Implementing Knowledge and Data Fusion in a Versatile Software Environment for Adaptive Learning and Decision-Making

Nik Kasabov[1], David Tuck[2], Michael Watts[1]

[1] Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand
Ph (+64-3) 4798319, Fax (+64-3) 479-8311, nkasabov@otago.ac.nz, mike@kel.otago.ac.nz

[2]Imaging & Sensing Team, Industrial Research Limited, PO Box 2225, Auckland, New Zealand
Ph (+64-9) 3034116, Fax (+64-9) 302-8106, email: D.Tuck@irl.cri.nz

ABSTRACT: *Data fusion is used today in many engineering and managerial applications to help resolve complex planning, control and optimisation problems. The purpose of this paper is to introduce practical and versatile tools and an environment for implementing data fusion that also provides a reverse-engineering methodology to extract comprehensible rules developed from the data. The environment has two major tools (among several others) - a fuzzy neural network FuNN, and evolving fuzzy neural network EFuNN applicable for both off-line and on line adaptive learning and rule manipulation. The EFuNNs allow for on-line fusion of variables over time sequences of information through adaptive learning. Two case study time-series applications are presented and discussed: a water flow prediction and a provisional robot control example.*

Keywords: Decision-making, On-line Prediction, Fuzzy Neural Networks, Evolving Fuzzy Neural Networks, Rule Extraction.

1. Introduction

While artificial neural networks (ANN) *per se* can provide the ability to produce a model for the mechanisms underlying the information in sources data used in decision-making and time-series prediction processes, hybrid neuro-fuzzy systems that include both learning from data and fuzzy rules manipulation, add much more to this useful property [1, 8, 16, 17]. Many past data fusion applications have utilised *ad hoc* designs at some level in the decision-making process to include explicit information or *a priori* knowledge constraints, and a structure to assist in highly dynamic applications or poorly defined problem solutions. This capability has been made available in the fuzzy neural network structures and in the hybrid connectionist-based environments described here.

One particular example for combining neural networks and fuzzy systems is the concept of fuzzy neural networks (FNN) [17, 8]. By fuzzifying a neural network, the quantisation of the inputs and outputs, through the application of membership functions, extra robustness is provided when used with redundant, noisy or incomplete input data. Further, this fuzzification technique can provide the means for extracting the information learnt in the form of rules. It is also now possible to add explicit information or *a priori* knowledge constraints to the network and thereby improve the interpretation of the rules learnt by the network, after training.

Here, two types of FNNs are illustrated as part of a hybrid software environment: the fuzzy neural network FuNN [8-11], used for off-line learning rule manipulation, and the evolving fuzzy neural network EFuNN [12-14] used for on-line real time learning and prediction.

Since the paradigm of hybrid connectionist-rule based systems was established [6] there are now several software environments that implement this paradigm. The first generation of such environments (see for example COPE [7]) implemented in a logical way, different types of ANN (such as multi-layer perceptrons, Kohonen self-organising maps [15], adaptive-resonance theory ANN [1]), to be combined with the CLIPS-based production systems. Here, an ANN could be called for training, or for recall of the

action part of the production rules [6, 7]. The second generation of such environments included fuzzy rules and fuzzy neural networks. Such an environment was FuzzyCOPE [8]. This new data fusion environment further developed the main principles of COPE [7] through a combination of the Fuzzy-CLIPS (an extension of CLIPS) developed by the NRC in Canada in 1994, http://ai.iit.nrc.ca/fuzzy/fuzzy.html and fuzzy inference and fuzzy-neural network modules at http://divcom.otago.ac.nz/com/infosci/KEL/home.htm.

The latest development in the series of FuzzyCOPE environments, FuzzyCOPE/3, allows for the extraction of a more comprehensible interpretation of the underlying rules implicit in the data used in training. It also has a module (EFuNN) for on-line learning where the inputs (sources of information) are not pre-defined and can vary during the on-line learning process, thus allowing for "on the fly" fusion of different sources of information and fuzzy rules.

