <Output1 is B 4.5>;
r3) if <Input2 is A 1.1> <Input3 is B 0.9> <Input4 is C 0.7> <Input5 is A 1.3> or <Input5 is B 1.1> <Input7 is B 2.0> or <Input7 is C 0.6> <Input8 is C 0.7> Input9 is A 0.7> <Input10 is B 0.8> <Input11 is B 0.8> <Input12 is C 0.4> <Input13 is C 1.0> then <Output1 is C 2.6>.

The rules extracted in the second experiment explain what is in the data and how the FuNN solves this problem.

6. Evolving connectionist systems (ECOS) and evolving fuzzy neural networks (EFuNNs)

The ECOS and the EFuNNs paradigms were introduced in [9,10] and [11] respectively. An ECOS is a modular ’open’ structure evolving over time. Initially it is a mesh of nodes (neurons) with very little connections between them, pre-defined through *prior* knowledge or ’genetic’ information. These connections mainly connect modules of the initial connectionist structure. 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 previous ones (with a level of differentiation set through dynamically changing parameters.

The functioning of the ECOS [9] is based on the following general principles:

(1) Input patterns are presented one by one, in a pattern mode, having not necessarily the same input feature sets. After each input example is presented, 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. After the presentation of each new input example the system is able to react properly on both new and old examples.

(2) The representation module evolves in two phases. In phase one input vector **x** is passed through the representation module and the case nodes become activated based on the similarity between the input vector and their input connection weights. If there is no node activated above a certain *sensitivity threshold (Sthr)* a new rule neuron (*rn*) is created and its input weights are set equal to the values of the input vector **x** and the output weights - 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 with activation 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 which class (e.g., phoneme) the current vector **x** belongs to, is made in the higher-level decision module that may activate an adaptation process. Then the connections of the representation nodes to the class output nodes, and to the input nodes are updated with the use of *learning rate coefficients lr1 and lr2*, correspondingly. If the class activated is not the desired one, then a new case node is created. The feedback from the higher level decision module goes also to the feature selection and filtering part. New features may be involved 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.

(3) Along with growing, 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 error it contributes to, and the density of the surrounding area of neurons. Pruning is performed through the fuzzy rule:

IF case node (j) is OLD, and 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.

(4) 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 neuron is created when the input vector **x** was presented, it is allocated to the neuron, which had the highest activation to the input vector **x**.

(4) 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 previously 'seen' input patterns are used as ECO training examples. The connection weights that represent stored input patterns are now used as compressed 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:

(1) cascade eco training;
(2) sleep eco training.

In *cascade eco training* a new NN module is created when a new class data is presented. The module is trained on the positive examples of this class, plus the negative examples of the coming next different class data, and on the negative examples of previously stored patterns in previously created modules. In the sleep ECO training mode , modules are created with positive examples only when data is presented. Then the modules are trained on the stored in the other modules patterns as negative examples.

(6) ECOS provide explanation information extracted from the strcuture of the NNs. Each case (rule) node is interpreted as an IF-THEN rule as it is in the FuNN fuzzy neural networks.

(7) ECOS are biologically inspired. Some biological motivations for evolving systems are given in [10,11].

(8) The ECOS framework can be applied to different types of NNs, different types of neurons, activation functions, etc. One realisation that uses the ECOS framework is the evolving fuzzy neural networks EFuNNs and the EFuNN algorithm as given in [10,11].

EFuNNs are FuNN structures that evolve according to the ECOS principles. All nodes in an EFuNN are created during learning. The nodes representing membership functions (fuzzy label neurons) can also be modified during learning. As in FuNN, each input variable is represented here by a group of spatially arranged neurons that represent a fuzzy quantisation of this variable. For example, three neurons can be used to represent "small", "medium" and "large" fuzzy values of a variable. Different types of membership functions can be attached to these neurons (triangular, Gaussian, etc.). New neurons are created if for a given input vector the corresponding variable value does not belong to any of the existing membership functions to a membership degree greater than a membership threshold, e.g. 0.8. A new fuzzy label neuron, or an input variable neuron, can be created during the adaptation phase of an EFuNN. The EFuNN algorithm is introduced and illustrated in [11]. In the next section EFuNNs are used for on-line adaptation of a stock index CB-IDESS.

