Control and Cybernetics

vol. **27** (1998) No. 4

A fuzzy neural network for knowledge acquisition in complex time series

by

N. Kasabov, J. Kim and R. Kozma

Department of Information Science,
University of Otago,
P.O. Box 56, Dunedin,
New Zealand

Abstract: A novel fuzzy neural network, called FuNN, is applied here for time-series modelling. FuNN models have several features that make them well suited to a wide range of knowledge engineering applications. These strengths include fast and accurate learning, good generalisation capabilities, excellent explanation facilities in the form of semantically meaningful fuzzy rules, and the ability to accommodate both numerical data and existing expert knowledge about the problem under consideration. We investigate the effectiveness of the proposed *neuro-fuzzy* hybrid architectures for manipulating the future behaviour of nonlinear dynamical systems and interpreting fuzzy if-then rules. A well-known example of Box and Jenkins is used as a benchmark time series in the proposed modelling approach and the other modelling approach. Finally, experimental results and comparisons with the other popular neuro-fuzzy inference system, namely Adaptive Network-based Fuzzy Inference System (ANFIS) are also presented.

Keywords: fuzzy neural net, time-series and dynamical system, knowledge acquisition, computational neural net, fuzzy logic, and adaptation

1. Introduction

Over the last decade, significant advances have been made in two distinct technological areas: fuzzy logic and computational neural networks. The theory of fuzzy logic (Zadeh, 1965) provides a mathematical framework to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. Also, it provides a mathematical morphology to emulate certain perceptual and linguistic attributes associated with human cognition. On the other hand, the computational neural network paradigms have evolved in the process of understanding the learning and adaptive features of neuronal mechanisms inherent in certain biological species. The integration of two fields has given birth to an emerging technological field—the fuzzy neural networks. The fuzzy neural networks have the potential to capture the benefits of the two fascinating fields, fuzzy logic and neural networks, into a single entity (Lin & Lee, 1991; Yamakawa *et. al.*, 1992; Jang, 1993; Ishibuchi *et. al.*, 1994; Hauptmann & Heesche, 1995; Kasabov, 1996a; Kasabov, 1996b; Kasabov *et. al.*, 1997). It is also shown that neuro-fuzzy hybrid architecture is capable of modelling chaotic time series data (Katayama *et. al.*, 1995).

The intent of this paper is to demonstrate how a novel fuzzy neural network, called FuNN (FUzzy Neural Network) (Kasabov, 1996a; Kasabov, 1996b; Kasabov *et. al.*, 1998), can be used for time-series data modelling and for a wide range of knowledge engineering applications. Experiments with a highly irregular time series data are used to illustrate the effectiveness of the suggested type of fuzzy neural network for modelling, prediction, knowledge acquisition and adaptation.

2. Time series data modelling and the case study problem

Time-series data modelling is a generic problem which permeates all fields of science. The increased interest in nonlinear systems is also related to the discovery of chaos, as chaos can readily occur in all natural and living systems where *nonlinearity* is present. Chaos is currently one of the most exciting topics in nonlinear systems research.

The gas-furnace data set of Box-Jenkins is well known and is often used as a standard test for the system identification (Box & Jenkins, 1970). The time series used for identification purposes consists of 296 successive pairs of input-output observations measured at a sampling rate of 9 sec from a gas furnace system where the input is the gas flow rate into the furnace and a single output is the concentration of CO_2 in the exhaust gas.

The task of this process identification is to provide a prognosis for the carbon-dioxide concentration at the moment (t) given the methane portion at a time moment (t-4) and the carbon-dioxide concentration at a time moment (t-1) as input variables. In our experiments, the data set consists of only 292 consecutive values of methane (t-4) and the produced in a furnace CO_2 (t-1) as input variables and the produced CO_2 at the moment (t) as an output variable. Fig. 1 shows the CO_2 time series data, together with its various characteristic functions. The power spectral density (PSD) has large statistical fluctuations. It is rather constant at low frequencies and appropriates a constant slope at high frequencies. The histogram is more flat than a Gaussian distribution, the auto correlation function (ACF) indicates a decay constant of about 5 time lags. The 3-dimensional delay-coordinate plot indicates certain periodicities superimposed upon an approximately higher regression behaviour.

In the next section we discuss the qualitative modelling of a dynamic process

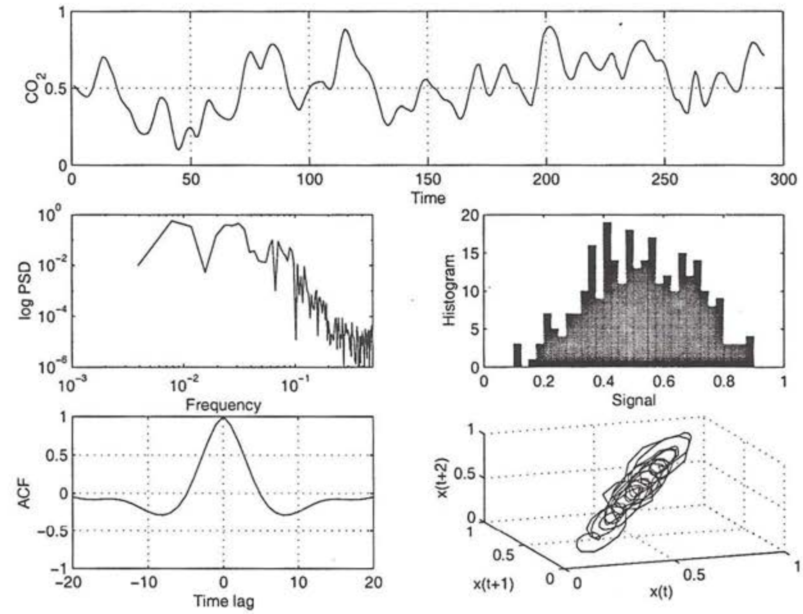

Figure 1. Characterisation of gas-furnace time series: (a) Original time series; (b) Power spectrum density; (c) Histogram; (d) Auto correlation function; (e) 3-dimensional phase diagram.

