[1]

ANFIS:

Adaptive-Network-Based Fuzzy Inference System

Jyh-Shing Roger Jang

Department of Electrical Engineering and Computer Science

University of California, Berkeley, CA 94720

Abstract

This paper presents the architecture and learning procedure underlying ANFIS (Adaptive-Network-based Fuzzy Inference System), a fuzzy inference system implemented in the framework of adaptive networks. By using a hybrid learning procedure, the proposed ANFIS can construct an input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and stipulated input-output data pairs. In our simulation, we employ the ANFIS architecture to model nonlinear functions, identify nonlinear components on-linely in a control system, and predict a chaotic time series, all yielding remarkable results. Comparisons with artificail neural networks and earlier work on fuzzy modeling are listed and discussed. Other extensions of the proposed ANFIS and promising applications to automatic control and signal processing are also suggested.

1 Introduction

System modeling based on conventional mathematical tools (e.g., differential equations) is not well suited for dealing with ill-defined and uncertain systems. By contrast, a fuzzy inference system employing fuzzy if-then rules can model the qualitative aspects of human knowledge and

---

*Research supported in part by NASA Grant NCC-2-275; LLNL Grant No. ISCR 89-12; MICRO State Program Award No. 90-191; and OMRON Corporation.

reasoning processes without employing precise quantitative analyses. This *fuzzy modeling* or *fuzzy identification*, first explored systematically by Takagi and Sugeno [50], has found numerous practical applications in control [42, 34], prediction and inference [16, 17]. However, there are some basic aspects of this approach which are in need of better understanding. More specifically:

1. No standard methods exist for transforming human knowledge or experience into the rule base and database of a fuzzy inference system.

2. There is a need for effective methods for tuning the membership functions (MF's) so as to minimize the output error measure or maximize performance index.

In this perspective, the aim of this paper is to suggest a novel architecture called *Adaptive-Network-Based Fuzzy Inference System*, or simply *ANFIS*, which can serve as a basis for constructing a set of fuzzy if-then rules with appropriate membership functions to generate the stipulated input-output pairs. The next section introduces the basics of fuzzy if-then rules and fuzzy inference systems. Section 3 describes the structures and learning rules of adaptive networks. By embedding the fuzzy inference system into the framework of adaptive networks, we obtain the ANFIS architecture which is the backbone of this paper and it is covered in section 4. Application examples such as nonlinear function modeling and chaotic time series prediction are given in section 5. Section 6 concludes this paper by giving important extensions and future directions of this work.

2 Fuzzy If-Then Rules and Fuzzy Inference Systems

A. Fuzzy If-Then Rules

Fuzzy if-then rules or *fuzzy conditional statements* are expressions of the form *IF A THEN B*, where A and B are labels of *fuzzy sets* [62] characterized by appropriate membership functions. Due to their concise form, fuzzy if-then rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. An example that describes a simple fact is

If pressure is high, then volume is small.

where *pressure* and *volume* are *linguistic variables* [63], *high* and *small* are *linguistic values* or *labels* that are characterized by membership functions.

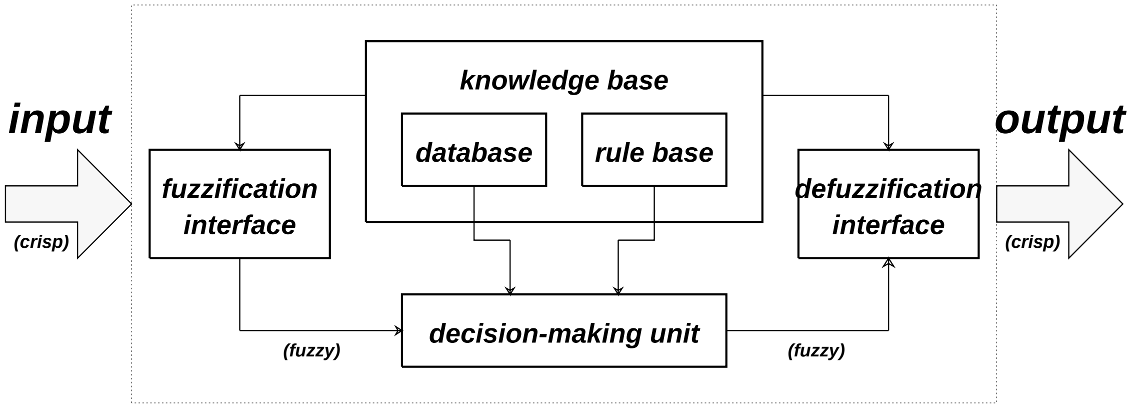


Figure 1: *Fuzzy inference system.*

Another form of fuzzy if-then rule, proposed by Takagi and Sugeno [49], has fuzzy sets involved only in the premise part. By using Takagi and Sugeno's fuzzy if-then rule, we can describe the resistant force on a moving object as follows:

$$\textit{If velocity is high, then } force = k * (velocity)^2.$$

where, again, *high* in the premise part is a linguistic label characterized by an appropriate membership function. However, the consequent part is described by a nonfuzzy equation of the input variable, velocity.

Both types of fuzzy if-then rules have been used extensively in both modeling and control. Through the use of linguistic labels and membership functions, a fuzzy if-then rule can easily capture the spirit of a "rule of thumb" used by humans. From another angle, due to the qualifiers on the premise parts, each fuzzy if-then rule can be viewed as a local description of the system under consideration. Fuzzy if-then rules form a core part of the fuzzy inference system to be introduced below.

A. Fuzzy Inference Systems

Fuzzy inference systems are also known as *fuzzy-rule-based systems*, *fuzzy models*, *fuzzy associative memories (FAM)*, or *fuzzy controllers* when used as controllers. Basically a fuzzy inference system is composed of five functional blocks (Figure 1):

- a **rule base** containing a number of fuzzy if-then rules;
- a **database** which defines the membership functions of the fuzzy sets used in the fuzzy rules;

- a **decision-making unit** which performs the inference operations on the rules;
- a **fuzzification interface** which transforms the crisp inputs into degrees of match with linguistic values;
- a **defuzzification interface** which transform the fuzzy results of the inference into a crisp output.

Usually, the rule base and the database are jointly referred to as the *knowledge base*.

The steps of *fuzzy reasoning* (inference operations upon fuzzy if-then rules) performed by fuzzy inference systems are:

1. Compare the input variables with the membership functions on the premise part to obtain the membership values (or compatibility measures) of each linguistic label. (This step is often called *fuzzification*).
2. Combine (through a specific T-norm operator, usually multiplication or min.) the membership values on the premise part to get *firing strength* (*weight*) of each rule.
3. Generate the qualified consequent (either fuzzy or crisp) of each rule depending on the firing strength.
4. Aggregate the qualified consequents to produce a crisp output. (This step is called *defuzzification*.)

Several types of fuzzy reasoning [22, 23] have been proposed in the literature. Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types (Figure 2):

Type 1: The overall output is the weighted average of each rule's crisp output induced by the rule's firing strength (the product or minimum of the degrees of match with the premise part) and output membership functions. The output membership functions used in this scheme must be monotonically non-decreasing [51].

Type 2: The overall fuzzy output is derived by applying "max" operation to the qualified fuzzy outputs (each of which is equal to the minimum of firing strength and the output membership function of each rule). Various schemes have been proposed to choose the final crisp output based on the overall fuzzy output; some of them are center of area, bisector of area, mean of maxima, maximum criterion, etc [22, 23].

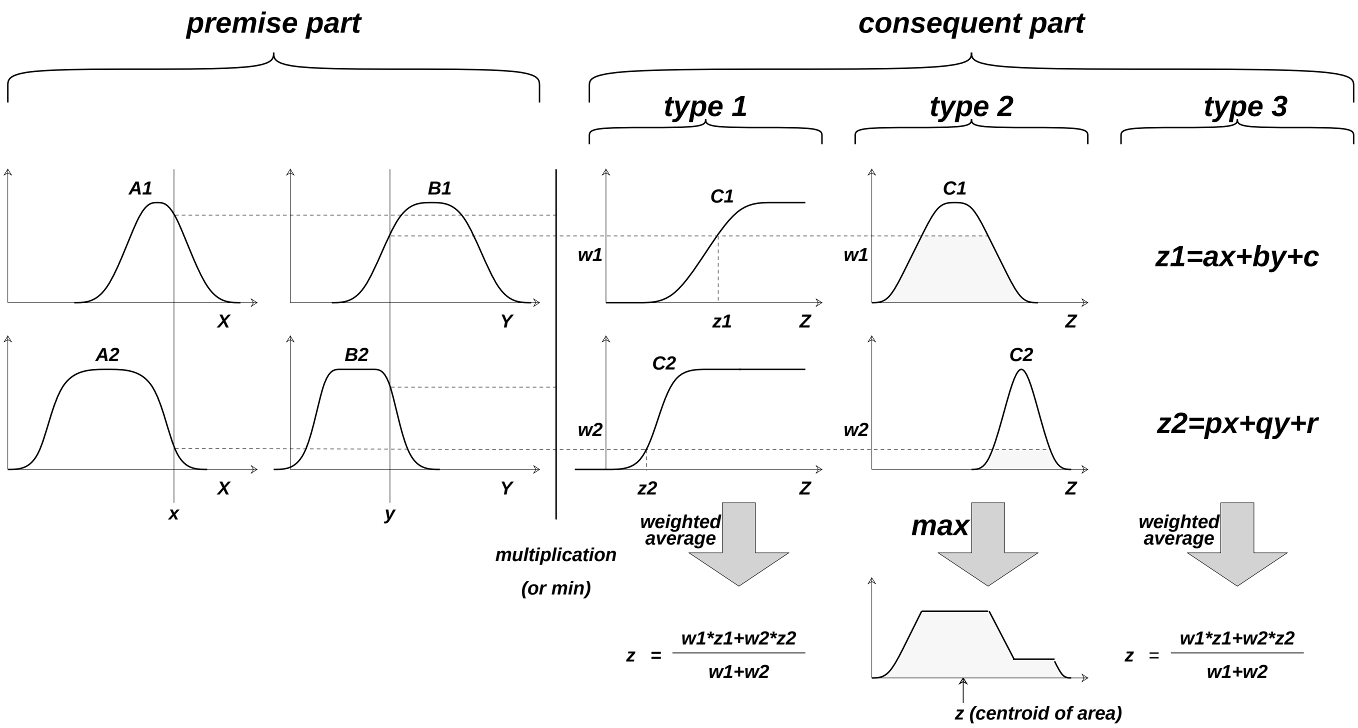


Figure 2: *Commonly used fuzzy if-then rules and fuzzy reasoning mechanisms*

Type 3: Takagi and Sugeno's fuzzy if-then rules are used [49]. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output.

Figure 2 utilizes a two-rule two-input fuzzy inference system to show different types of fuzzy rules and fuzzy reasoning mentioned above. Be aware that most of the differences come from the specification of the consequent part (monotonically non-decreasing or bell-shaped membership functions, or crisp function) and thus the defuzzification schemes (weighted average, centroid of area, etc) are also different.

3 Adaptive Networks: Architectures and Learning Algorithms

This section introduces the architecture and learning procedure of the adaptive network which is in fact a superset of all kinds of feedforward neural networks with supervised learning capability. An adaptive network, as its name implies, is a network structure consisting of nodes and directional

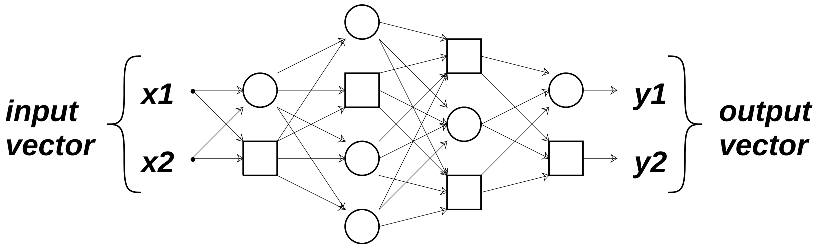


Figure 3: *An adaptive network.*

links through which the nodes are connected. Moreover, part or all of the nodes are adaptive, which means each output of these nodes depends on the parameter(s) pertaining to this node, and the learning rule specifies how these parameters should be changed to minimize a prescribed error measure.

The basic learning rule of adaptive networks is based on the gradient descent and the chain rule, which was proposed by by Werbos [57] in the 1970's. However, due to the state of artificial neural network research at that time, Werbos' early work failed to receive the attention it deserved. In the following presentation, the derivation is based on the author's work [11, 10] which generalizes the formulas in [37].

Since the basic learning rule is based the gradient method which is notorious for its slowness and tendency to become trapped in local minima, here we propose a hybrid learning rule which can speed up the learning process substantially Both the batch learning and the pattern learning of the proposed hybrid learning rule is discussed below, though our simulations are mostly based on the batch learning.