2. The Fuzzy Neural Network

Fuzzy neural networks (FNNs) are connectionist models for fuzzy rules implementation and inference [8-11, 17]. However, there are a wide variety of architectures and functionality, differing in the type of fuzzy rules, type of inference method, and modes of operation. In general the architecture of these FNNs consist of five layers, Fig. 1. These layers in order are:

A. An input layer, where the neurones represent the linguistic variables of the input data;
B. A fuzzification, or condition layer, where the neurones represent the fuzzy values;
C. A rules layer, where the neurones represent the fuzzy rules;
D. An action layer, where the neurones represent the fuzzy values of the output variables, and finally;
E. An output layer, where the neurones represent the output linguistic variables.

The example illustrated in Fig.1 has two inputs, with two fuzzy membership functions (MF) each, two rule nodes, and two outputs, again with two MF each.

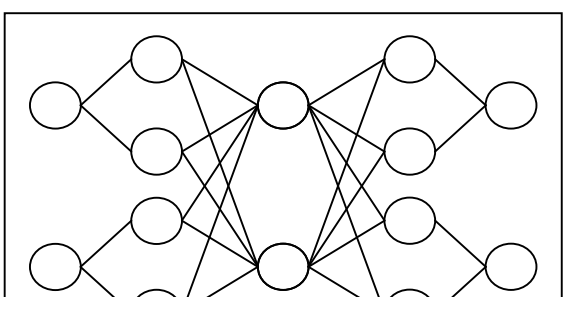

Figure 1: A general structure of a fuzzy neural network

FuNN is a FNN developed and presented in [8-11]. It is characterised by the following features: using weighted fuzzy rules [8]; modified back-propagation algorithms for training that include training with forgetting; using genetic algorithms, to improve and speed up training [2]; training with or without modifying the membership functions [11]; different types of rule extraction (e.g. simple fuzzy rules, weighted fuzzy rules, aggregated rules [10,11]); and rule insertion.

FuNNs have four basic advantages over ANNs (and standard fuzzy systems):

1. The FuNN structure is interpretable by fuzzy linguistic "if-then" rules – not so readily achieved for ANNs;
2. A FuNN is more likely to converge to a global minimum in error-weight space under arbitrary conditions, than an ANN;
3. FuNNs show a remarkable improvement in learning speed and accuracy compared to an equivalent ANN;
4. A FuNN can learn to predict signal variation well, even if it is of a chaotic signal type.

Usually, the FuNNs employ standard triangular membership functions and the number of rule nodes and rules are defined and fixed by the analyst prior to initialisation. However, FuNNs do have some difficulties when applied to on-line modelling and prediction [5], but these can be overcome by the evolving FuNNs as described below.

3. Evolving Fuzzy Neural Networks

Evolving fuzzy neural networks (EfuNNs) were introduced in [12-14]. In this extension of the

FuNN architecture, the network begins with an empty rule layer. As training patterns are presented to the network, examples that are not adequately represented by the rule layer, trigger the addition of nodes to represent these new examples. Each rule node, after training, therefore represents one or several training examples.

EFuNNs have the following characteristics:

- Memory-based learning where exemplars of data are stored as they arrive at the inputs;
- Open structure – the number of the inputs and the outputs of the EFuNN can vary from example to example thus making fusion from an unknown number of sources possible in an on-line, "on the fly" mode, and;
- Local tuning of connection weights [12-14].

EFuNNs also exhibit the following advantages over conventional FuNNs:

- Rapid, one pass training;
- Good generalisation capability, both local and global;
- Robustness to forgetting, and;
- Rapid adaptation to new data.