through a novel architecture for *fuzzy neural network* (FNN), called **FuNN**, which stands for **Fu**zzy **N**eural **N**etwork, a investigate some learning and adaptation strategies using the example of the gas furnace system identification.

3. FuNN—A fuzzy neural network for modelling and knowledge discovery

The fuzzy Neural network FuNN (Kasabov, 1996a; Kasabov, 1996b; Kasabov *et. al.*, 1998) uses a multi-layered perceptron (MLP) network and a extended BP training algorithm. In this connectionist structure, the input and output nodes represent the input states and output control/decision signals respectively, and in the hidden layers, there are nodes functioning as membership functions (MFs) and rules. This eliminates the disadvantage of a normal feedforward multi-layer net which is difficult for an outside observer to understand or to modify.

The architecture facilitates learning from data and approximate reasoning, as well as fuzzy rule extraction and insertion. It allows for the combination of both numerical and fuzzy data and fuzzy rules to be used in one system, thus producing the synergistic benefits associated with the two sources. In addition, it allows for adaptive learning in a dynamically changing environment.

Below a brief description of the components of the FuNN structure and functionalities, and the philosophy behind this architecture, are given.

3.1. The architecture of FuNN

The general FuNN architecture consists of 5 layers with partial feedforward connections as shown in Fig. 2. In this connectionist structure a modified BP training algorithm was developed. The first and last layer act as the fuzzifier and the defuzzifier, respectively. In the condition layer, uniformly distributed triangular membership functions are used. Singletons are applied in between the action and the output layer, as connection weights, which represent the centre of a membership functions. FuNN is also adaptable where the membership functions of the fuzzy predicates, as well as the fuzzy rules inserted before training or adaptation, may adapt and change according to new training data.

- **Input Layer (Layer One):** Nodes in layer one are input nodes which represent input linguistic variables (Zadeh, 1973). The nodes in this layer only transmit input values to the next layer, *condition element layer*.
- **Condition Layer (Layer Two):** Nodes in this layer acts as fuzzification processors. The input values are fed to the condition element layer which performs fuzzification. This is implemented using three-point triangular membership functions with centres represented as weights. The triangles are completed with the minimum and maximum points attached to adjacent centres, or shouldered in the case of the first and last membership functions.