A. Architecture and Basic Learning Rule

An adaptive network (Figure 3) is a multi-layer feedforward network in which each node performs a particular function (*node function*) on incoming signals as well as a set of parameters pertaining to this node. The nature of the node functions may vary from node to node, and the choice of each node function depends on the overall input-output function which the adaptive network is required to carry out. Note that the links in an adaptive network only indicate the flow direction of signals between nodes; no weights are associated with the links.

To reflect different adaptive capabilities, we use both circle and square nodes in an adaptive network. A square node (adaptive node) has parameters while a circle node (fixed node) has none. The parameter set of an adaptive network is the union of the parameter sets of each adaptive node.

In order to achieve a desired input-output mapping, these parameters are updated according to given training data and a gradient-based learning procedure described below.

Suppose that a given adaptive network has L layers and the k-th layer has $\#(k)$ nodes. We can denote the node in the i-th position of the k-th layer by (k, i), and its node function (or node output) by O_i^k. Since a node output depends on its incoming signals and its parameter set, we have

$$O_i^k = O_i^k(O_1^{k-1}, \ldots O_{\#(k-1)}^{k-1}, a, b, c, \ldots), \tag{1}$$

where a, b, c, etc. are the parameters pertaining to this node. (Note that we use O_i^k as both the node output and node function.)

Assuming the given training data set has P entries, we can define the *error measure* (or *energy function*) for the p-th $(1 \leq p \leq P)$ entry of training data entry as the sum of squared errors:

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - O_{m,p}^L)^2, \tag{2}$$

where $T_{m,p}$ is the m-th component of p-th target output vector, and $O_{m,p}^L$ is the m-th component of actual output vector produced by the presentation of the p-th input vector. Hence the overall error measure is $E = \sum_{p=1}^{P} E_p$.

In order to develop a learning procedure that implements gradient descent in E over the parameter space, first we have to calculate the *error rate* $\frac{\partial E_p}{\partial O}$ for p-th training data and for each node output O. The error rate for the output node at (L, i) can be calculated readily from equation (2):

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2(T_{i,p} - O_{i,p}^L). \tag{3}$$

For the internal node at (k, i), the error rate can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k}, \tag{4}$$

where $1 \leq k \leq L-1$. That is, the error rate of an internal node can be expressed as a linear combination of the error rates of the nodes in the next layer. Therefore for all $1 \leq k \leq L$ and $1 \leq i \leq \#(k)$, we can find $\frac{\partial E_p}{\partial O_{i,p}^k}$ by equation (3) and (4).

Now if α is a parameter of the given adaptive network, we have

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}, \tag{5}$$

where S is the set of nodes whose outputs depend on α. Then the derivative of the overall error measure E with respect to α is

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}. \tag{6}$$

Accordingly, the update formula for the generic parameter α is

$$\triangle \alpha = -\eta \frac{\partial E}{\partial \alpha}, \tag{7}$$

in which η is a learning rate which can be further expressed as

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} (\frac{\partial E}{\partial \alpha})^2}}, \tag{8}$$

where k is the *step size*, the length of each gradient transition in the parameter space. Usually, we can change the value of k to vary the speed of convergence. The heuristic rules for changing k are discussed in the section 5 where we report simulation results.

Actually, there are two learning paradigms for adaptive networks. With the *batch learning* (or *off-line learning*), the update formula for parameter α is based on equation (6) and the update action takes place only after the whole training data set has been presented, i.e., only after each *epoch* or *sweep*. On the other hand, if we want the parameters to be updated immediately after each input-output pair has been presented, then the update formula is based on equation (5) and it is referred to as the *pattern learning* (or *on-line learning*). In the following we will derive a faster hybrid learning rule and both of its learning paradigms.

B. Hybrid Learning Rule: Batch (Off-Line) Learning

Though we can apply the gradient method to identify the parameters in an adaptive network, the method is generally slow and likely to become trapped in local minima. Here we propose a hybrid learning rule [10] which combines the gradient method and the least squares estimate (LSE) to identify parameters.

For simplicity, assume that the adaptive network under consideration has only one output

$$output = F(\vec{I}, S), \tag{9}$$

where $\vec{I}$ is the set of input variables and S is the set of parameters. If there exists a function H such that the composite function $H \circ F$ is linear in some of the elements of S, then these elements can be identified by the least squares method. More formally, if the parameter set S can be decomposed into two sets

$$S = S_1 \oplus S_2, \tag{10}$$

(where $\oplus$ represents direct sum) such that $H \circ F$ is linear in the elements of S_2, then upon applying H to equation (9), we have

$$H(output) = H \circ F(\vec{I}, S), \tag{11}$$

which is linear in the elements of S_2. Now given values of elements of S_1, we can plug P training data into equation (11) and obtain a matrix equation:

$$AX = B \tag{12}$$

where X is an unknown vector whose elements are parameters in S_2. Let $|S_2| = M$, then the dimensions of A, X and B are $P \times M$, $M \times 1$ and $P \times 1$, respectively. Since P (number of training data pairs) is usually greater than M (number of linear parameters), this is an overdetermined problem and generally there is no exact solution to equation (12). Instead, a *least squares estimate* (*LSE*) of X, X^*, is sought to minimize the squared error $\|AX - B\|^2$. This is a standard problem that forms the grounds for linear regression, adaptive filtering and signal processing. The most well-known formula for X^* uses the pseudo-inverse of X:

$$X^* = (A^T A)^{-1} A^T B, \tag{13}$$

where A^T is the transpose of A, and $(A^T A)^{-1} A^T$ is the pseudo-inverse of A if $A^T A$ is non-singular. While equation (13) is concise in notation, it is expensive in computation when dealing with the matrix inverse and, moreover, it becomes ill-defined if $A^T A$ is singular. As a result, we employ sequential formulas to compute the LSE of X. This sequential method of LSE is more efficient (especially when M is small) and can be easily modified to an on-line version (see below) for systems with changing characteristics. Specifically, let the ith row vector of matrix A defined in equation (12) be a_i^T and the ith element of B be b_i^T, then X can be calculated iteratively using the sequential formulas widely adopted in the literature [1, 7, 25, 41]:

$$\left.\begin{array}{rcl} X_{i+1} & = & X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} & = & S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \quad i = 0, 1, \cdots, P-1 \end{array}\right\}, \tag{14}$$

where S_i is often called the *covariance matrix* and the least squares estimate X^* is equal to X_P. The initial conditions to bootstrap equation (14) are $X_0 = 0$ and $S_0 = \gamma I$, where γ is a positive large number and I is the identity matrix of dimension $M \times M$. When dealing with multi-output adaptive networks (*output* in equation (9) is a column vector), equation (14) still applies except that b_i^T is the i-th rows of matrix B.

Now we can combine the gradient method and the least squares estimate to update the parameters in an adaptive network. Each epoch of this hybrid learning procedure is composed of a forward pass and a backward pass. In the forward pass, we supply input data and functional signals go forward to calculate each node output until the matrices A and B in equation (12) are obtained, and the parameters in S_2 are identified by the sequential least squares formulas in equation (14). After identifying parameters in S_2, the functional signals keep going forward till the error measure is calculated. In the backward pass, the error rates (the derivative of the error measure w.r.t. each node output, see equation(3) and (4)) propagate from the output end toward the input end, and the parameters in S_1 are updated by the gradient method in equation (7).

For given fixed values of parameters in S_1, the parameters in S_2 thus found are guaranteed to be the global optimum point in the S_2 parameter space due to the choice of the squared error measure. Not only can this hybrid learning rule decrease the dimension of the search space in the gradient method, but, in general, it will also cut down substantially the convergence time.

Take for example an one-hidden-layer back-propagation neural network with sigmoid activation functions. If this neural network has p output units, then the *output* in equation (9) is a column vector. Let $H(\cdot)$ be the inverse sigmoid function

$$H(x) = ln(\frac{x}{1-x}), \tag{15}$$

then equation (11) becomes a linear (vector) function such that each element of $H(output)$ is a linear combination of the parameters (weights and thresholds) pertaining to layer 2. In other words,

S_1 = weights and thresholds of hidden layer,

S_2 = weights and thresholds of output layer.

Therefore we can apply the back-propagation learning rule to tune the parameters in the hidden layer, and the parameters in the output layer can be identified by the least squares method. However, it should be keep in mind that by using the least squares method on the data transformed by $H(\cdot)$, the obtained parameters are optimal in terms of the transformed squared error measure instead of the original one. Usually this will not cause practical problem as long as $H(\cdot)$ is monotonically increasing.

C. Hybrid Learning Rule: Pattern (On-Line) Learning

If the parameters are updated after each data presentation, we have the *pattern learning* or *on-line learning* paradigm. This learning paradigm is vital to the on-line parameter identification for systems with changing characteristics. To modify the batch learning rule to its on-line version, it is obvious that the gradient descent should be based on E_p (see equation (5)) instead of E. Strictly speaking, this is not a truly gradient search procedure to minimize E, yet it will approximate to one if the learning rate is small.

For the sequential least squares formulas to account for the time-varying characteristics of the incoming data, we need to decay the effects of old data pairs as new data pairs become available. Again, this problem is well studied in the adaptive control and system identification literature and a number of solutions are available [7]. One simple method is to formulate the squared error measure as a weighted version that gives higher weighting factors to more recent data pairs. This amounts to the addition of a *forgetting factor* λ to the original sequential formula:

$$\left.\begin{array}{rcl} X_{i+1} & = & X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} & = & \frac{1}{\lambda}[S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}}] \end{array}\right\}, \tag{16}$$

where the value of λ is between 0 and 1. The smaller *lambda* is, the faster the effects of old data decay. But a small *lambda* sometimes causes numerical unstability and should be avoided.

4 ANFIS: Adaptive-Network-Based Fuzzy Inference System

The architecture and learning rules of adaptive networks have been described in the previous section. Functionally, there are almost no constraints on the node functions of an adaptive network except piecewise differentiability. Structurally, the only limitation of network configuration is that it should be of feedforward type. Due to these minimal restrictions, the adaptive network's applications are immediate and immense in various areas. In this section, we propose a class of adaptive networks which are functionally equivalent to fuzzy inference systems. The proposed architecture is referred to as *ANFIS*, standing for *Adaptive-Network-based Fuzzy Inference System*. We describe how to decompose the parameter set in order to apply the hybrid learning rule. Besides, we demonstrate how to apply the Stone-Weierstrass theorem to ANFIS with simplified

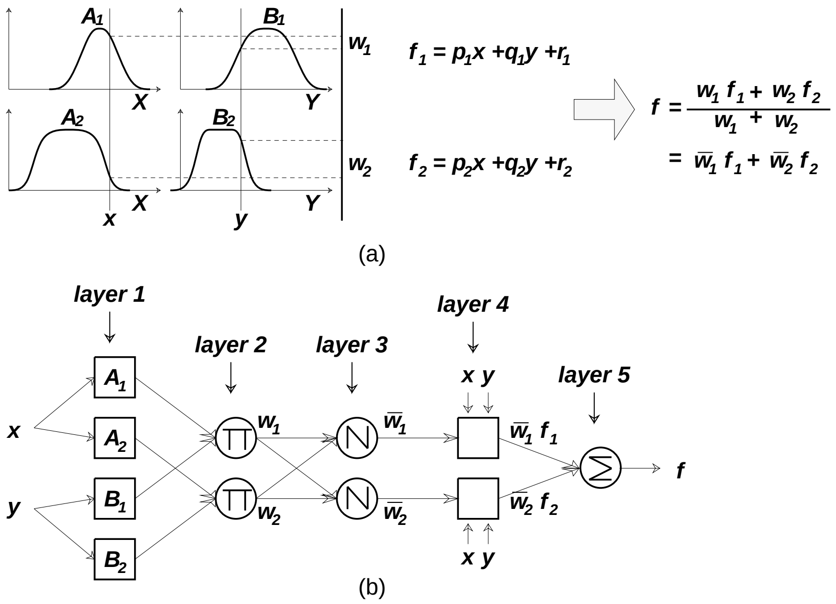


Figure 4: *(a) Type-3 fuzzy reasoning; (b) equivalent ANFIS (type-3 ANFIS).*

fuzzy if-then rules and how the radial basis function network relate to this kind of simplified ANFIS.