7. FuNNs and EFuNNs for time series prediction and decision making: a case study on stock index assessment

Here the NZ SE40 index is used for a case study. 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 only slightly different from a random fluctuation. A FuNN trained with the structural learning with forgetting algorithm is used in [14] for the prediction of the SE40. Ten time-lags have been initially set in the training data and ten rule nodes. After training, only four rule nodes are left in the FuNN structure suggesting that the rest of the rules are not important for the prediction task. The results are better than the obtained by using the random walk method.

Here, two experiments are presented with the use of a selected data set from the SE40 data (available from: http://divcom.otago.ac.nz:800/com/infosci/KEL/home.ht) - see fig.3a. The first one uses a FuNN and the second one - an EFuNN.

Experiment 1. FuNNs for the SE40 assessment

Three input variables are used to describe the SE40 time series: (1) the change in the current 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 (t+1). 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 (the last two months); epochs 1000, lr=0.1, mom=0.8. The obtained after training test error RMSE is 0.3. Ten rules are extracted from the trained FuNN:

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>.

The average training time for FuNN per example is 10^7 operations (summation, multiplication, etc). The next experiment shows a fast adaptive training of EFuNNs for the same case study.

Experiment 2. EFuNNs for adaptive, on-line SE40 assessement.

Here the same input and output variables are used as in experiment 1. The following evolving parameter values are used in the EFuNN: sensitivity threshold Sthr=0.92, error threshold Errthr=0.05, number of rule nodes evolved rn=910 (after pruning this number is 730); learning rate lr=0. The SE40 daily change is assessed in an on-line mode. The root mean square error is RMSE= 0.22 (on the 49 test data points as in the previous case), while the random walk gives RMSE=4.32. Five more EFuNNs were evolved to predict two, three, etc. 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 ahead prediction will give a better result than the random walk one-day ahead 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 before.

7. Conclusions

The paper suggests a general framework and its realisation for intelligent decision support that includes classification and prediction tasks. It illustrates how on-line decision support systems can be built based on fast, on-line, adaptive fuzzy neural networks evolving, and rule extraction.

References

1. Baestalus, Dik-Emma, van den Bergh, W.M.,Wood, D. Neural network solutions for trading financial market, Pitman Publications, 1994
2. Beltraffi, A., Margarita, S., Terna, P. Neural networks for economics and financial modelling, Int. Thomson Computer Press, 1996
3. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B., FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps, IEEE Trans. on Neural Networks,vol.3, No.5 (1991), 698-713
4. DeBoeck, L. Trading on the edge. Kluwer Academics, 1994
5. Heskes, T.M., Kappen, B. On-line learning processes in artificial neural networks, in: Math. foundations of neural networks, Elsevier, Amsterdam, (1993)199-233
6. Ishikawa, M., "Structural Learning with Forgetting", Neural Networks 9, 501-521 (1996).
7. Jang, R. "ANFIS: adaptive network-based fuzzy inference system", IEEE Trans. on Syst.,Man, Cybernetics, 23, 665-685 (1993).
8. 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
9. Kasabov, N. ECOS: A framework for evolving connectionist systems and the eco learning paradigm, Proc. of ICONIP'98, Kitakyushu, Oct. 1998
10. Kasabov, N. Evolving connectionist agents and systems. *IEEE Transactions on Man, Machine and Cybernetics*, submitted
11. Kasabov, N. Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation, in Proc. of Iizuka'98, Iizuka, Japan, Oct.1998
12. Kasabov, N. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, The MIT Press, CA, MA (1996).
13. Kasabov, N., Kim, JS, Watts, M. and Gray, A. FuNN/2 - A fuzzy neural network architecture for adaptive learning and knowledge acquisition. *Information Sciences* 101 (3-4): 155-175 (1997)
14. 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)
15. Le Cun, Y., J.S. Denker and S.A. Solla, "Optimal Brain Damage", in: Touretzky,ed., *Advances in Neural nform.Proc.Systems*,MorganKaufmann,2,598-605(90).
16. Lin, C.T. and C.S. G. Lee, "Neuro Fuzzy Systems", Prentice Hall (1996).