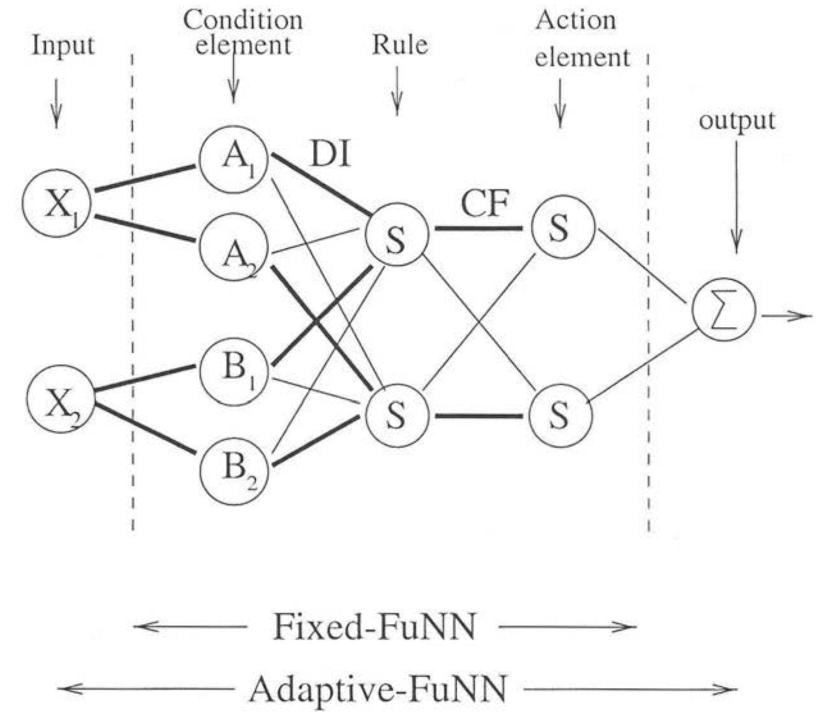

Figure 2. A FuNN structure for two fuzzy rules

The triangular membership functions are allowed to be non-symmetrical and any input value will belong to maximum of two membership functions with degrees differing from zero. It will always involve two unless the input value falls exactly on a membership function *centre* in which case the single membership will be activated. These membership degrees for any given input will always sum up to one, ensuring that some rules will be given the opportunity to fire for all points in the input space. This centre-based membership approach taken by FuNN avoids the problems of uncovered regions in the input space. These do not always limit centres and widths in such a way as to ensure complete coverage. While algorithms could be formulated and used in such cases to force the memberships to cover the input space, the simple centre-based approach taken by FuNN seems both more efficient and more natural, with fewer arbitrary restrictions. It should be noted that there are no *bias* connections necessary for this representation in FuNN.
Initially the membership functions are spaced equally over the weight space, although if any expert knowledge is available this can be used for initialisation. In order to maintain the semantic meaning of the membership functions some restrictions are introduced. When adaptation takes place the centres are limited to remain within equally sized partitions of the weight space. This avoids problems with violating the semantic ordering of membership functions. Therefore, under the FuNN architecture labels can be attached to weights when the network is constructed and these will remain valid for the *lifetime* of the network. For example, a membership function weight representing *low* always have a centre less than *medium*, which will always be less than *high*.
Simple activation functions are used in the condition element nodes to perform fuzzification.

- **Rule Layer (Layer Three):** Each node in this layer is a rule node which represents a single fuzzy rule. Thus, all the nodes in this layer form a fuzzy rule base.
  The activation function is the sigmoid (logistic) function with a variable gain coefficient (a default value of 1 is used). The connection weights from the Condition Layer are initialised randomly with small values and fully connected.
  The semantic meaning of the activation of a node is that it represents the degree to which input data matches the antecedent component of the associated fuzzy rule. However the synergistic nature of rules in a fuzzy-neural architecture must be remembered when interpreting such rules. The connection weights from the *Condition element Layer* (also called the *membership functions layer*) to the *Rule Layer* represent semantically the **degrees of importance** (DI) of the corresponding condition elements for the activation of this node.

- **Action Layer (Layer Four):** In this layer links define the consequences of the rules and a node represents a fuzzy label from the fuzzy quantisation space of an output variable. The activation of the node represents the degree to which this membership function is supported by all fuzzy rules together. So this is the level to which the membership function for this fuzzy linguistic label is *cut* according to the rules and current facts. The connections from the *Rule Layer* to the *Action Element Layer* represent conceptually the **confidence factors** (CF) or certainties of the corresponding rules when inferring fuzzy output values. They are subject to constraints that require them to remain in specified intervals as for the condition element layer with the same advantages of semantic interpretability. The activation function for the nodes of this layer is the sigmoid (logistic) function with the same or variable gain factor, and connection weights are initialised as in the previous layer. This gain factor should be adjusted appropriately given the size of the weight boundary.
- **Output Layer (Layer Five):** It represents the output variables of the system. This node and links attached to them act as the defuzzifier. This layer performs the centre of gravity (COG) defuzzification.
  Singletons are used as membership functions for the output labels, which is equivalent to having the centres only of triangular membership functions, as it was the case of the input variables, and are attached as connection weights to the corresponding links. Linear activation functions are used here.
  Adapting the output membership functions means moving the centres. The requirement that the membership degrees to which a particular output value belongs to the various fuzzy labels must always sum to one, is always satisfied. For each centre, there is a constraining band (partition) where this value can move to. This principle applies in the same way as the input membership function centres restrictions are.

Details of the supervised learning algorithms of FuNN are given below.

3.2. Basic learning algorithm

This section explains the algorithm used for the FuNN system, both the forward phase and the backward phase of errors.

3.2.1. Forward pass

This phase computes the activation values of all the nodes in the network from the first to fifth layers. In this a superscript indicates the layer and a subscript describes connection weights between layers.

- **Input Layer:** The nodes in this layer only transmit input values (crisp values) to the next layer directly without modification.

- **Condition Layer:** The output function of this node is the degree that the input belongs to the given membership function. The input weight represents the centre for that particular membership function, with the minimum and maximum determined using the adjacent membership's centres.
  In the case of the first and last membership function for a particular variable a shoulder is used instead. Hence, this layer acts as the fuzzifier. Each membership function is triangular and an input signal(x) activates only two neighbouring membership functions simultaneously, the sum of the grades of these two adjacent membership functions for any given input is always equal to 1.
  For a triangle-shaped membership function as in FuNN, the activation functions for a node (i) are:
$$\begin{aligned} Act_i^c &= 1 - \frac{x - a_i}{a_{i+1} - a_i}, \quad a_i < x < a_{i+1}, \\ Act_i^c &= 1 - \frac{a_i - x}{a_i - a_{i-1}}, \quad a_{i-1} < x < a_i, \\ Act_i^c &= 1, \quad x = a_i, \end{aligned} \tag{1}$$
  where a is the centre of the triangular membership function.
- **Rule Layer:** The connections from the condition to this layer are used to perform pre-condition matching of fuzzy rules. The connection weights may be set either randomly and then trained or according to a set of rules, namely rules insertion. The net inputs and activations are respectively,
$$\begin{aligned} Net^r &= \sum_c w_{rc} Act^c, \\ Act^r &= \frac{1}{1 + \epsilon^{-gNet^r}}, \end{aligned} \tag{2}$$
  where g is a gain factor.
- **Action Layer:** The nodes and connection weights in this layer function as those in the **Rule Layer** for Net input and activation:
$$\begin{aligned} Net^a &= \sum_r w_{ar} Act^r, \\ Act^a &= \frac{1}{1 + \epsilon^{-gNet^a}}. \end{aligned} \tag{3}$$
- **Output Layer:** This layer performs defuzzification to produce a crisp output value. Among the commonly used defuzzification strategies, the *Centre of Gravity* (COG) method was used:
$$\begin{aligned} Net^o &= \sum_a w_{oa} Act^a, \\ Act^o &= \frac{Net^o}{\sum Act^a}. \end{aligned} \tag{4}$$