A. ANFIS architecture

For simplicity, we assume the fuzzy inference system under consideration has two inputs x and y and one output z. Suppose that the rule base contains two fuzzy if-then rules of Takagi and Sugeno's type [49]:

$$\text{Rule 1: If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \text{ then } f_1 = p_1 x + q_1 y + r_1,$$
$$\text{Rule 2: If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \text{ then } f_2 = p_2 x + q_2 y + r_2.$$

then the type-3 fuzzy reasoning is illustrated in Figure 4(a), and the corresponding equivalent ANFIS architecture (*type-3 ANFIS*) is shown in Figure 4(b). 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

$$O_i^1 = \mu_{A_i}(x), \tag{17}$$

where x is the input to node i, and A_i is the linguistic label (*small* , *large*, etc.) associated with this node function. In other words, 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. Usually we choose $\mu_{A_i}(x)$ to be bell-shaped with maximum equal to 1 and minimum equal to 0, such as

$$\mu_{A_i}(x) = \frac{1}{1 + [(\frac{x-c_i}{a_i})^2]^{b_i}}, \tag{18}$$

or

$$\mu_{A_i}(x) = exp\{-[(\frac{x - c_i}{a_i})^2]^{b_i}\}, \tag{19}$$

where $\{a_i,\ b_i,\ c_i\}$ is the parameter set. As the values of these parameters change, the bell-shaped functions vary accordingly, thus exhibiting various forms of membership functions on linguistic label A_i. In fact, any continuous and piecewise differentiable functions, such as commonly used trapezoidal or triangular-shaped membership functions, are also qualified candidates for node functions in this layer. Parameters in this layer are referred to as *premise parameters*.

Layer 2 Every node in this layer is a circle node labeled Π which multiplies the incoming signals and sends the product out. For instance,

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y),\ \ i = 1, 2. \tag{20}$$

Each node output represents the firing strength of a rule. (In fact, other *T-norm* operators that perform generalized AND can be used as the node function in this layer.)

Layer 3 Every node in this layer is a circle node labeled N. The i-th node calculates the ratio of the i-th rule's firing strength to the sum of all rules' firing strengths:

$$\overline{w}_i = \frac{w_i}{w_1 + w_2},\ \ i = 1, 2. \tag{21}$$

For convenience, outputs of this layer will be called called *normalized firing strengths*.

Layer 4 Every node i in this layer is a square node with a node function

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i), \tag{22}$$

where $\overline{w}_i$ 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 *consequent parameters*.

Layer 5 The single node in this layer is a circle node labeled Σ that computes the overall output as the summation of all incoming signals, i.e.,

$$O_1^5 = overall\ output = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{23}$$

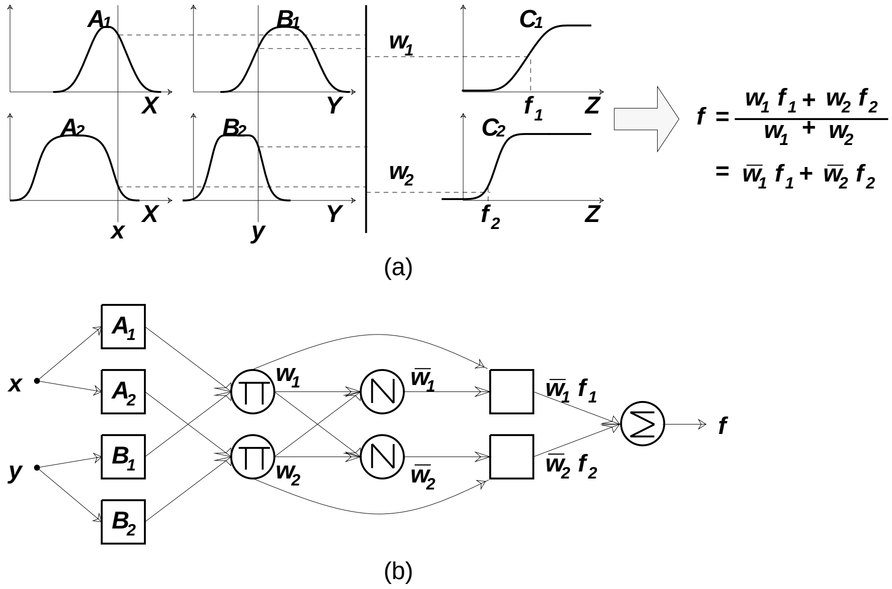


Figure 5: *(1) Type-1 fuzzy reasoning; (b) equivalent ANFIS (type-1 ANFIS).*

Thus we have constructed an adaptive network which is functionally equivalent to a type-3 fuzzy inference system. For type-1 fuzzy inference systems, the extension is quite straightforward and the type-1 ANFIS is shown in Figure 5 where the output of each rule is induced jointly by the output membership funcion and the firing strength. For type-2 fuzzy inference systems, if we replace the centroid defuzzification operator with a discrete version which calculates the approximate centroid of area, then type-3 ANFIS can still be constructed accordingly. However, it will be more complicated than its type-3 and type-1 versions and thus not worth the efforts to do so.

Figure 6 shows a 2-input, type-3 ANFIS with 9 rules. Three membership functions are associated with each input, so the input space is partitioned into 9 fuzzy subspaces, each of which is governed by a fuzzy if-then rules. The premise part of a rule delineates a fuzzy subspace, while the consequent part specifies the output within this fuzzy subspace.

B. Hybrid Learning Algorithm

From the proposed type-3 ANFIS architecture (Figure 4), it is observed that given the values of premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output f in Figure 4 can be rewritten as

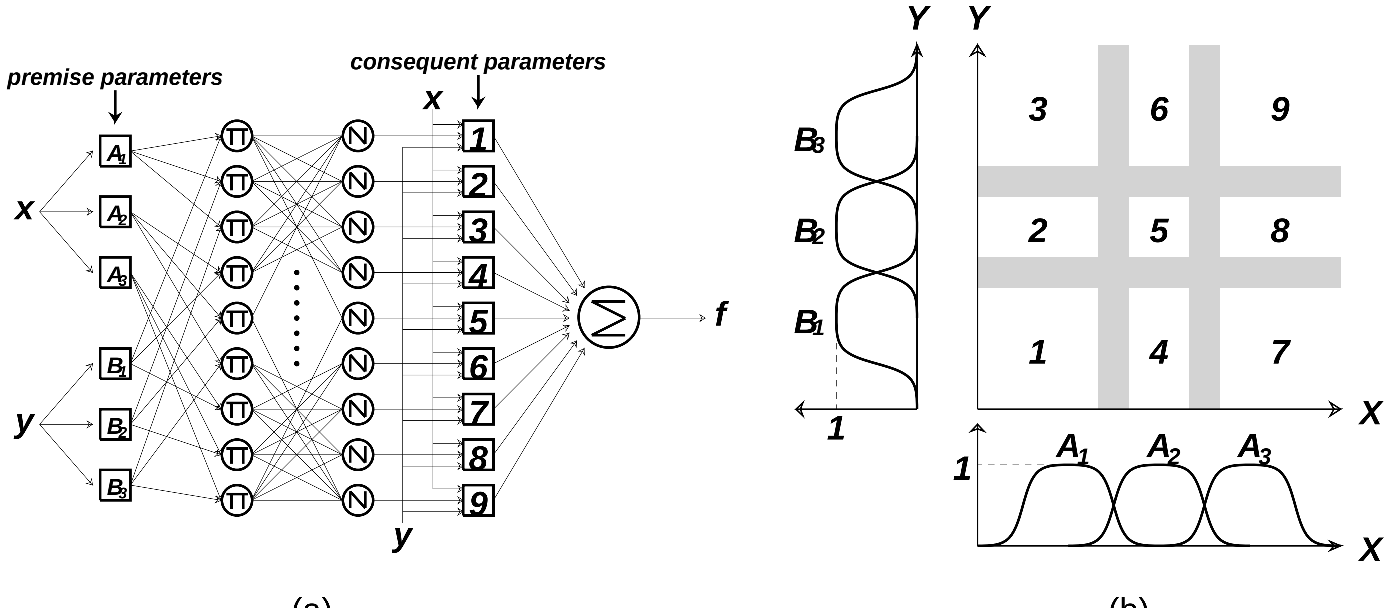


Figure 6: *(a) 2-input type-3 ANFIS with 9 rules; (b) corresponding fuzzy subspaces .*

$$\begin{aligned} f &= \frac{w_1}{w_1+w_2} f_1 + \frac{w_2}{w_1+w_2} f_2 \\ &= \overline{w}_1 f_1 + \overline{w}_2 f_2 \\ &= (\overline{w}_1 x) p_1 + (\overline{w}_1 y) q_1 + (\overline{w}_1) r_1 + (\overline{w}_2 x) p_2 + (\overline{w}_2 y) q_2 + (\overline{w}_2) r_2, \end{aligned} \tag{24}$$

which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2 and r_2). As a result, we have

S = set of total parameters,

S_1 = set of premise parameters,

S_2 = set of consequent parameters,

in equation (10); $H(\cdot)$ and $F(\cdot, \cdot)$ are the identity function and the function of the fuzzy inference system, respectively. Therefore the hybrid learning algorithm developed in the previous chapter can be applied directly. More specifically, in the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. Table 1 summarizes the activities in each pass.

As mentioned earlier, the consequent parameters thus identified are optimal (in the consequent parameter space) under the condition that the premise parameters are fixed. Accordingly the hybrid approach is much faster than the strict gradient descent and it is worthwhile to look for the possibility of decomposing the parameter set in the manner of equation (10). For type-1

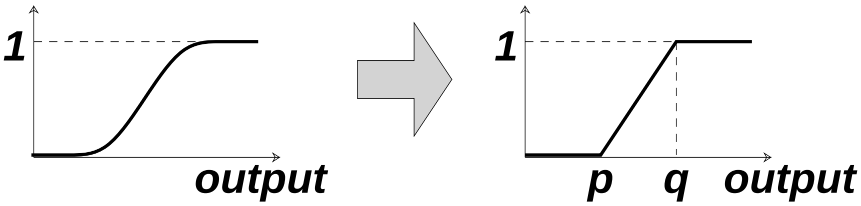


Figure 7: *Piecewise linear approximation of membership functions on the consequent part of type-1 ANFIS.*

ANFIS, this can be achieved if the membership function on the consequent part of each rule is replaced by a piecewise linear approximation with two consequent parameters (Figure 7). In this case, again, the consequent parameters constitute set S_2 and the hybrid learning rule can be employed directly.

However, it should be noted that the computation complexity of the least squares estimate is higher than that of the gradient descent. In fact, there are four methods to update the parameters, as listed below according to their computation complexities:

1. *Gradient descent only* : all parameters are updated by the gradient descent.
2. *Gradient descent and one pass of LSE* : the LSE is applied only once at the very beginning to get the initial values of the consequent parameters and then the gradient descent takes over to update all parameters.
3. *Gradient descent and LSE* : this is the proposed hybrid learning rule.
4. *Sequential LSE only* : the ANFIS is linearized w.r.t. all parameters and the extended kalman filter algorithm is employed to update all parameters. This has been proposed in the neural network literature [40, 39].

The choice of above methods should be based on the trade-off between computation complexity and resulting performance. Our simulations presented in the next section are performed by the

-	forward pass	backward pass
premise parameters	fixed	gradient descent
consequent parameters	least squares estimate	fixed
signals	node outputs	error rates

Table 1: *Two passes in the hybrid learning procedure for ANFIS.*

third method.

Because the update formulas of the premise and consequent parameters are decoupled in the hybrid learning rule (see Table 1), further speedup of learning is possible by using other versions of the gradient method on the premise parameters, such as conjugate gradient descent, second-order back-propagation [33], quick-propagation [5], nonlinear optimization [54] and many others.

C. Fuzzy Inference Systems with Simplified Fuzzy If-Then Rules

Though the reasoning mechanisms (Figure 2) introduced earlier are commonly used in the literature, each of them has inherent drawbacks. For type-1 reasoning (Figure 2 or 5), the membership functions on the consequence part are restricted to monotonically non-decreasing functions which are not compatible with linguistic terms such as "medium" whose membership function should be bell-shaped. For type-2 reasoning (Figure 2), the defuzzification process is time-consuming and systematic fine-tuning of the parameters are not easy. For type-3 reasoning (Figure 2 or 4), it is just hard to assign any appropriate linguistic terms to the consequence part which is bilinear in the input variables. To cope with these disadvantages, simplified fuzzy if-then rules of the following form are introduced:

If x is big and y is small, then z is d.

where d is a *crisp* value. Due to the fact that the output z is described by a crisp value (or equivalently, a singular membership function), this class of simplified fuzzy if-then rules can employ all three types of reasoning mechanisms. More specifically, the consequent part of this simplified fuzzy if-then rule is represented by a step function (centered at $z = d$) in type 1, a singular membership function (at $z = d$) in type 2, and a constant output function in type 3, respectively. The three reasoning mechanisms are unified under this simplified fuzzy if-then rules.

Most of all, with this simplified fuzzy if-then rule, it is possible to prove that under certain circumstance, the resulting fuzzy inference system has unlimited approximation power to match any nonlinear functions arbitrarily well on a compact set. We will proceed this in a descriptive way by applying the Stone-Weierstrass theorem [18, 36] stated below.