4. The Hybrid Environment FuzzyCOPE/3

FuzzyCOPE/3 is a suite of data processing and neural network tools for the Microsoft Windows environment. FuzzyCOPE was developed by the Knowledge Engineering Laboratory of the Department of Information Science at the University of Otago. It consists of a graphical user interface built on top of a computational engine. The engine, which is encapsulated within a dynamic link library (DLL), is actually a simple command interpreter capable of creating and manipulating multiple instances of various classes of objects. These include data sets, multi-layer perceptrons, self-organising maps, and different types of fuzzy neural networks. Communication between the interface and engine is via customised formatted commands and result strings. These strings are assembled and parsed by specially written Application Programming Interface (API) libraries. This approach was adopted for maximum flexibility: it eliminates problems with handling C++ style pointers, it avoids problems with passing data in proprietary formats, it simplifies use of the engine (only the API library functions need be considered at the application level) and it lends itself readily to future expansion, such as a possible client-server architecture, or even the implementation of a specialised programming language. The FuzzyCOPE/3 environment is currently being used by more than 35 universities from all over the world as a teaching environment for courses in computational intelligence. There are also more than 200 developers of intelligent information systems using it. The environment is available from the web site at
http://kel.otago.ac.nz/software/FuzzyCOPE3/

5. Case Study I - Water Flow Prediction

5.1 The Problem

This first example problem chosen for this paper was that of water flow prediction to a sewage plant (see also [8]). Given the time of day t, (0 - 23), whether or not it is a holiday (0 or 1), and the water flow over the past few hours (t-1, t-2 etc.), the task is to predict the water flow for the next hour. This is a time-series prediction problem useful for resource management. Accurate prediction of the water flow is necessary to allow for finer control of the sewage plant process.

The data is highly variable, with large differences between the hourly water flow for a workday as compared to a holiday. An extract of the data, shown in Fig.2, demonstrates the typical difference between holiday (dotted line) and workday (solid line) flows.

5.2 Experimental Data Sets

Two data sets, a training set and a testing set, were prepared. Each data set contained four input variables and one output variable. The input variables represented the time of day, whether or not it was a holiday, and the water flow over each of the two preceding time intervals. There were 503 examples in the training set, 176 examples in the test set. Due to the requirements of the FuNN and EFuNN architectures, each data set was

linearly normalised so that the values all reside within the range [0, 1].

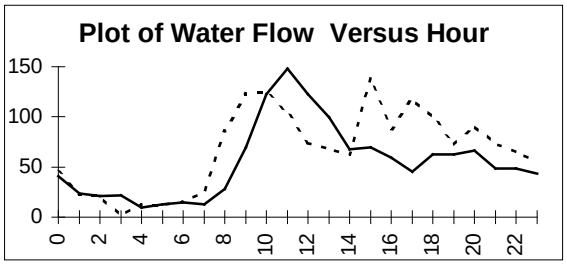

Figure 2: Extract of water flow data for holiday and workday (see text).

5.3 Off-line Training, Prediction and Rule Extraction with FuNNs

A fuzzy neural network FuNN was created within the FuzzyCOPE/3 environment. It consisted of four inputs, one for each input variable described above in 5.2. The first input had four MF attached (representing early morning, morning, afternoon and evening). The second and third inputs each had three MF attached (representing low, medium and high water flow). The final input had two MF attached (either a holiday, or not). Ten rule nodes were used, and the single output had three MF (again representing low, medium and high water flow).

This network was trained for 10,000 epochs using the backpropagation algorithm, and the results were recalled with the test data. The results of the recall are presented in Fig.3, where the actual (solid line) and predicted (dotted line) water flow are plotted.

After recall, a set of fuzzy rules was extracted. These rules seem to explain well the relationship between the input variables and the expected water flow. A set of example rules is presented below.