3.2.2. Backward pass

The goal of the system is to minimise the following function:

$$E = \frac{1}{2}\sum(y^d - y^o)^2 \tag{5}$$

where y^d is the desired output and y^o is the current output. Hence the general learning rule (gradient descent) used is

$$\begin{aligned} \Delta w &\approx -\frac{\partial E}{\partial w}, \\ \Delta w_{t+1} &= \eta(-\frac{\partial E}{\partial w}) + \alpha \Delta w_t, \end{aligned} \tag{6}$$

where η is the learning rate and α is the momentum coefficient, and using *chain rule* we have

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial Net}\frac{\partial Net}{\partial w} = -\delta Act. \tag{7}$$

Hence the weight update rule is:

$$\Delta w_{t+1} = \eta \delta Act + \alpha \Delta w_t. \tag{8}$$

- **Output Layer:** The error signal δ^o is derived as in the following:

$$\begin{aligned} \delta^o &= -\frac{\partial E}{\partial Net^o} \\ &= -\frac{\partial E}{\partial Act^o}\frac{\partial Act^o}{\partial Net^o} \\ &= y^d - y^o \end{aligned} \tag{9}$$

- **Action Layer:** The error for nodes in this layer is calculated based on fuzzification of desired outputs and activation of each node. The fuzzification of desired output for this layer is same as Eq. (1). Hence we have

$$\begin{aligned} \delta^a &= f^{'}(Net^a) \times E^a \\ &= Act^a(1 - Act^a)\sum(d^a - Act^a) \end{aligned} \tag{10}$$

- **Rule Layer:** As in the Action layer, the error signals need to be computed and this error signal can be derived as

$$\delta^r = Act^r(1 - Act^r)\sum(w_{ar}\delta^a) \tag{11}$$

- **Condition Layer:** If inputs lies in the fuzzy segment, then the corresponding weight should be increased directly proportional to the propagated error from the previous layer, because the error is caused by the weight. This proposition can be represented by the following equation:

$$\delta^c = \frac{\partial Act_i^c}{\partial a_i}\sum(w_{rc}\delta^r). \tag{12}$$

Using Eq. (1), the adaptive rule of the centre a_i, is derived as

$$\frac{\partial Act_i^c}{\partial a_i} = \begin{cases} \frac{\partial a_i - x}{\partial a_i - a_{i+1}{}^2}, & \text{if} \quad a_i \leq \mathrm{x} \leq a_{i+1}, \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Hence the adaptive rule of connection weights becomes

$$\Delta w_{t+1} = \eta \delta^c x + \alpha \Delta w_t. \tag{14}$$

3.3. Training and adaptation in FuNN

Several methods have been developed for training and adaptation in a FuNN structure, namely,

1. A *partially adaptive training*, where the membership functions (MF) of the input and the output variables do not change during training (fixed or frozen mode) and a modified backpropagation algorithm is used for the purpose of rule adaptation only. This adaptation mode can be suitable for systems where the membership functions to be used are known in advance or where the implementation is constructed by the problem in some way.
2. A *fully adaptive training* with an extended backpropagation algorithm. This version allows changes to be made to both rules and membership functions, subject to constraints necessary for retaining semantic meaning.
3. A *partially adaptive version* as in (1) but a special type of *network pruning* is applied, which is a modified backpropagation learning algorithm with forgetting introduced to the connection weights. This method belongs to the class of structural learning algorithms. By applying learning with forgetting, the weights decrease continuously, unless they are reinforced by the backpropagation rule. At the end of the training, only the essential weights deviate significantly from zero. By pruning the weights which are close to zero, a skeleton network is obtained (Kozma *et. al.*, 1998).
4. An adaptive training with the use of the other special type of network pruning, so called *Method of Training and Zeroing*. This is practically implemented by *zeroing* the small connection weights using a variable threshold. This connections can be left in the structure for further change or can be pruned. The used method is in contrast to the structural learning with forgetting method where the connections having small weights are gradually removed during the training process.

3.4. Rules extraction methods in FuNN

Several different methods for fuzzy rules extraction are applicable on the FuNN system. Fuzzy facts may have certainty factors (CF) attached to the conclusion, which show how certain is the fact and relative coefficients of importance (DI) of the condition elements in the antecedent, *noise tolerance* (NT) and *sensitivity factor* (SF) coefficients, have been introduced in the *generalised production rules* in addition to the confidence factors (CF) (Kasabov, 1996a, pp. 195). Simple rules without degrees of importance (DI) and a weighted rules with their associated weights representing degrees of importance (DI) and confidence factors (CF) can be extracted as explained in Kasabov (1996a; 1996b; 1998). One rule node is represented by several fuzzy rules each of them representing a combination of the input condition elements which would activate that node. These can be interpreted later in a classical fuzzy inference method outside of the FuNN system.

For interpreting a FuNN structure in terms of aggregated fuzzy rules an algorithm is also implemented (Kasabov *et. al.*, 1998). Each rule node is represented as one fuzzy rule. The strongest connection from a condition element node for an input variable to the rule node, along with the neighbouring condition element nodes, are represented in the associated rule. The connection weights of these connections are interpreted as degrees of importance (DI) attached to the corresponding condition elements.