Theorem 4.1 *Let domain D be a compact space of N dimensions, and let $\mathcal{F}$ be a set of continuous real-valued functions on D, satisfying the following criteria:*

1. **Identity function***: The constant $f(x) = 1$ is in $\mathcal{F}$.*

2. **Separability**: *For any two points* $x_1 \neq x_2$ *in* D*, there is an* f *in* $\mathcal{F}$ *such that* $f(x_1) \neq f(x_2)$.

3. **Algebraic closure**: *If* f *and* g *are any two functions in* $\mathcal{F}$*, then* fg *and* $af + bg$ *are in* F *for any two real numbers* a *and* b.

Then $\mathcal{F}$ *is dense in* $C(D)$*, the set of continuous real-valued functions on* D*. In other words, for any* $\epsilon > 0$*, and any function* g *in* $C(D)$*, there is a function* f *in* $\mathcal{F}$ *such that* $|g(x) - f(x)| < \epsilon$ *for all* $x \in D$.

In application of fuzzy inference systems, the domain in which we operate is almost always closed and bounded and therefore it is compact. For the first and second criteria, it is trivial to find simplified fuzzy inference systems that satisfy them. Now all we need to do is examine the algebraic closure under addition and multiplication. Suppose we have two fuzzy inference systems S and $\tilde{S}$; each has two rules and the output of each system can be expressed as

$$S: \; z = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}, \tag{25}$$

$$\tilde{S}: \; \tilde{z} = \frac{\tilde{w}_1 \tilde{f}_1 + \tilde{w}_2 \tilde{f}_2}{\tilde{w}_1 + \tilde{w}_2}, \tag{26}$$

where f_1, f_2, $\tilde{f}_1$ and $\tilde{f}_2$ are constant output of each rule. Then $az + b\tilde{z}$ and $z\tilde{z}$ can be calculated as follows:

$$\begin{aligned} az + b\tilde{z} &= a\frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} + b\frac{\tilde{w}_1 \tilde{f}_1 + \tilde{w}_2 \tilde{f}_2}{\tilde{w}_1 + \tilde{w}_2} \\ &= \frac{w_1\tilde{w}_1(af_1 + b\tilde{f}_1) + w_1\tilde{w}_2(af_1 + b\tilde{f}_2) + w_2\tilde{w}_1(af_2 + b\tilde{f}_1) + w_2\tilde{w}_2(af_2 + b\tilde{f}_2)}{w_1\tilde{w}_1 + w_1\tilde{w}_2 + w_2\tilde{w}_1 + w_2\tilde{w}_2}, \end{aligned}$$

$$z\tilde{z} = \frac{w_1\tilde{w}_1 f_1\tilde{f}_1 + w_1\tilde{w}_2 f_1\tilde{f}_2 + w_2\tilde{w}_1 f_2\tilde{f}_1 + w_2\tilde{w}_2 f_2\tilde{f}_2}{w_1\tilde{w}_1 + w_1\tilde{w}_2 + w_2\tilde{w}_1 + w_2\tilde{w}_2}, \tag{27}$$

which are of the same form as equation (25) and (26). Apparently the ANFIS architectures that compute $az + b\tilde{z}$ and $z\tilde{z}$ are of the same class of S and $\tilde{S}$ if and only if the class of membership functions is invariant under multiplication. This is loosely true if the class of membership functions is the set of all bell-shaped functions, since the multiplication of two bell-shaped function is almost always still bell-shaped. Another more tightly defined class of membership functions satisfying this criteria, as pointed out by Wang [52, 53], is the scaled Gaussian membership function:

$$\mu_{A_i}(x) = a_i exp[-(\frac{x - c_i}{a_i})^2], \tag{28}$$

Therefore by choosing an appropriate class of membership functions, we can conclude that the ANFIS with simplified fuzzy if-then rules satisfy the four criteria of the Stone-Weierstrass theorem. Consequently, for any given $\epsilon > 0$, and any real-valued function g, there is a fuzzy inference system S such that $|g(\vec{x}) - S(\vec{x})| < \epsilon$ for all $\vec{x}$ in the underlying compact set. Moreover, since the simplified ANFIS is a proper subset of all three types of ANFIS in Figure 2, we can draw the conclusion that all the three types of ANFIS have unlimited approximation power to match any given data set. However, caution has to be taken in accepting this claim since there is no mention about how to construct the ANFIS according to the given data set. That is why learning plays a role in this context.

Another interesting aspect of the simplified ANFIS architecture is its functional equivalence to the radial basis function network (RBFN). This functional equivalence is established when the Gaussian membership function is used in the simplified ANFIS. A detailed treatment can be found in [13]. This functional equivalence provides us with a shortcut for better understanding of ANFIS and RBFN and advances in either literatures apply to both directly. For instance, the hybrid learning rule of ANFIS can be apply to RBFN directly and, vice versa, the approaches used to identify RBFN parameters, such as clustering preprocess [28, 29], orthogonal least squares learning [3], generalization properties [2], sequential adaptation [15], among others [14, 30], are all applicable techniques for ANFIS.

5 Application Examples

This section presents the simulation results of the proposed type-3 ANFIS with both batch (off-line) and pattern (on-line) learning. In the first two examples, ANFIS is used to model highly nonlinear functions and the results are compared with earlier work. In the third example, ANFIS is used as an identifier to identify a nonlinear component on-linely in a discrete control system. Lastly, we use ANFIS to predict a chaotic time series and compare the results with various statistical and connectionist approaches.

A. Practical Considerations

In a conventional fuzzy inference system, the number of rules is decided by an expert who is familiar with the system to be modeled. In our simulation, however, no expert is available and the number of membership functions assigned to each input variable is chosen empirically, i.e., by

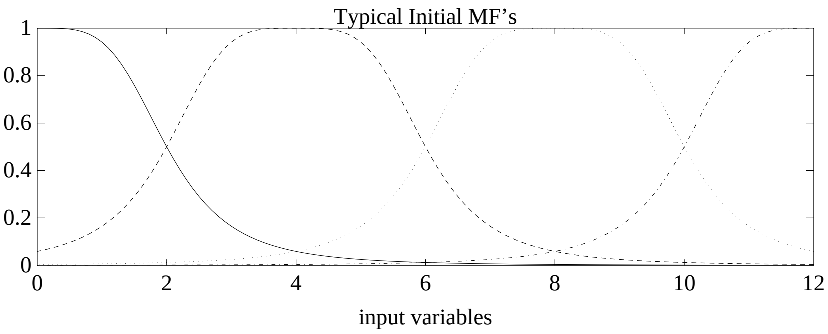


Figure 8: *A typical initial membership function setting in our simulation. (The operating range is assumed to be* $[0, 12]$*.)*

examining the desired input-output data and/or by trial and error. This situation is much the same as that of neural networks; there are no simple ways to determine in advance the minimal number of hidden nodes necessary to achieve a desired performance level.

After the number of membership functions associated with each inputs are fixed, the initial values of premise parameters are set in such a way that the membership functions are equally spaced along the operating range of each input variable. Moreover, they satisfy ϵ-*completeness* [22, 23] with $\epsilon = 0.5$, which means that given a value x of one of the inputs in the operating range, we can always find a linguistic label A such that $\mu_A(x) \geq \epsilon$. In this manner, the fuzzy inference system can provide smooth transition and sufficient overlapping from one linguistic label to another. Though we did not attempt to keep the *epsilon*-completeness during the learning in our simulation, it can be easily achieved by using the constrained gradient method [61]. Figure 8 shows a typical initial membership function setting when the number of membership function is 4 and the operating range is $[0, 12]$.

We mentioned that the step size k in equation (8) may influence the speed of convergence. It is observed that if k 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 k is large, convergence will initially be very fast, but the algorithm will oscillate about the optimum. Based on these observations, we update k according to the following two heuristic rules (see Figure 9):

1. If the error measure undergoes 4 consecutive reductions, increase k by 10%.

2. If the error measure undergoes 2 consecutive combinations of 1 increase and 1 reduction,

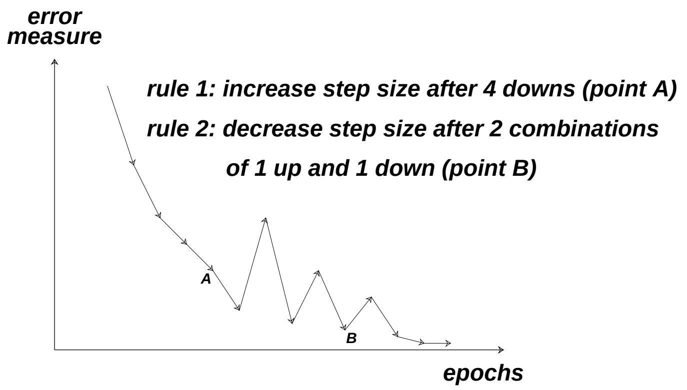


Figure 9: *Two heuristic rules for updating step size k.*

decrease k by 10%.

Though the numbers 10%, 4 and 2 are chosen more or less arbitrarily, the results shown in our simulation appear to be satisfactory. Furthermore, due to this dynamical update strategy, the initial value of k is usually not critical as long as it is not too big.

B. Simulation Results

Example 1: Modeling 2-input Nonlinear Function

In this example, we consider using ANFIS to model a nonlinear *sinc* equation

$$z = sinc(x, y) = \frac{sin(x)}{x} \times \frac{sin(y)}{y}. \tag{29}$$

From the grid points of the range $[-10, 10] \times [-10, 10]$ within the input space of the above equation, 121 training data pairs were obtained first. The ANFIS used here contains 16 rules, with four membership functions being assigned to each input variable and the total number of fitting parameters is 72. (We also tried ANFIS with 4 rules and 9 rules, but obviously they are too simple to describe the highly nonlinear sinc function.) Note that throughout the simulation examples in this section, all the membership functions used are defined as in equation (18) which contains three fitting parameters. Thus the 72 parameters in this example are composed of 24 premise parameters and 48 consequent parameters.

Figure 10 shows the RMSE (root mean squared error) curves for both the 2-18-1 neural network and the ANFIS. Each curve is the average of ten runs: for the neural network, this ten runs were started from 10 different set of initial random weights; for the ANFIS, 10 different initial step

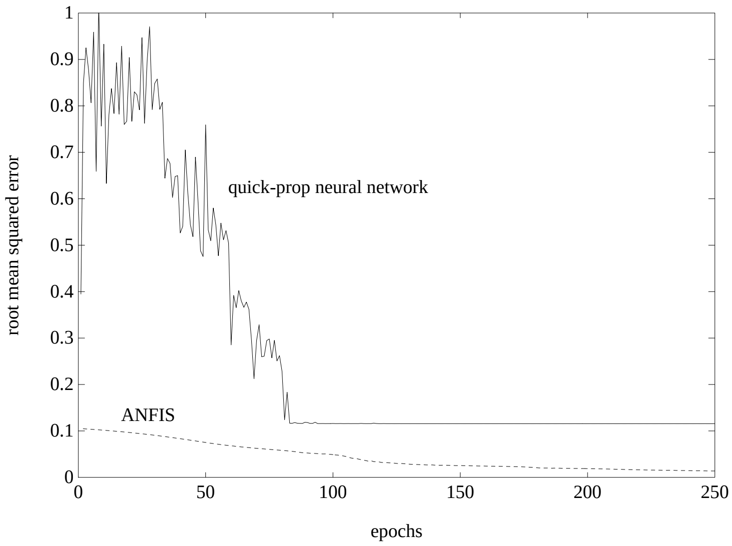


Figure 10: *RMSE curves for the quick-propagation neural networks and the ANFIS.*

size (= $0.01, 0.02, \ldots, 0.10$) were used. The neural network, containing 73 fitting parameters (connection weights and thresholds), was trained with quick propagation [5] which is considered one of the best learning algorithms for connectionist models. Figure 10 demonstrate how ANFIS can effectively model a highly nonlinear surface as compared to neural networks. However, this comparison cannot taken to be universal since we did not attempt an exhaustive search to find the optimal settings for the quick-propagation learning rule of the neural networks.

The training data and other reconstructed surfaces at different epoch numbers are shown in Figure 11. (Since the error measure is always computed after the forward pass is over, the epoch numbers shown in Figure 11 always end with ".5".) Note that the reconstructed surface after 0.5 epoch is due to the identification of consequent parameters only and it already looks similar to the training data surface.

Figure 12 lists the initial and final membership functions. It is interesting to observe that the sharp changes of the training data surface around the origin is accounted for by the moving of the membership functions toward the origin. Theoretically, the final memberships on both x and y should be symmetric with respect to the origin. However, they are not symmetric due to the computer truncation errors and the approximate initial conditions for bootstrapping the calculation of the sequential least squares estimate 14.