If <Time is EarlyMorning 4.63992> **and** <Flow_T-2 is Low 1.63653> **and** <Holiday is Is 1.77835>, **then** <Flow is Medium 3.81119>

If <Time is Morning 13.5842> **and** <Flow_T-1 is High 13.8779> **and** <Flow_T-2 is Low 5.44741>, **then** <Flow is Medium 1.86714>

If <Time is Afternoon 19.1327> and <Flow_T-1 is Low 24.77> **and** <Flow_T-2 is Medium 7.791> **and** <Holiday is IsNot 5.04419>, **then** <Flow is Medium 1.58037>

If <Time is Evening 6.96259> **and** <Flow_T-1 is High 7.72363> **and** <Flow_T-2 is Medium 3.65387>, **then** <Flow is Medium 0.955361>

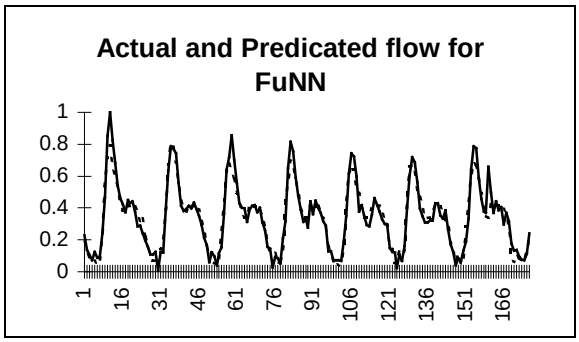

Figure 3: Plot of actual and predicted water flow for the trained FuNN.

5.4 On-line Prediction with EFuNNs

An evolving fuzzy neural network, EFuNN was first created with the same number of inputs and outputs (and input and output MFs). Because EFuNNs add rule nodes as required, the rule layer initially consisted of one node.

This network was then trained in an on-line mode, so that after the first data input vector had been presented, the network was next tested to predict the new hourly flow value. Finally, when the actual flow value became known, the input – output association was added to the EFuNN through a one-epoch adaptive training. Then the cycle repeats and the EFuNN was used to predict the next new value, etc. After the presentation of the first 75 examples two new inputs were added to the EFuNN without re-training the whole system, these were the moving average 12 hours and the moving average 24 hours of the flow data. The EFuNN continued to grow. When the number of nodes reached 70 the EFuNN then

started pruning the nodes as explained in [12-14]. A fuzzy rule for pruning was used based on the total activation of the rule nodes and the "age" (the time from creation). Fig. 4 presents the actual water flow (solid line) and the predicted (dotted line) on-line mode water flow.

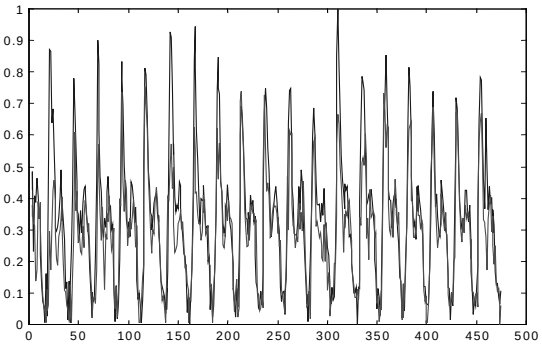

Figure 4: The actual and the predicted EFuNN on-line mode water flow.

It is clear that at the beginning the EFuNN could not predict well, not having any training or *a priori* knowledge. The more it was trained on the incoming data the better the prediction became.

EFuNN simulators are written in MATLAB and C++ and are part of the NZ-RICBIS – the New Zealand Repository for Intelligent Connectionist-based Information Systems. This is available at http:divcom.otago.ac.nz/infosci/kel/CBIIS.html

The water flow data is available from http://kel.otago.ac.nz/software.

5.5 Comparative Analysis of the Different Fusion Techniques for the Water Flow Prediction Problem

Both the FuNN and EFuNN were able to approximate the data to a reasonable degree of accuracy. However, while the FuNN required 10,000 training epochs (taking approximately 20 minutes on a 233-Mhz Pentium II), the EFuNN required only one pass through the training data, taking less than 20 seconds. It is this rapid training capability that is one of the major advantages of EFuNNs. Rules from an EFuNN can also be extracted and inserted [12-14].