The extracted rules from the FuNN can be inserted in the other FuNN modules through the *rule insertion module*.

4. Modelling and prediction of time-series in FuNN and in other fuzzy neural networks

4.1. Modelling of time-series in FuNN

In order to demonstrate the potential of the proposed FuNN to irregular time series processing the case study problem of gas-furnace data is used. In addition to the standard backpropagation, a structural learning algorithm with forgetting has been used. For details of the method and practical implementation, see Ishikawa (1996) and Kozma *et. al.* (1996).

For this purpose, the following experiment was performed: a 2-10-7-5-1 FuNN (see Fig. 3) was trained with the modified backpropagation algorithm and fixed MFs. After the first 200 iterations with *fixed learning mode*, forgetting was introduced with a forgetting factor of $\epsilon = 10^{-5}$ for more 1000 epochs. It is observed that if the learning rate (α) is small, the gradient method will closely approximate the gradient path, but convergence will be slow since the gradient must be calculated many times. On the other hand, if α is large, convergence will initially be very fast, but the algorithm will oscillate about the optimum. Based on these observations, α was variable during training and individually set for each of the layers in the FuNN, while the momentum and gain factor in the logistic activation function were 0.9 and 1, respectively, for layers 2 to 5. Fig. 4 shows the actual and the predicted values for the CO_2.

Using the aggregated rules extraction mode, seven rules were extracted from the trained FuNN, as shown next. As explained in Section 2, the FuNN model uses 2 inputs and one output. Five membership functions are attached to each input and output linguistic variable. Input1 and Input2 denote methane (t-4) and CO_2 (t-1), respectively, and Output1 denotes CO_2 concentration at the moment (t). A, B, C, D, and E show the fuzzy labels of five membership functions such as very small, small, medium, large, and very large, respectively. Extracted Rules for FuNN are shown in Table 1.

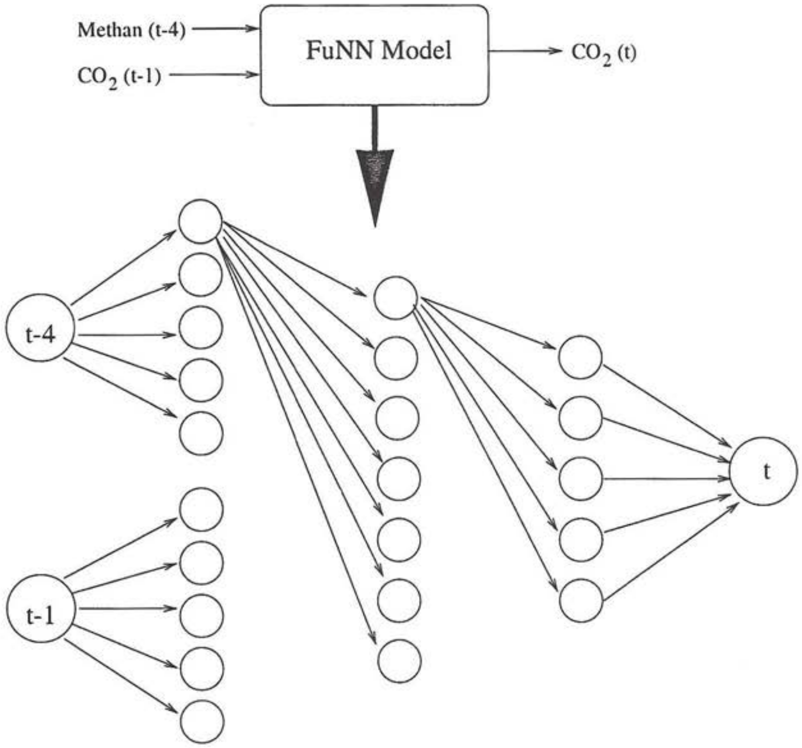

Figure 3. The FuNN architecture for gas-furnace.

Fuzzy rules	IF				THEN	
	x_1 *is*	*DI*	x_2 *is*	*DI*	y *is*	*CF*
1	D	6.290	E	12.071	A	2.76
2	A	5.390	B	6.025	E	1.979
3	B	0.687	D	5.969	D	1.574
4	B	1.839	D	4.138	B	1.248
5	D	2.463	E	3.674	B	2.12
6	B	4.283	B	17.155	B	1.787
7	E	2.654	A	4.082	C	1.795

Table 1. Fuzzy Rules generated from the FuNN: DI for *degree of importance*; CF for *certainty factor*