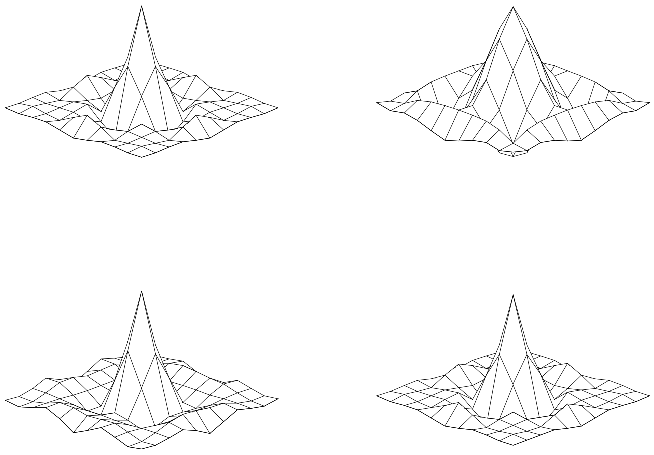


Figure 11: *Training data (upper left) and reconstructed surfaces at 0.5 (upper right), 99.5 (lower left) and 249.5 (lower right) epochs. (Example 1).*

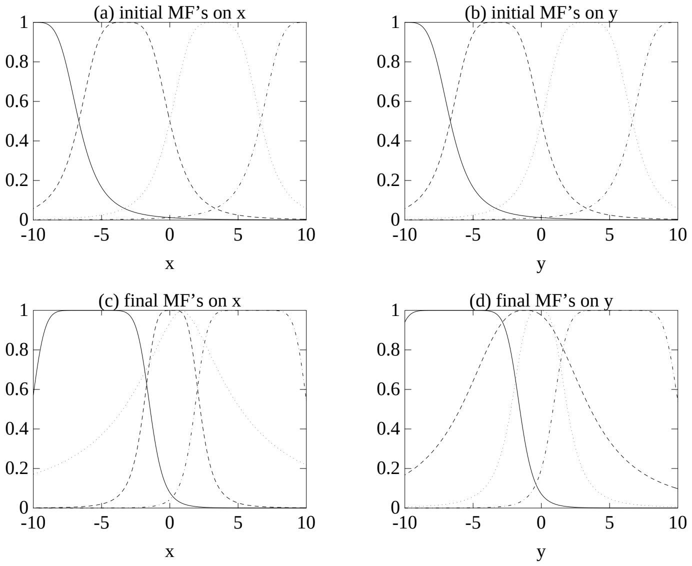


Figure 12: *Initial and final membership functions of example 1.*

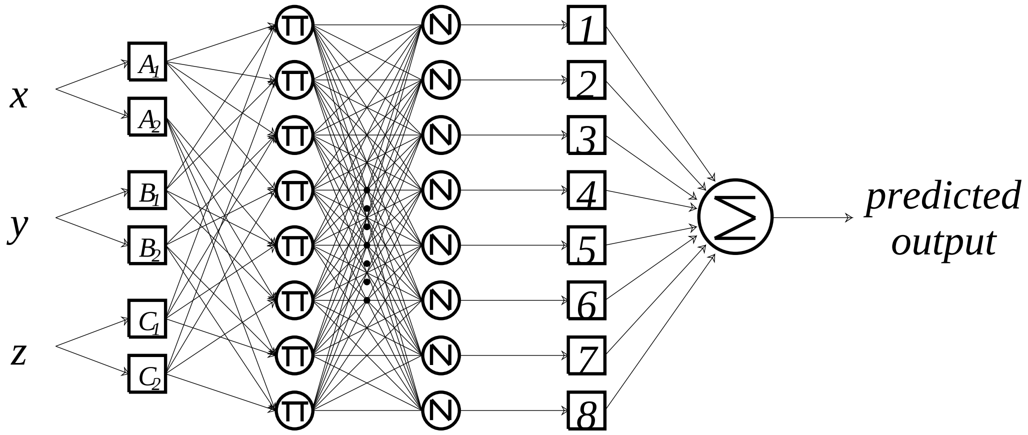


Figure 13: *The ANFIS architecture for example 2. (Note that the connections from inputs to layer 4 are not shown.)*

Example 2: Modeling a 3-input Nonlinear Function

The training data in this example are obtained from

$$output = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2, \tag{30}$$

which was also used by Takagi et al. [48], Sugeno et al. [43] and Kondo [19] to verify their approaches. The ANFIS (see Figure 13) used here contains 8 rules, with 2 membership functions being assigned to each input variable. 216 training data and 125 checking data were sampled uniformly from the input ranges $[1,6]\times[1,6]\times[1,6]$ and $[1.5,5.5]\times[1.5,5.5]\times[1.5,5.5]$, respectively. The training data was used for the training of ANFIS, while the checking data was used for verifying the identified ANFIS only. To allow comparison, we use the same performance index adopted in [43, 19]:

$$APE = average\ percentage\ error = \frac{1}{P}\sum_{i=1}^{P}\frac{|\ T(i) - O(i)\ |}{|\ T(i)\ |} * 100\%. \tag{31}$$

where P is the number of data pairs; $T(i)$ and $O(i)$ are i-th desired output and calculated output, respectively.

Figure 14 illustrates the membership functions before and after training. The training error curves with different initial step sizes (from 0.01 to 0.09) are shown in Figure 15(a), which demonstrates that the initial step size is not too critical on the final performance as long as it is not too big. Figure 15(b) is the training and checking error curves with initial step size equal to 0.1. After 199.5 epochs, the final results are $APE_{trn} = 0.043\%$ and $APE_{chk} = 1.066\%$, which is listed

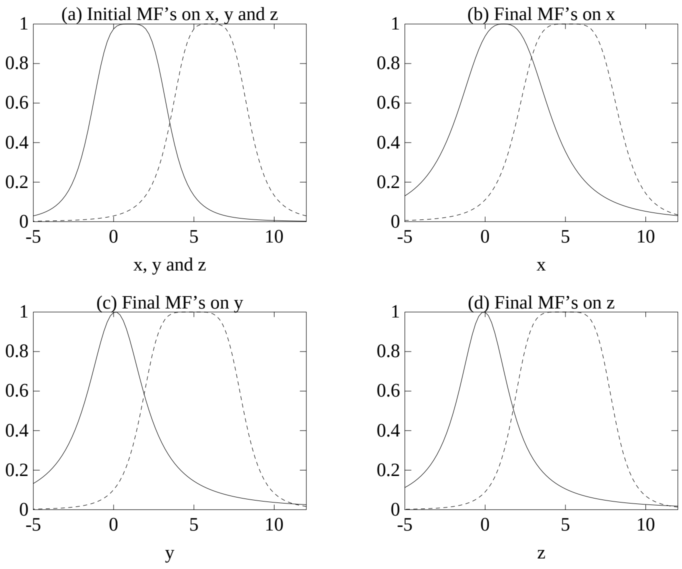


Figure 14: *Example 2, (a) membership functions before learning; (b)(c) (d) membership functions after learning.*

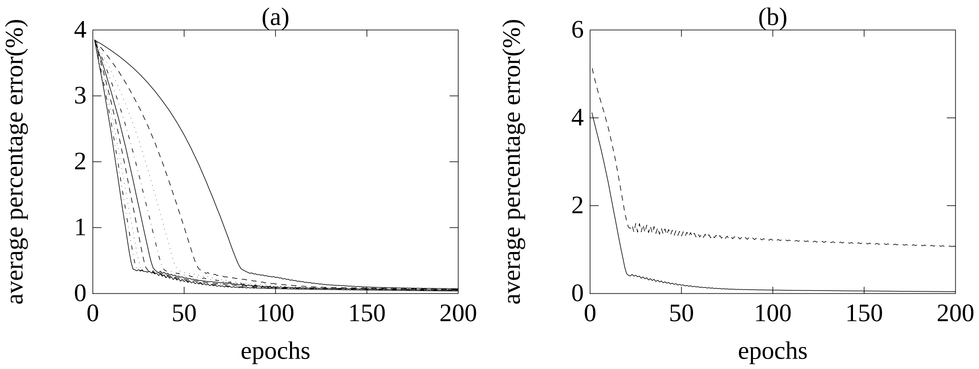


Figure 15: Error curves of example 2: (a) 9 training error curves for 9 initial step size from 0.01 (solid line) to 0.09; (b) training (solid line) and checking (dashed line) error curves with initial step size equal to 0.1

.

Model	APE_{trn} (%)	APE_{chk} (%)	Parameter no.	Training Set Size	Checking Set Size
ANFIS	0.043	1.066	50	216	125
GMDH model [19]	4.7	5.7	-	20	20
Fuzzy model 1 [43]	1.5	2.1	22	20	20
Fuzzy model 2 [43]	0.59	3.4	32	20	20

Table 2: *Example 2: comparisons with earlier work. (The last three rows are from [43].)*

in Table 2 along with other earlier work [43, 19]. Since each simulation cited here was performed under different assumptions and with different training and checking data sets, we cannot make conclusive comments here.

Example 3: On-line Identification in Control Systems

Here we repeat the simulation example 1 of [31] where a 1-20-10-1 neural network is employed to identify a nonlinear component in a control system, except that we use ANFIS to replace the neural network. The plant under consideration is governed by the following difference equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f(u(k)), \tag{32}$$

where y(k) and u(k) are the output and input, respectively, at time index k, and the unknown function $f(\cdot)$ has the form

$$f(u) = 0.3sin(\pi u) + 0.6sin(3\pi u) + 0.5sin(5\pi u). \tag{33}$$

In order to identify the plant, a series-parallel model governed by the difference equation

$$\hat{y}(k+1) = 0.3\hat{y}(k) + 0.6\hat{y}(k-1) + F(u(k)) \tag{34}$$

was used where $F(\cdot)$ is the function implemented by ANFIS and its parameters are updated at each time index. Here the ANFIS has 7 membership functions on its input (thus 7 rules, and 35 fitting parameters) and the pattern (on-line) learning paradigm was adopted with a learning rate $\eta = 0.1$ and a forgetting factor $\lambda = 0.99$. The input to the plant and the model was a sinusoid $u(k) = sin(2\pi k/250)$ and the adaptation started at $k = 1$ and stopped at $k = 250$. As shown in Figure 16, the output of the model follows the output of the plant almost immediately even after the adaptation stopped at $k = 250$ and the $u(k)$ is changed to $0.5sin(2\pi k/250) + 0.5sin(2\pi k/25)$ after $k = 500$. As a comparison, the neural network in [31] fails to follow the plant when the

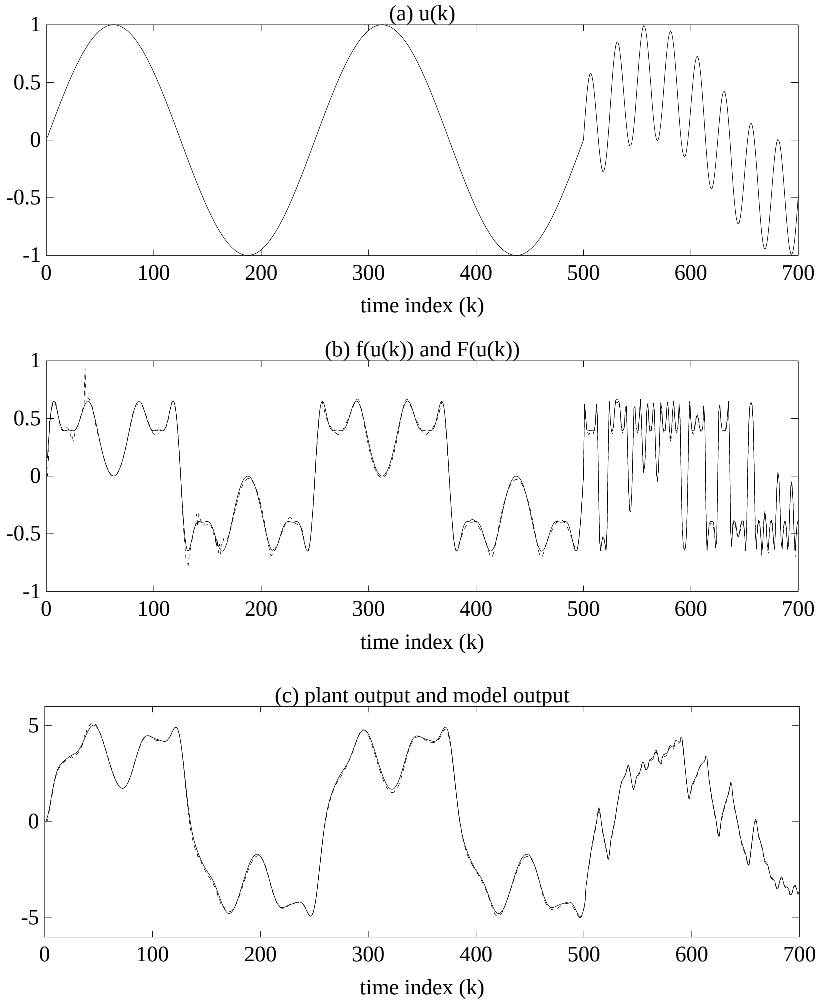


Figure 16: *Example 3: (a) $u(k)$; (a) $f(u(k))$ and $F(u(k))$; (b) plant output and model output.*