6. Case Study II - On-line Robot Control

6.1 The Problem

In a New Zealand meat-works, a sheep is valued for both its pelt and meat products. Lamb meat is an important export product and the fluffy sheepskins make great souvenirs for our tourist visitors.

In order to remove the carcass pelt without damage to itself or the flesh underneath, extreme care is required in the initial cutting operation of the skin. For the purpose of this example, a new robot cutting path planner approach has been investigated. At present an algorithmic path planning robotics system has been developed and is being trialed in a New Zealand meat-works, so far showing great potential over the traditional manual butchering preparation. However, this current approach is somewhat limited by the rather restricted algorithmic method of the semi-automated implementation developed.

We have started to explore use of the FuNN tool from FuzzyCOPE/3 to first develop a model of this current algorithmic planner. Then later, if the model demonstrates success, we propose to pursue and utilise the on-line adaptation properties of the EFuNN to continue learning to compensate for the highly variable sizes and shapes of this natural product (sheep). The present the algorithmic method allows for some on-line modification to the cutting path planning, when sheep variations demand it, but only by manual operator intervention through the adjustment of certain parameters which effect the two cut intersection point in the Y-Z plane.

6.2 Experimental Data Sets

The carcass de-pelting process starts with what is termed as a “Y-cut”, performed on the sheep carcass while hanging upside down on a moving conveyor chain, Fig. 5. This skin Y-cut, really two separate cuts, begins with one front leg at the hoof and follows down that leg and across the lower neck/chest region of the carcass (also known as the brisket), terminating just past the midline of the body. A second cut is then carried

out following a similar path, but mirroring the first cut, beginning at the hoof of the second front leg, continuing down it to finish just past the point of intersection with the first cut. When the completed Y- cut is performed correctly, the pelt can be pulled off the carcass as a whole piece and with minimal damage.

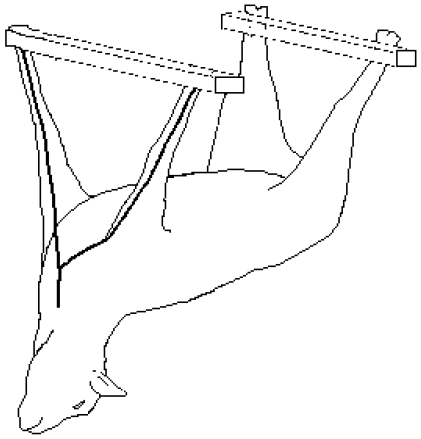

Figure 5: An example of the carcass Y-cut path.

For this second time-series study, the sheep carcass Y-cut sensor data was used, together with the algorithm path data, for training with a fixed parameter setting. Three sensors provide three-dimensional measurements of important points on the carcass so that the robotic skin cutting operation can be planned. These measurements are: the separation between the two front hooves; the highest point on the brisket; and finally the horizontal offset between the brisket and the trachea region of the neck. At present, the *ad hoc* algorithmic intersection point for the two cuts is determined by manual parameter settings. In a future development of an EFuNN multi-sensor data model to determine corrections to the algorithm calculations, we aim to fully automate this the path prediction despite the sheep variations by using the on-line learning and adaptation mode of the EFuNN.

Because the carcass is hung from an overhead conveyor line from its hooves the starting points are easily identified and provide the [0, 0, 0] reference in space for the cut. However, the meeting point of the cuts and their paths down the front legs of the carcass in space are very much dependent on the size and breed of the animal. Also, because the carcass is continually moving along the conveyor line, the cut intersection point needs to be accurately determined and tracked, although cutting is assisted by design of the hook shaped knife. The shape pulls the skin away from the flesh and helps ensure the knife just cuts through it.