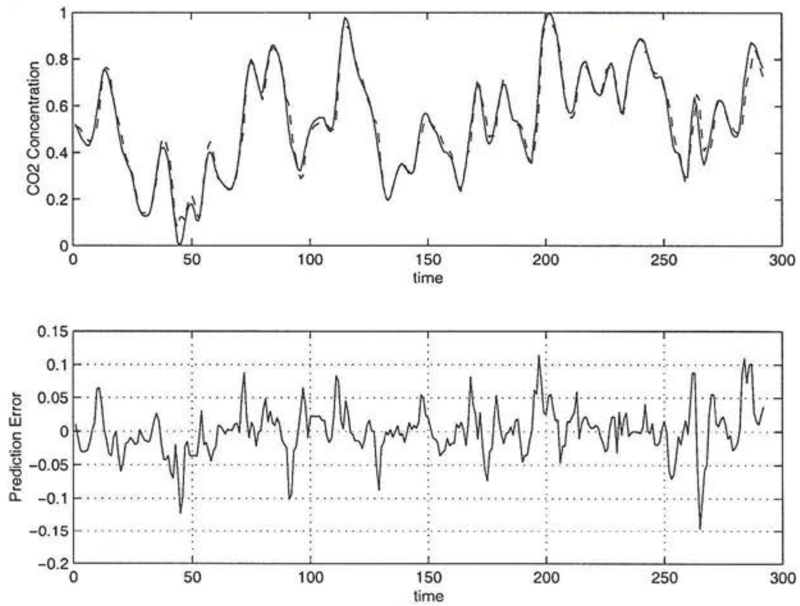

Figure 4. Model performance of FuNN with five linguistic labels on the Box and Jenkins gas furnace data. Actual data is shown by the solid line (—), model data by the dotted line.

4.2. Modelling time-series in ANFIS

The architectures and learning algorithms of ANFIS have been described in Jang (1993). ANFIS architecture is depicted in Fig. 5. The structure of ANFIS is a five-layer feedforward neural network with supervised learning capability which is functionally equivalent to fuzzy inference systems and is the same as Type II Fuzzy Neural Network in Horikawa *et.al.* (1990). Note that the links in the structure only indicated the flow direction of signal between layers. There are no adjustable weights that are associated with the links. The used MFs ($\mu_{A_i}(x)$) are bell-shaped with maximum equal to 1 and minimum equal to 0, such as

$$\mu_{A_i}(x) = \frac{1}{1 + [(\frac{x-c_i}{a_i})^2]^{b_i}}, \tag{15}$$

where (a_i, b_i, c_i) is the **premise parameter set**. As the values of these parameters change, the bell-shaped functions vary accordingly.

Let us briefly look at the architecture of ANFIS. Suppose that the we have the following two implications of first-order Takagi and Sugeno's type with two inputs x_1 and x_2 and one output z (Takagi & Sugeno, 1983):

R^1: If x_1 is $\mathbf{A}_1$ and x_2 is $\mathbf{B}_1$, then $y_1 = \mathbf{p}_1 x_1 + \mathbf{q}_1 x_2 + \mathbf{r}_1$,
R^2: If x_1 is $\mathbf{A}_2$ and x_2 is $\mathbf{B}_2$, then $y_2 = \mathbf{p}_2 x_1 + \mathbf{q}_2 x_2 + \mathbf{r}_2$.

This type of fuzzy reasoning and the corresponding equivalent **ANFIS** architecture is shown in Fig. 5.

A square node has parameters while a circle node has none. The node functions in the same layer are of the same function family as described below:

- **Layer 1:** Every node i in this layer is a square node with a node function
$$O_i^1 = \mu_{A_i}(x), \tag{16}$$
where x is the input to node i, and A_i is the linguistic label. O_i^1 is the membership function of A_i and it specifies the degree to which the given x satisfies the quantifier A_i. $\mu_{A_i}(x)$ can be bell-shaped with maximum equal to 1 and minimum equal to 0.
- **Layer 2:** Every node in this layer is a circle node labelled $\prod$ which multiplies the incoming signals and sends the product out. For instance,
$$w_i = \mu_{A_i}(x_1) \times \mu_{B_i}(x_2), \quad i = 1, 2. \tag{17}$$
Each node output represents the *firing strength* of a rule. In fact *T-norm* operators that perform generalised *AND* can be used as the node function in this layer.
- **Layer 3:** Every node in this layer is a circle node labelled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strength:
$$\bar{w} = \frac{w_i}{W_1 + W_2}, \quad i = 1, 2. \tag{18}$$
For convenience, outputs of this layer will be called *normalised firing strengths.*

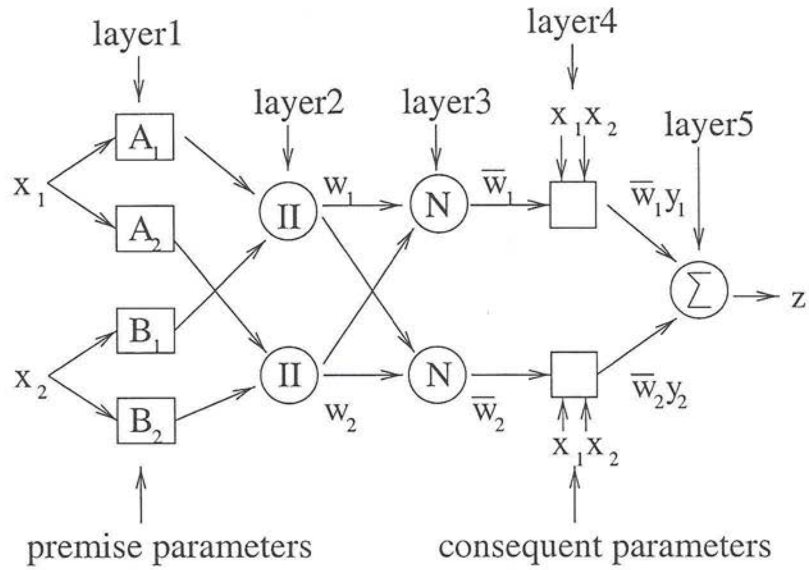

Figure 5. Takagi-Sugeno's type ANFIS

- **Layer 4:** Every node i in this layer is a square node with a node function
$$\begin{aligned} O_i^4 &= \bar{w}_i y_i \\ &= \bar{w}_i(p_i x_1 + q_i x_2 + r_i), \end{aligned} \tag{19}$$
where $\bar{w}_i$ is the output of layer 3, and (p_i, q_i, r_i) is the parameter set. Parameters in this layer will be referred to as *consequence parameters*.
- **Layer 5:** The single node in this layer is a circle node labelled $\sum$ that computes the overall output as the summation of all incoming signals, i.e.,
$$O_i^5 = z = \sum_i \bar{w}_i y_i = \frac{\sum_i w_i y_i}{\sum_i w_i}. \tag{20}$$