Method	Parameter Number	Time Steps of Adaptation
NN	261	50000
ANFIS	35	250

Table 3: *Example 3: comparison with NN identifier [31].)*

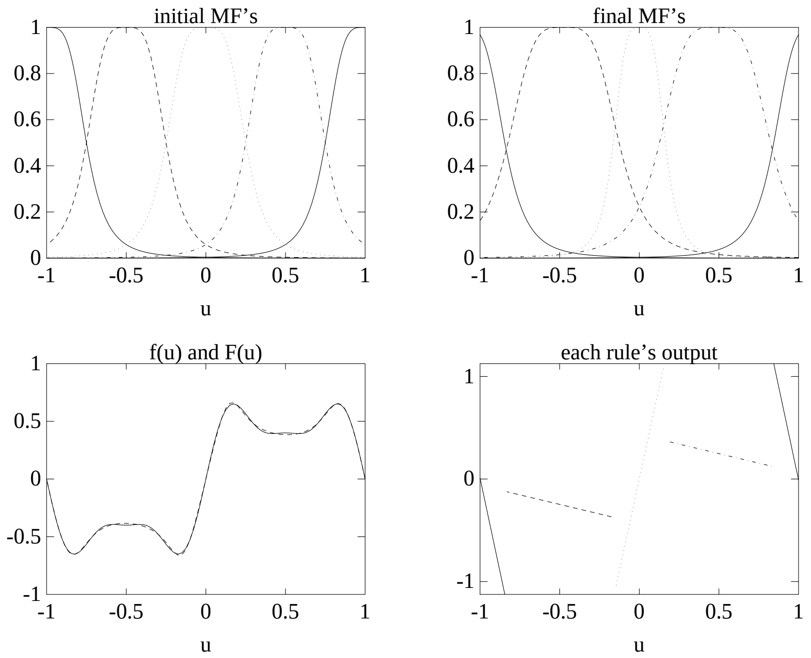


Figure 17: *Example 3: batch learning with 5 MF's.*

adaptation stopped at $k = 500$ and the identification procedure had to continue for $50,000$ time steps using a random input. Table 3 summarizes the comparison.

In the above, the MF number is determined by trial and errors. If the MF number is below 7 then the model output will not follow the plant output satisfactorily after 250 adaptations. But can we decrease the parameter numbers by using batch learning which is supposed to be more effective? Figure 17, 18 and 19 show the results after 49.5 epochs of batch learning when the MF numbers are 5, 4 and 3, respectively. As can be seen, the ANFIS is a good model even when the MF is as small as 3. However, as the MF number is getting smaller, the correlation between $F(u)$ and each rule's output is getting less obvious in the sense that it is harder to sketch $F(u)$ from each rule's consequent part. In other words, when the parameter number is reduced mildly, usually the ANFIS can still do the job but at the cost of sacrificing its semantics in terms of the local-description nature of fuzzy if-then rules; it is less of a structured knowledge representation and more of a black-box model (like neural networks).

Example 4: Predicting Chaotic Dynamics

Example 1, 2 and 3 show that the ANFIS can be used to model highly nonlinear functions

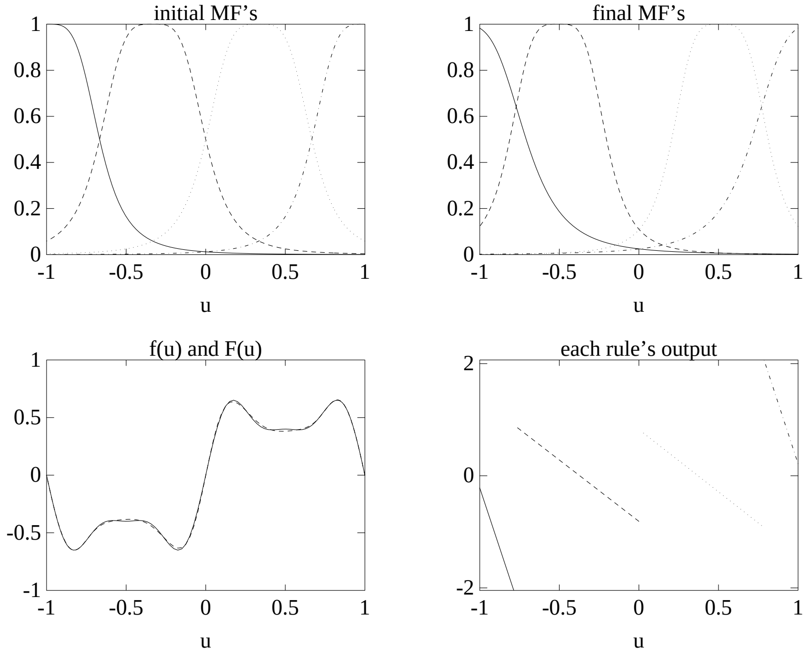


Figure 18: *Example 3: batch learning with 4 MF's.*

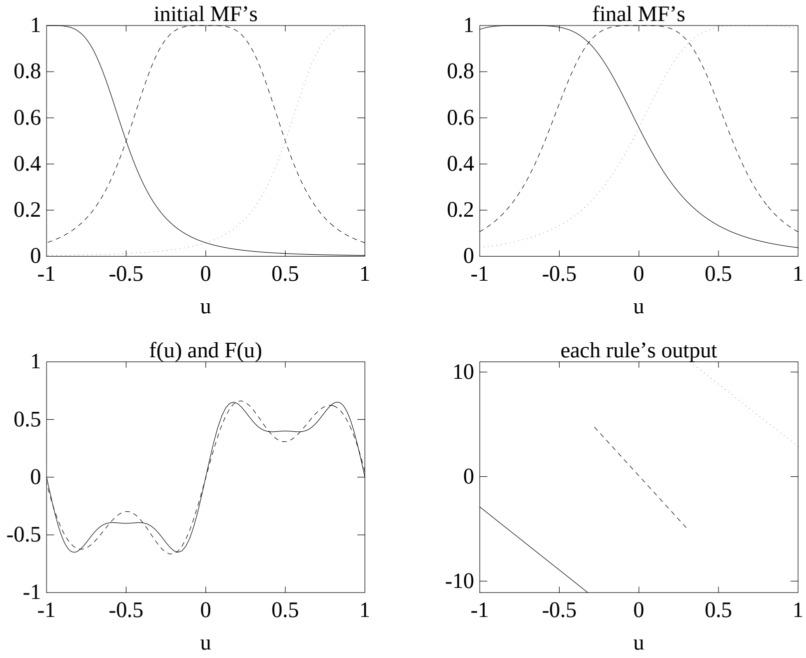


Figure 19: *Example 3: batch learning with 3 MF's.*

effectively. In this example, we will demonstrate how the proposed ANFIS can be employed to predict future values of a chaotic time series. The performance obtained in this example will be compared with the results of a cascade-correlation neural network approach reported in [35] and a simple conventional statistical approach, the auto-regressive (AR) model.

The time series used in our simulation is generated by the chaotic Mackey-Glass differential delay equation [26] defined below:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{35}$$

The prediction of future values of this time series is a benchmark problem which has been considered by a number of connectionist researchers (Lapedes and Farber [21], Moody [29, 27], Jones et al. [14], Crower [35] and Sanger [38]).

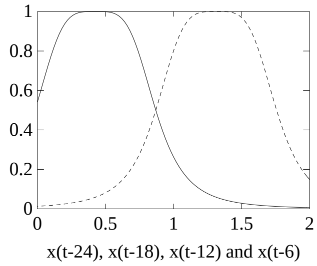
The goal of the task is to use known values of the time series up to the point $x = t$ to predict the value at some point in the future $x = t + P$. The standard method for this type of prediction is to create a mapping from D points of the time series spaced $\triangle$ apart, that is, $(x(t-(D-1)\triangle)$, ..., $x(t-\triangle)$, $x(t))$, to a predicted future value $x(t+P)$. To allow comparison with earlier work (Lapedes and Farber [21], Moody [29, 27], Crower [35]), the values $D = 4$ and $\triangle = P = 6$ were used. All other simulation settings in this example were purposedly arranged to be as close as possible to those reported in [35].

To obtain the time series value at each integer point, we applied the fourth-order Runge-Kutta method to find the numerical solution to equation (35). The time step used in the method is 0.1, initial condition $x(0) = 1.2$, $\tau = 17$, and $x(t)$ is thus derived for $0 \leq t \leq 2000$. (We assume $x(t) = 0$ for $t < 0$ in the integration.) From the Mackey-Glass time series $x(t)$, we extracted 1000 input-output data pairs of the following format:

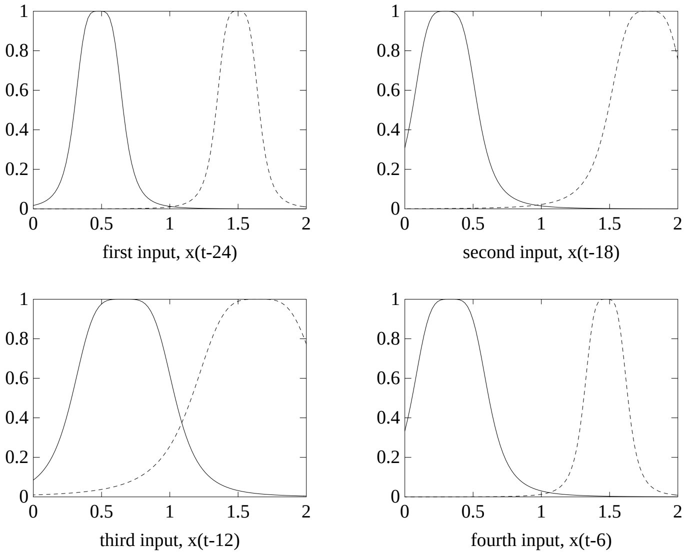
$$[x(t-24), x(t-18), x(t-12), x(t-6); x(t)], \tag{36}$$

where $t = 124$ to 1123. The first 500 pairs (training data set) was used for training the ANFIS while the remaining 500 pairs (checking data set) were used for validating the identified model. The number of membership functions assigned to each input of the ANFIS was arbitrarily set to 2, so the rule number is 16. Figure 20 (a) is the initial membership functions for each input variable. Note that the ANFIS used here contains a total of 104 fitting parameters, of which 24 are premise parameters and 80 are consequent parameters

After 499.5 epochs, we had $RMSE_{trn} = 0.0016$ and $RMSE_{chk} = 0.0015$, which are much



(a) Initial MF's for all four input variables.



(b) Final MF's for each input variable.

Figure 20: *Membership functions of example 3, (a) before learning; (b) after learning.*

better when compared with other approaches explained below. The desired and predicted values for both training data and checking data are essentially the same in Figure 21(a); their differences (Figure 21(b)) can only be seen on a finer scale. Figure 20 (b) is the final membership functions; Figure 22 shows the RMSE curves which indicate most of the learning was done in the first 100 epochs. It is interesting to note the unusual phenomenon that $RMSE_{trn} < RMSE_{chk}$ during the learning process. Considering both the RMSE's are vary small, we conclude that: (1) the ANFIS has captured the essential components of the underlying dynamics; (2) the training data contains the effects of the initial conditions (remember that we set $x(t) = 0$ for $t \leq 0$ in the integration) which might not be easily accounted for by the essential components identified by the ANFIS.

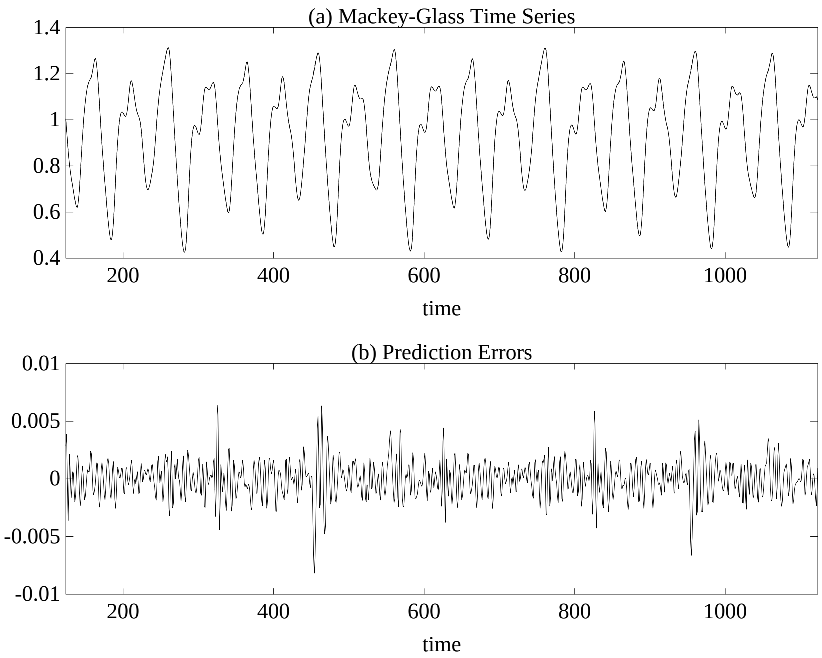


Figure 21: Example 3, (a) Mackey-Glass time series from $t = 124$ to 1123 and six-step ahead prediction (which is indistinguishable from the time series here); (b) prediction error.