6.3 Preliminary Results - Training Path Planning with FuNN

An off-line fuzzy neural network cutting path planning model is being developed using FuNN to predict the next knife position for time, t. The input consists of 12 nodes, each having five membership functions (MFs) for fuzzification. The first three inputs are the X, Y, and Z carcass sensor measurements made on each sheep as described above. The next three inputs are the x, y, and z coordinates of the last (t-1) knife position. The final two sets of three inputs are the time lagged (t-2) and (t-3) coordinate positions. Three output nodes [x_o, y_o, z_o] with 7 MFs each generate the 3-D predicted cutting path sequence.

Data for the Y-cuts, taken from 83 sheep were used for this preliminary investigation - 50 for training and 33 for testing. The curent algorithm generated the time-series sequence of 100 cutting path positions, each sheep, which control the robot arm manipulation of the cutting knife. A limited range of animals sizes and shapes were included. Each carcass cutting path data set of 100 vectors contained the 12 input data values (X, Y, Z measurements followed by the three time lags of the previous knife positions), and then the next [x_o, y_o, z_o] predicted position for the knife, to be learnt.

The best results obtained so far have been with a 15 node rule layer and after only 100 training epochs. Further experimentation is obviously required to refine the model. Figures 6 and 7 display typical results of a single cut for one sheep, x versus y and z versus y respectively, with the actual (solid line) and the FuNN predicted (dotted line) cut paths superimposed. The average RMS differences are 4.6, 11.8 and

8.5 (mm) for the x, y, and z directions respectively for 50 carcasses.

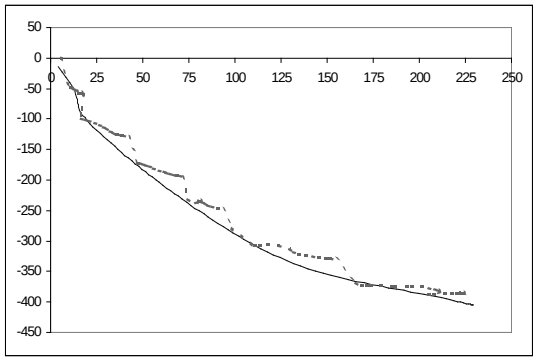

Figure 6: Typical x-y cutting path (mm) from FuNN.

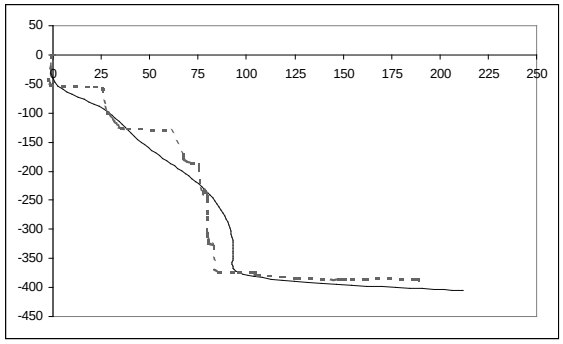

Figure 7: Resultant z-y cutting path (mm) for Fig. 6.

7. Discussion and Conclusions

Connectionist-based algorithms are robust when the appropriate techniques are used. They allow the analyst to learn relationships between the input and output variables without making assumptions about the data distribution. Thus, improving the prediction or classification accuracy is based on updating the transfer function and not manipulating the incoming data flow. Also the fuzzified connectionist-based algorithms may now require fewer training examples than traditional sensor data fusion methods. The results of ANNs and FuNNs, over fuzzy rules and more traditional statistical methods can be shown to have a distinct advantage [8]. For example, the adaptive learning algorithms enable the EFuNNs to learn relationships between input data and output data in an iterative way [12-14] and on-line. Finally, fuzzy rules may then be extracted and updated from all the classes of FuNN to help explain what the network has learned.