By using a hybrid learning rule (Jang, 1991) which combines the gradient method and the least squares estimate (LSE), ANFIS can achieve a desired input-output mapping in the form of Takagi-Sugeno's type fuzzy if-then rules (Takagi & Sugeno, 1983). The membership functions that form the premise part as well as MFs that form the consequence parts are parametrised. These premise parameters are updated according to given training data and a gradient-based learning procedure. Each element of outputs is a linear combination of input variables plus a constant term, so the parameters in the consequent parts can be identified by the least squares method.

Here the ANFIS has 5 membership functions on its input and batch learning paradigm was adapted with a learning rate $\eta = 0.1$. Thus the ANFIS used here contains 25 rules and the total number of fitting parameters is 105 which are composed of 30 premise parameters and 75 consequent parameters. Fig. 6

Fuzzy rules	IF		THEN		
	x_1 *is*	x_2 *is*	$f = px_1 + qx_2 + r$		
1	A	A	-20.7	16.33	-4.594
2	A	B	22.352	-10.756	3.679
3	A	C	5.905	1.223	-0.399
4	A	D	2.971	3.769	-1.526
5	A	E	-6.165	8.319	-6.363
6	B	A	-1.551	-8.058	3.575
7	B	B	0.807	9.272	-3.154
8	B	C	-0.647	1.05	0.113
9	B	D	0.482	-2.204	1.572
10	B	E	-0.288	-3.915	4.556
11	C	A	-7.235	-3.409	3.400
12	C	B	6.535	1.901	-2.414
13	C	C	-0.643	1.799	-0.777
14	C	D	4.0542	-3.452	-0.126
15	C	E	-5.069	-18.307	18.508
16	D	A	0.548	2.441	-0.008
17	D	B	-1.397	-2.819	0.979
18	D	C	-17.609	19.578	-2.753
19	D	D	0.726	-3.154	2.466
20	D	E	0.240	0.365	0.3905
21	E	A	0.855	0.055	-0.862
22	E	B	-0.999	2.916	0.665
23	E	C	8.529	-16.936	6.358
24	E	D	-0.127	4.527	-2.252
25	E	E	-0.017	0.950	0.073

Table 2. Fuzzy Rules generated from the ANFIS

demonstrates how ANFIS can model the gas-furnace time series.

The ANFIS structure after training can be interpreted as a set of Takagi-Sugeno type of fuzzy rules. Extracted rules (25 if-then rules) for ANFIS (Takagi-Sugeno Type) are described in Table 2.

FuNN and ANFIS use different inference fuzzy representations and different inference techniques. They have different strengths and FuNN uses fuzzy rules which are easy to be interpreted by the end users. It has a rich set of methods for training and adaptation. ANFIS uses Gaussian membership functions which may result in a better approximation of complex non-linear time series. Fig. 7 demonstrate how FuNN can effectively model a highly nonlinear surface as compared to ANFIS, but we did not attempt an exhaustive search to find the optimal settings for the ANFIS. The overall performance of FuNN and ANFIS

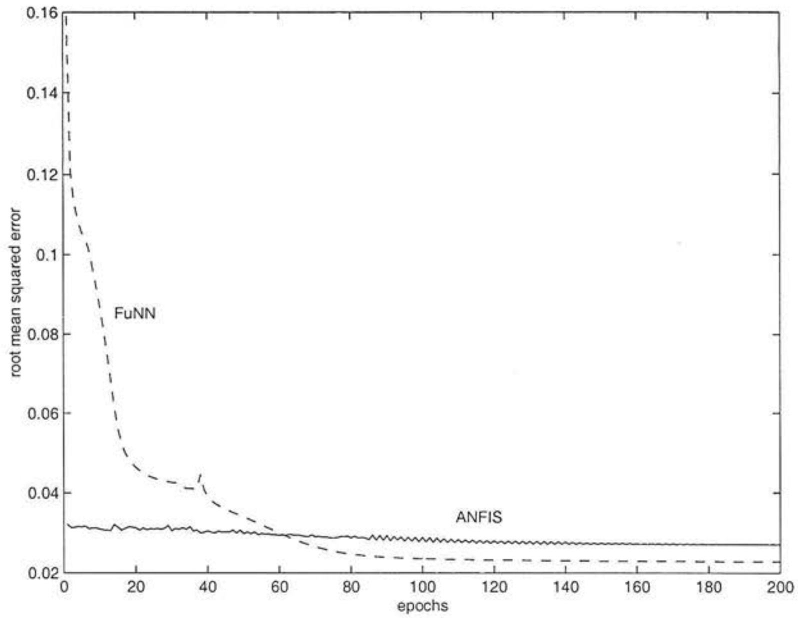

Figure 6. RMSE curves for the FuNN and the ANFIS.