As a comparison, we performed the same prediction by using the auto-regressive (AR) model with the same number of parameters:

$$x(t) = a_0 + a_1 x(t-6) + a_2 x(t - 2 * 6) + ... + a_{103} x(t - 103 * 6), \tag{37}$$

where there are 104 fitting parameters a_k, $k = 0$ to 103. From $t = 718$ to 1717, we extracted 1000 data pairs, of which the first 500 were used to identify a_k and the remaining were used for checking. The results obtained through the standard least squares estimate are $RMSE_{trn} = 0.005$ and $RMSE_{chk} = 0.078$ which is much worse than those of ANFIS. Figure 23 shows the predicted values and the prediction errors. Obviously, the over-parameterization of the AR model causes over-fitting in the training data and large errors in the checking data. To search for the best AR model in terms of generalization capability, we tried out different AR models with parameter number being varied from 2 to 104; Figure 24 shows the results where the AR model with the best generalization capability is obtained when the parameter number is 45. Based on this best AR model, we repeat the generalization test and Figure 25 shows the results where there is no over-fitting at the price of larger training errors.

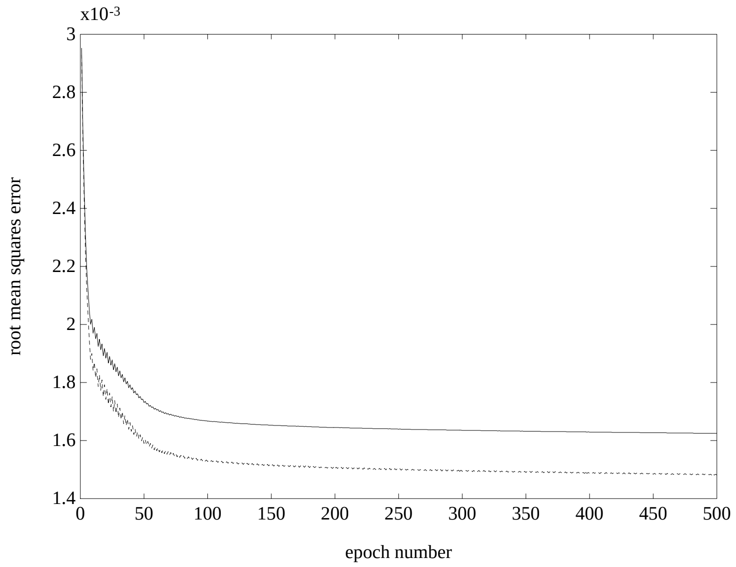


Figure 22: *Training and checking RMSE curves for ANFIS modeling.*

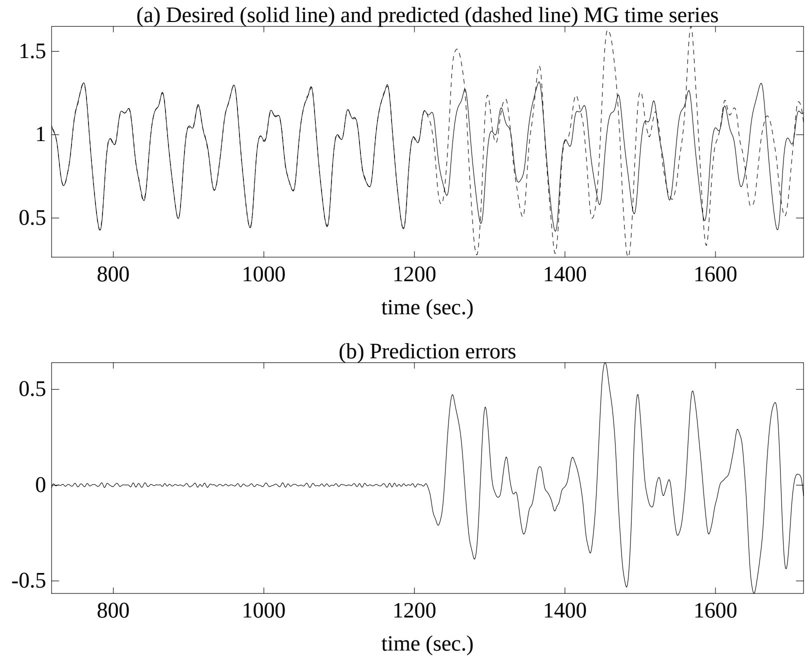


Figure 23: (a) Mackey-Glass time series (solid line) from $t = 718$ to 1717 and six-step ahead prediction (dashed line) by AR model with parameter = 104; (b) prediction errors.

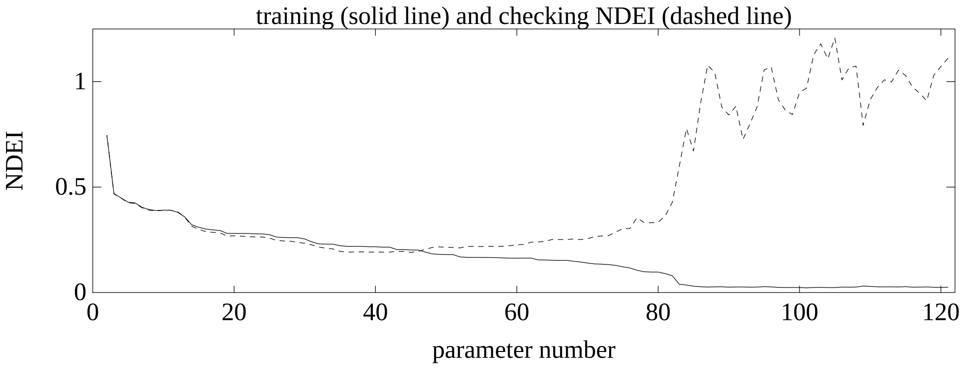


Figure 24: *Training (solid line) and checking (dashed line) errors of AR models with different parameter numbers.*

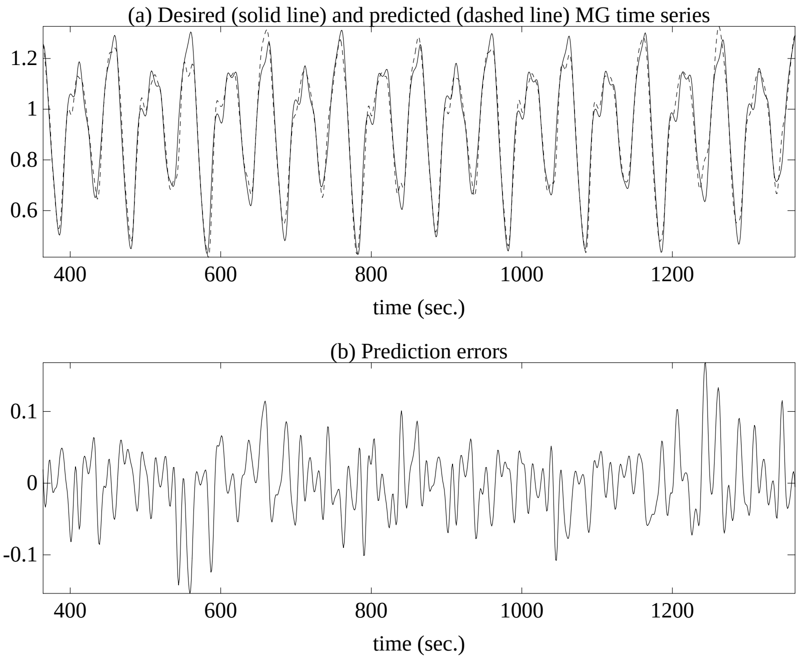


Figure 25: Example 3, (a) Mackey-Glass time series (solid line) from $t = 364$ to 1363 and six-step ahead prediction (dashed line) by the best AR model (parameter number = 45); (b) prediction errors.

It goes with saying that the nonlinear ANFIS outperforms the linear AR model. However, it should be noted that the identification of the AR model took only a few seconds, while the ANFIS simulation took about 2.5 hours on a HP Apollo 700 Series workstation. (We did not pay special attention on the optimization of the codes, though.)

Table 4 lists other methods' generalization capabilities which are measured by using each method to predict 500 points immediately following the training set. Here the *non-dimensional error index* (NDEI) [21, 35] is defined as the root mean square error divided by the standard deviation of the target series. (Note that the *average relative variance* used in [55, 56] is equal to the square of NDEI.) The remarkable generalization capability of the ANFIS, we believe, comes from the following facts:

1. The ANFIS can achieve a highly nonlinear mapping as shown in Example 1, 2 and 3, therefore it is superior to common linear methods in reproducing nonlinear time series.
2. The ANFIS used here has 104 adjustable parameters, much less than those of the cascade-correlation NN (693, the median size) and back-prop NN (about 540) listed in Table 4.
3. Though without *a priori* knowledge, the initial parameter settings of ANFIS are intuitively reasonable and it leads to fast learning that captures the underlying dynamics.

Method	Training Cases	Non-Dimensional Error Index
ANFIS	500	0.007
AR Model	500	0.19
Cascaded-Correlation NN	500	0.06
Back-Prop NN	500	0.02
6th-order Polynomial	500	0.04
Linear Predictive Method	2000	0.55

Table 4: *Generalization result comparisons for $P = 6$. (The last four rows are from [35].)*

Table 5 lists the results of the more challenging generalization test when $P = 84$ (the first six rows) and $P = 85$ (the last four rows). The results of the first six rows were obtained by iterating the prediction of $P = 6$ till $P = 84$. ANFIS still outperforms these statistical and connectionist approaches unless a substantially large amount of training data (i.e., the last row of Table 5) were used instead. Figure 26 illustrates the generalization test for the ANFIS where the first 500 points were used for the desired outputs while the last 500 are the predicted outputs for $P = 84$.

Method	Training Cases	Non-Dimensional Error Index
ANFIS	500	0.036
AR Model	500	0.39
Cascaded-Correlation NN	500	0.32
Back-Prop NN	500	0.05
6th-order Polynomial	500	0.85
Linear Predictive Method	2000	0.60
LRF	500	0.10-0.25
LRF	10000	0.025-0.05
MRH	500	0.05
MRH	10000	0.02

Table 5: *Generalization result comparisons for $P = 84$ (the first six rows) and 85 (the last four rows). Results for the first six methods are generated by iterating the solution at $P = 6$. Results for localized receptive fields (LRF) are multi-resolution hierarchies (MRH) are for networks trained for $P = 85$. (The last eight rows are from [35].)*

6 Concluding Remarks

A. Summary and Extensions of Current work

We have described the architecture of adaptive-network-based fuzzy inference systems (ANFIS) with type-1 and type-3 reasoning mechanisms. By employing a hybrid learning procedure, the proposed architecture can refine fuzzy if-then rules obtained from human experts to describe the input-output behavior of a complex system. However, if human expertise is not available, we can still set up intuitively reasonable initial membership functions and start the learning process to generate a set of fuzzy if-then rules to approximate a desired data set, as shown in the simulation examples of nonlinear function modeling and chaotic time series prediction.

Due to the high flexibility of adaptive networks, the ANFIS can have a number of variants from what we have proposed here. For instance, the membership functions can be changed to *L-R representation* [4] which could be asymmetric, Furthermore, we can replace Π nodes in layer 2 with the parameterized *T-norm* [4] and let the learning rule to decide the best T-norm operator for a specific application. By employing the adaptive network as a common framework, we have also proposed other adaptive fuzzy models tailored for data classification [45, 46] and feature extraction [47] purposes.