When using FuNNs and EFuNNs one should always refer to traditional statistical methods and compare the results with them, if possible. However, there exist disadvantages in applying statistical algorithms to determine the input-output transfer function characteristics. First, this approach requires large amounts of sample data for processing. Second, it is not capable of handling conflicting information that can arise in the transfer function it is trying to model and this cannot be updated without changing the input data - there is no feedback process for statistical algorithms to learn from *a posteriori* knowledge. For example, they do not cope well where the data distribution is bimodal or very non-normal, which are the case here. Also, the sensitivity for the separation between output states is a function of all the inputs, so closely positioned states are not well distinguished. However, statistical methods can suit some models where the data is uni-modal and normal. Then this approach has the advantages of being computationally efficient and capable of producing highly accurate results.

In the first study, two of the hybrid neuro-fuzzy modules of FuzzyCOPE/3, FuNNs and EFuNNs have been demonstrated and in the second case study a preliminary FuNN application looks promising. Work on this robotic path planning problem is to continue and it is expected that a fully automated solution can be developed. While the modules performed acceptably in both cases, it is expected that recurrent versions of these networks, scheduled to be included in the next version of FuzzyCOPE, will yield even better results.

The objective of this paper has been to promote awareness of this new and versatile data fusion, FuzzyCOPE/3 environment, and to entice others to investigate and apply it to new real world problems. The results presented here, hopefully demonstrate the potential of this fusion environment for providing solutions to previously difficult to solve problems.

Acknowledgments

This research is partially supported by the Foundation for Research, Science and Technology, New Zealand through a grant UOO808 to the University of Otago, and NSOF at Industrial Research Limited. Also thanks must go to Dr Malcolm Taylor for the acquisition and generation of the robotic cutting path data used in the second case study.

References

1. Carpenter, G. S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, "FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps," IEEE Transactions of Neural Networks , vol.3, No.5, 698-713 (1991).
2. Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley (1989).
3. Hassibi and Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," in: Advances in Neural Information Processing Systems, 4, 164-171, (1992).
4. Hech-Nielsen, R. "Counter-propagation networks", IEEE First int. conference on neural networks, San Diego, vol.2, pp.19-31 (1987).
5. Heskes, T.M., B. Kappen, "On-line learning processes in artificial neural networks", in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233, (1993).
6. Kasabov N. Hybrid connectionist rule based systems. In Artificial Intelligence IV Methodology, Systems, Applications. P.Jorrand and V.Sgurev eds. Amsterdam, North-Holland (1990) 227- 235.
7. Kasabov, N. COPE-a hybrid connectionist production system environment. In *Proceedings of the Third Australian Conference on Neural Networks (ACNN'92).* Sydney, Sydney University Electrical Engineering (1992) 135-138.
8. Kasabov, N., Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA, (1996).
9. Kasabov, N. "Adaptable connectionist production systems". Neurocomputing, 13 (2-4) 95-117, (1996).
10. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2-20 (1996).
11. Kasabov, N., Kim, J., Watts, M. and Gray, A. Architecture for adaptive learning and knowledge acquistion, Information sciences, 101 (3-4_: 155-175 (1997).
12. Kasabov, N., "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in: in: Yamakawa and Matsumoto (eds), Methodologies for the Conception, design and Application of Soft Computing, World Scientific, 1998, 271-274.
13. Kasabov, N., "ECOS: A framework for evolving connectionist systems and the ECO learning paradigm", Proc. of ICONIP'98, Kitakyushu, Japan, Oct. 1998, IOS Press, 1222-1235.
14. Kasabov, N. The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems, Journal of Advanced Computational Intelligence, 2 (6) 1998, 1-8.
15. Kohonen, T., Self-Organizing Maps, second edition, Springer Verlag, 1997.
16. Tuck, D., "Preliminary CBIS Findings: Forecasting a Variable Multiple Cyclic Process with Confidence!" Proc. ICONIP'97, ANZIIS'97 &ANNES'97, Dunedin, NZ, November, 1997, Springer, Vol.2, 992-995.
17. Yamakawa, T., H. Kusanagi, E. Uchino and T. Miki, "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017-1020, (1993).