is not much different in this example (c. f. Fig. 4 and Fig. 6), but the sets of extracted fuzzy rules differ significantly. Apart from the several basic types of fuzzy rules, i.e. Zadeh-Mamdani's fuzzy rules and Takagi-Sugeno's fuzzy rules, fuzzy rules having coefficients of uncertainty have often been used in practice. A fuzzy rule that contains a confidence factor (CF) of the validity of the conclusion has the form of:

```
if x is A, then y is B (CF or weight).
```

In addition, very often the condition elements in the antecedent part of the rule are not equally important for the rule to infer an output value. In this sense, we might see the rules extracted from FuNN (see Table 1) have more precise condition parts, allowing for degrees of importance (DI) to be extracted and used. The conclusion part of the ANFIS rules (see Table 2) complicate for the simple form of the condition parts.

5. Concluding remarks

Combined hybrid systems between neural networks and fuzzy logic are rapidly gaining popularity in the design of many complex systems. Experience shows that this type of combined system yields results sometimes superior to those obtained by the fuzzy control systems. Moreover, fuzzy neural networks are a promising paradigm in the area of *Soft Computing*. They have strengths in both learning from data and monitoring knowledge.

In this paper, the properties of FuNN and ANFIS are compared. It has been shown that FuNN is capable of extracting semantically meaningful rules, whereas the rules from ANFIS are not easily interpretable because the system was adapted Takagi-Sugeno type of an inference system. At the same time the accuracy of the predicting of FuNN is comparable or even better that that of ANFIS.

Future research is anticipated in applying adaptive fuzzy-genetic, neuro-genetic, and neuro-fuzzy-genetic systems. Such systems are likely to dominated the area of hybrid intelligent information systems in the near future. On-line adaptation of fuzzy neural networks is still to be investigated. That may well be the most important criterion to compare different fuzzy neural network structures.

Acknowledgements

This work is partially supported by the PGSF UOO 606 research grant funded by the New Zealand Foundation for Research, Science and Technology.

References

Box, G.E.P. and Jenkins, G.M. (1970) *Time series analysis, forecasting and control.* Holden Day, San Francisco.

Hauptmann, W. and Heesche, K. (1995) A Neural Net Topology for Bidirectional Fuzzy-Neuro Transformation. *Proc. of the Fuzz-IEEE/IFES*, Yokohama, Japan, 1511-1518.

Horikawa, S., Fruhashi, T., Okuma, S., and Uchikawa, Y. (1990) Composition Methods of Fuzzy Neural Networks. *Proc. of IEEE Industrial Electronics*, 1253-1258.

Ishibuchi, H., Tanaka, H., and Okada, H. (1994) Interpolation of fuzzy if-then rules by neural networks. *International Journal of Approximate Reasoning*, **10**, 1, 3-27.

Ishikawa, M. (1996) Structural learning with forgetting. *Neural Networks*, **9**, 3, 509-521.

Jang, J.S.R. (1991) Fuzzy modelling using generalised neural networks and Kalman filter algorithm. *Proc. Ninth Nat. Conf. Artificial Intell. (AAAI-91)*, July, 762-767.

Jang, J.S.R. (1993) ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Transactions on Systems, Man and Cybernetics*, **23**, 3, 665-685.

Kasabov, N. (1996A) *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering.* The MIT Press, CA, MA.

Kasabov, N. (1996B) Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy Sets and Systems*, **82**, 135-149.

KASABOV, N., KIM, J., WATTS, M., and GRAY, A. (1997) FuNN/2–A fuzzy neural network architecture for adaptive learning and knowledge acquisition. *Information Sciences*, **101**, 3, 155-175.

KATAYAMA, R., KUWATA, K., KAJITANI, Y., and WATANABE M. (1995) Embedding dimension estimation of chaotic time series using self-generating radial basis function network. *Fuzzy Sets and Systems*, **71**, 3, 311-327.

KOZMA, R., KASABOV, N., KIM, J., and COHEN, T. (1998) Integration of connectionist methods and chaotic time series analysis for the prediction of process data. *Int. J. of Intelligent Systems*, **13**, 6, 519-538.

KOZMA, R., SAKUMA, M., YOKOYAMA, Y., and KITAMURA, M. (1996) On the accuracy of mapping by neural networks trained by backpropagation with forgetting. *Neurocomputing*, **13**, 295-311.

LIN, C.T. and LEE, C.S.G. (1991) Neural-networks-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, **40**, 12, 1320-1366.

TAKAGI, T. and SUGENO, M. (1983) Derivation of fuzzy control rules from human operator's control actions. *Proc. IFAC Symp. Fuzzy Inform. Knowledge Representation and Decision Analysis*, July, 55-60.

YAMAKAWA, T., UCHINO, E., MIKI, T., and KUSANAGI, H. (1992) A Neo Fuzzy Neuron and Its Application to System Identification and prediction of the System Behaviour. *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, 477-483.

ZADEH, L.A. (1965) Fuzzy Sets. *Information and Control*, **8**, 338-353.

ZADEH, L.A. (1973) Outline of a new approach to the analysis of complex systems and decision process. *IEEE Transactions on Systems, Man, and Cybernetics*, **3**, 1, 28-44.