Another important issue in the training of ANFIS is how to preserve the human-plausible features such as bell-shaped membership functions, ϵ-completeness [22, 23] or sufficient overlapping

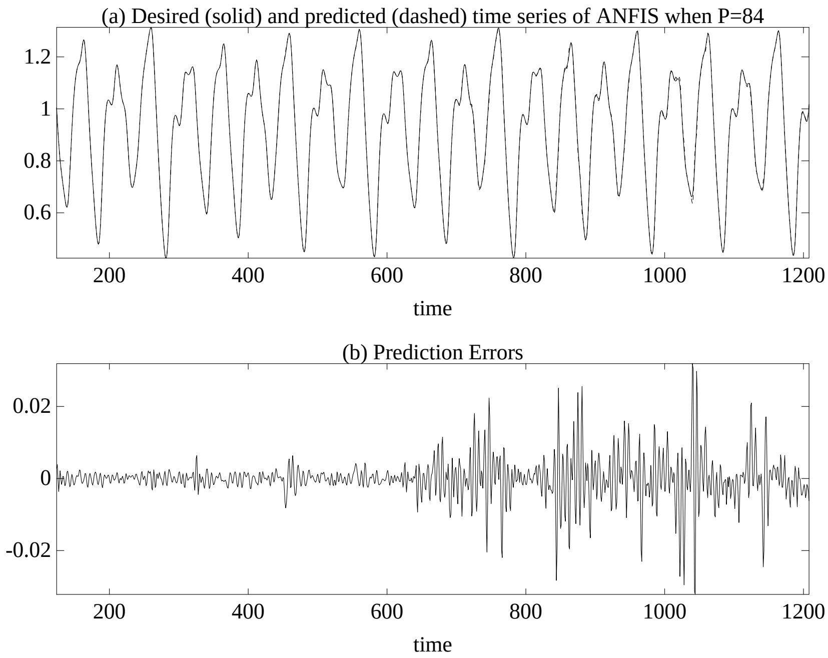


Figure 26: *Generalization test of ANFIS for* $P = 84$.

between adjacent membership functions, minimal uncertainty, etc. Though we did not pursue along this direction in this paper, mostly it can be achieved by maintaining certain constraints and/or modifying the original error measure as explained below.

- To keep bell-shaped membership functions, we need the membership functions to be bell-shaped regardless of the parameter values. In particular, equation (18) and equation (19) become up-side-down bell-shaped if $b_i < 0$; one easy way to correct this is to replace b_i with b_i^2 in both equations.

- The ϵ-completeness can be maintained by the constrained gradient descent [61]. For instance, suppose that $\epsilon = 0.5$ and the adjacent membership functions are of the form of equation (18) with parameter sets $\{a_i, b_i, c_i\}$ and $\{a_{i+1}, b_{i+1}, c_{i+1}\}$. Then the ϵ-completeness is satisfied if $c_i + a_i = c_{i+1} - a_{i+1}$ and this can be ensured throughout the training if the constrained gradient descent is employed.

- Minimal uncertainty refers to the situation that within most region of the input space, there should be a dominant fuzzy if-then rule to account for the final output, instead of multiple rules with similar firing strengths. This minimizes the uncertainty and make the rule set more informative. One way to do this is to use a modified error measure

$$E' = E + \sum_{i=1}^{P} [-\overline{w}_i \times ln(\overline{w}_i)], \tag{38}$$

where E is the original squared error ;P is the size of training data set; $\overline{w}_i$ is the normalized firing strength of the i-th rule (see equation (21)) and $\sum_{i=1}^{P}[-\overline{w}_i \times ln(\overline{w}_i)]$ is the *information entropy*. Since this modified error measure is not based on data fitting along, the ANFIS thus trained can also have a potentially better generalization capability. The improvement of generalization by using an error measure based both data fitting and weight elimination has been reported in the neural network literature [55, 56].

It is conjectured that this modified error measure could even decrease the possibility of over-fitting since the error measure is not based on data fitting alone. The effects of a similar error measure on neural networks was reported in [55].

In this paper, we assume the structure of the ANFIS is fixed and the *parameter identification* is solved through the hybrid learning rule. However, to make the whole approach more complete, the *structure identification* [43, 44] (which concerns with the selection of an appropriate input-space partition style and the number of membership functions on each input, etc.) is equally important to the successful applications of ANFIS. Effective partition of the input space can decrease the rule number and thus increase the speed in both learning and application phases. Advances on neural networks' structure identification [6, 24] can shed some lights on this aspect.

B. Applications to Automatic Control and Signal Processing

Fuzzy control is by far the most successful applications of the fuzzy set theory and fuzzy inference systems. Due to the adaptive capability of ANFIS, its applications to adaptive control and learning control are immediate. Most of all, it can replace almost any neural networks in control systems to serve the same purposes. For instance, Narendra's pioneering work of using neural networks in adaptive control [31] can be all achieved similarly by ANFIS. Moreover, four of the generic designs (i.e., *supervised control*, *direct inverse control*, *neural adaptive control* and *back-propagation of utility*) of neural networks in control, as proposed by Werbos [58, 9], are also directly applicable schemes for ANFIS. Particularly we have employed a similar method of the *back-propagation through time* [32] or *unfolding in time* to achieve a self-learning fuzzy controller with four rules that can balance an inverted pendulum in an near-optimal manner [12]. It is expected that the advances of neural network techniques in control can promote those of ANFIS as well, and vice versa.

The active role of neural networks in signal processing [60, 20] also suggests similar applications of ANFIS. The nonlinearity and structured knowledge representation of ANFIS are the primary advantages over classical linear approaches in adaptive filtering [8] and adaptive signal processing [59], such as identification, inverse modeling, predictive coding, adaptive channel equalization, adaptive interference (noise or echo) canceling, etc.

Acknowledgement

The guidance and help of Professor Lotfi A. Zadeh and other members of the "fuzzy group" at UC Berkeley is gratefully acknowledged.

References

[1] K. J. Aström and B. Wittenmark. *Computer Controller Systems: Theory and Design.* Prentice-Hall, Inc., 1984.

[2] S. M. Botros and C. G. Atkeson. Generalization properties of radial basis functions. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems III*, pages 707–713. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[3] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. on Neural Networks*, 2(2):302–309, March 1991.

[4] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic press, New York, 1980.

[5] S. E. Fahlman. Faster-learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1988 Connectionist Models Summer School*, pages 38–51, Carnegic Mellon University, 1988.

[6] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, G. Hinton, and T. Sejnowski, editors, *Advances in Neural Information Processing Systems II.* Morgan Kaufmann, 1990.

[7] G. C. Goodwin and K. S. Sin. *Adaptive filtering prediction and control.* Prentice-Hall, ENglewood Cliffs, N.J., 1984.

[8] S. S. Haykin. *Adaptive filter theory.* Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

[9] W. T. Miller III, R. S. Sutton, and P. J. Werbos, editors. *Neural Networks for Control.* Massachusetts Institute of Technology, 1990.

[10] J.-S. Roger Jang. Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In *Proc. of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 762–767, July 1991.

[11] J.-S. Roger Jang. Rule extraction using generalized neural networks. In *Proc. of the 4th IFSA World Congress*, July 1991.

[12] J.-S. Roger Jang. Self-learning fuzzy controller based on temporal back-propagation. *IEEE Trans. on Neural Networks*, September 1992.

[13] J.-S. Roger Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Trans. on Neural Networks*, 1992. (Forthcoming).

[14] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, and P. S. Lewis. Function approximation and time series prediction with neural networks. In *Proc. of IEEE International Joint Conference on Neural Networks*, pages I–649–665, 1990.

[15] V. Kadirkamanathan, M. Niranjan, and F. Fallside. Sequential adaptation of radial basis function neural networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems III*, pages 721–727. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[16] A. Kandel. *Fuzzy expert systems.* Addison-Wesley, 1988.

[17] A. Kandel. *Fuzzy expert systems.* CRC Press, Boca Raton, FL, 1992.

[18] L. V. Kantorovich and G. P. Akilov. *Functional analysis.* Pergamon, Oxford, 2 edition, 1982.

[19] T. Kondo. Revised GMDH algorithm estimating degree of the complete polynomial. *Tran. of the Society of Instrument and Controrl Engineers*, 22(9):928–934, 1986. (Japanese).

[20] B. Kosko. *Neural networks for signal processing.* Prentice Hall, Englewood Ciffs, NJ, 1991.

[21] A. S. Lapedes and R. Farber. Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, 1987.

[22] C.-C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 1. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):404–418, 1990.

[23] C.-C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 2. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):419–435, 1990.

[24] T.-C. Lee. *Structure level adaptation for artificial neural networks.* Kluwer Academic Publishers, 1991.

[25] L. Ljung. *System identification: theory for the user.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[26] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, July 1977.

[27] J. Moody. Fast learning in multi-resolution hierarchies. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, chapter 1, pages 29–39. Morgan Kaufmann, San Mateo, CA, 1989.

[28] J. Moody and C. Darken. Learning with localized receptive fields. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1988 Connectionist Models Summer School.* Carnegie Mellon University, Morgan Kaufmann Publishers, 1988.

[29] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[30] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5(4):595–603, 1992.

[31] K. S. Narendra and K. Parthsarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–27, 1990.

[32] D. H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, pages 18–23, April 1990.

[33] D. B. Parker. Optimal algorithms for adaptive networks: Second order back propagation, second order direct propagation, and second order Hebbian learning. In *Proc. of IEEE International Conference on Neural Networks*, pages 593–600, 1987.

[34] W. Pedrycz. *Fuzzy control and fuzzy systems.* Wiley, New York, 1989.

[35] III R. S. Crowder ' Predicting the Mackey-Glass timeseries with cascade-correlation learning. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1990 Connectionist Models Summer School*, pages 117–123, Carnegic Mellon University, 1990.

[36] H. L. Royden. *Real analysis.* Macmillan, New York, 2 edition, 1968.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volum 1*, chapter 8, pages 318–362. The MIT Press, 1986.

[38] T. D. Sanger. A tree-structured adaptive network for function approximate in high-dimensional spaces. *IEEE Trans. on Neural Networks*, 2(2):285–293, March 1991.

[39] S. Shar and F. Palmieri. MEKA-a fast, local algorithm for training feedforward neural networks. In *Proc. of International Joint Conference on Neural Networks*, pages III 41–46, 1990.

[40] S. Singhal and L. Wu. Training multilayer perceptrons with the extended kalman algorithm. In David S. Touretzky, editor, *Advances in neural information processing systems I*, pages 133–140. Morgan Kaufmann Publishers, 1989.

[41] P. Strobach. *Linear prediction theory: a mathematical basis for adaptive systems.* Springer-Verlag, 1990.

[42] M. Sugeno, editor. *Industrial applications of fuzzy control.* Elsevier Science Pub. Co., 1985.

[43] M. Sugeno and G. T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28:15–33, 1988.

[44] C.-T. Sun. Rulebase structure identificaiton in an adaptive network based fuzzy inference system. *IEEE Trans. on Neural Networks*, September 1992. (Forthcoming).

[45] C.-T Sun and J.-S. Roger Jang. Adaptive network based fuzzy classification. In *Proc. of the Japan-U.S.A. Symposium on Flexible Automation*, July 1992.

[46] C.-T. Sun and J.-S. Roger Jang. Fuzzy classification based on adaptive networks and genetic algorithms. *IEEE Trans. on Neural Networks*, 1992. (Submitted).

[47] C.-T. Sun and J.-S. Roger Jang. Plasma modeling: an application of fuzzy filters and neural networks. In *to be submitted to World Congress on Neural Networks*, 1993.

[48] H. Takagi and I. Hayashi. NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 5(3):191–212, 1991.

[49] T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. *Proc. of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 55–60, July 1983.

[50] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man, and Cybernetics*, 15:116–132, 1985.

[51] Y. Tsukamoto. An approach to fuzzy reasoning method. In Madan M. Gupta, Rammohan K. Ragade, and Ronald R. Yager, editors, *Advances in Fuzzy Set Theory and Applications*, pages 137–149. North-Holland, Amsterdam, 1979.

[52] L.-X. Wang. Fuzzy systems are universal approximators. In *Proc. of the IEEE International Conference on Fuzzy Systems*, San Diego, March 1992.

[53] L.-X. Wang and J. M. Mendel. Fuzzy basis function, universal approximation, and orthogonal least squares learning. *IEEE Trans. on Neural Networks*, September 1992. (Forthcoming).

[54] R. L. Watrous. Learning algorithms for connectionist network: applied gradient methods of nonlinear optimization. In *Proc. of IEEE International Conference on Neural Networks*, pages 619–627, 1991.

[55] A. A. Weigend, D. E. Rumelhart, and B. A. Huberman. Back-propagation, weight-elimination and time series prediction. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1990 Connectionist Models Summer School*, pages 105–116, Carnegic Mellon University, 1990.

[56] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination with application to forecasting. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems III*, pages 875–882. Morgan Kaufmann, San Mateo, CA, 1991.

[57] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University, 1974.

[58] P. J. Werbos. An overview of neural networks for control. *IEEE Control Systems Magazine*, 11(1):40–41, January 1991.

[59] B. Widrow and D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Englewood Cliffs, N.J., 1985.

[60] B. Widrow and R. Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer*, pages 25–39, March 1988.

[61] D. A. Wismer and R. Chattergy. *Introduction to nonlinear optimization: a problem solving approach*, chapter 6, pages 139–162. North-Holland Publishing Company, 1978.

[62] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[63] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on Systems, Man, and Cybernetics*, 3(1):28–44, January 1973.