Input Selection for ANFIS Learning

Jyh-Shing Roger Jang (jang@cs.nthu.edu.tw)
Department of Computer Science, National Tsing Hua University Hsinchu, Taiwan

Abstract

We present a quick and straightfoward way of input selection for neuro-fuzzy modeling using ANFIS. The method is tested on two real-world problems: the nonlinear regression problem of automobile MPG (miles per gallon) prediction, and the nonlinear system identification using the Box and Jenkins gas furnace data [1].

1. Introduction

For a real-world modeling problem, it is not uncommon to have tens of potential inputs to the model under construction. An excessive number of inputs not only impair the transparency of the underlying model, but also increase the complexity of computation necessary for building the model. Therefore, it is necessary to do input selection that finds the priority of each candidate inputs and uses them accordingly. Specifically, the purposes of input selection include:

- Remove noise/irrelevant inputs.
- Remove inputs that depends on other inputs.
- Make the underlying model more concise and transparent.
- Reduce the time for model construction.

In this paper, we present a quick and staightforward way of input selection for neuro-fuzzy modeling using ANFIS (Adaptive Neuro-Fuzzy Inference Systems) [2, 3], a previously proposed neuro-fuzzy network structure. The input selection method is tested on two real-world problems: the nonlinear regression problem of automobile MPG (miles per gallon) prediction, and the nonlinear system identification using the Box and Jenkins gas furnace data [1].

This paper is organized into six sections. In the next section, the basics of ANFIS are introduced. Section 3 explains how to proceed input selection for ANFIS modeling. Application to the problems of automobile MPG prediction and gas furnace identification are demonstrated in section 4 and 5, respectively. Section 5 gives concluding remarks.

2. ANFIS

This section introduces the basics of ANFIS network architecture and its hybrid learning rule. A detailed coverage of ANFIS can be found in [2, 3, 6].

The **Sugeno fuzzy model** was proposed by Takagi, Sugeno, and Kang [16, 15] in an effort to formalize a systematic approach to generating fuzzy rules from an input-output data set. A typical fuzzy rule in a Sugeno fuzzy model has the format

$$\textit{If } x \textit{ is } A \textit{ and } y \textit{ is } B \textit{ then } z = f(x,y),$$

where A and B are fuzzy sets in the antecedent; $z = f(x, y)$ is a crisp function in the consequent. Usually $f(x, y)$ is a polynomial in the input variables x and y, but it can be any other functions that can appropriately describe the output of the system within the fuzzy region specified by the antecedent of the rule. When $f(x, y)$ is a first-order polynomial, we have the **first-order** Sugeno fuzzy model, which was originally proposed in [16, 15]. When f is a constant, we then have the **zero-order** Sugeno fuzzy model, which can be viewed either as a special case of the Mamdani fuzzy inference system [9] where each rule's consequent is specified by a fuzzy singleton, or a special case of Tsukamoto's fuzzy model [17] where each rule's consequent is specified by a membership function of a step function centered at the constant. Moreover, a zero-order Sugeno fuzzy model is functionally equivalent to a radial basis function network under certain minor constraints [5].

Consider a first-order Sugeno fuzzy inference system which contains two rules:

Rule 1: If X is A_1 and Y is B_1, then
$f_1 = p_1x + q_1y + r_1$,
Rule 2: If X is A_2 and Y is B_2, then
$f_2 = p_2x + q_2y + r_2$.

0-7803-3645-3/96 $5.00©1996 IEEE

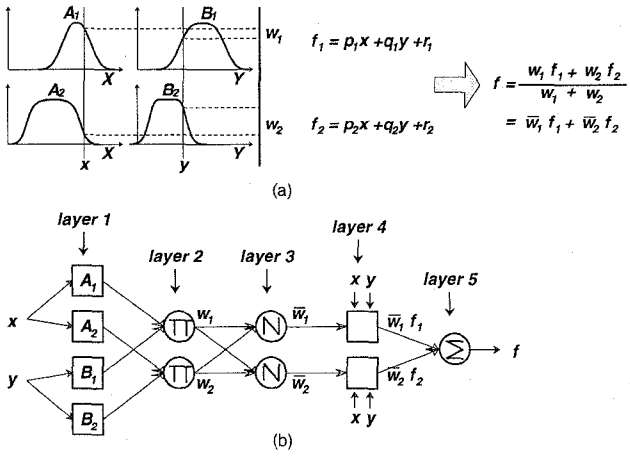

Figure 1. *(a) First-order Sugeno fuzzy model; (b) corresponding ANFIS architecture.*

Figure 1 (a) illustrates graphically the fuzzy reasoning mechanism to derive an output f from a given input vector $[x, y]$. The **firing strengths** w_1 and w_2 are usually obtained as the product of the membership grades in the premise part, and the output f is the weighted average of each rule's output.

To facilitate the learning (or adaptation) of the Sugeno fuzzy model, it is convenient to put the fuzzy model into the framework of adaptive networks that can compute gradient vectors systematically. The resultant network architecture, called **ANFIS** (Adaptive Neuro-Fuzzy Inference System), is shown in Figure 1 (b), where node within the same layer perform functions of the same type, as detailed below. (Note that O_i^j denotes the output of the i-th node in j-th layer.)

Layer 1 Each node in this layer generates a membership grades of a linguistic label. For instance, the node function of the i-th node may be a generalized bell membership function:

$$O_i^1 = \mu_{A_i}(x) = \frac{1}{1 + \left|\frac{x - c_i}{a_i}\right|^{2b_i}}, \qquad (1)$$

where x is the input to node i; A_i is the linguistic label (*small* , *large*, etc.) associated with this node; and $\{a_i,\ b_i,\ c_i\}$ is the parameter set that changes the shapes of the membership function. Parameters in this layer are referred to as the **premise parameters**.

Layer 2 Each node in this layer calculates the firing strength of a rule via multiplication:

$$O_i^2 = w_i = \mu_{A_i}(x)\mu_{B_i}(y),\ i = 1, 2. \qquad (2)$$

Layer 3 Node i in this layer calculates the ratio of the i-th rule's firing strength to the total of all firing strengths:

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2},\ i = 1, 2. \qquad (3)$$

Layer 4 Node i in this layer compute the contribution of i-th rule toward the overall output, with the following node function:

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

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 are referred to as the **consequent parameters**.

Layer 5 The single node in this layer computes the overall output as the summation of contribution from each rule:

$$O_1^5 = \textit{overall output} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \qquad (5)$$

The constructed adaptive network in Figure 1(b) is functionally equivalent to a fuzzy inference system in Figure 1(a). The basic learning rule of ANFIS is the backpropagation gradient descent [18], which calculates error signals (the derivative of the squared error with respect to each node's output) recursively from the output layer backward to the input nodes. This learning rule is exactly the same as the backpropagation learning rule used in the common feedforward neural networks [13].

From the ANFIS architecture in Figure 1, it is observed that given the values of premise parameters, the overall output f can be expressed as a linear combinations of the consequent parameters:

$$\begin{aligned} f &= \overline{w}_1 f_1 + \overline{w}_2 f_2 \\ &= (\overline{w}_1 x)p_1 + (\overline{w}_1 y)q_1 + (\overline{w}_1)r_1 \\ &\quad + (\overline{w}_2 x)p_2 + (\overline{w}_2 y)q_2 + (\overline{w}_2)r_2. \end{aligned} \qquad (6)$$

Based on this observation, we have proposed a hybrid learning algorithm [2, 3] which combines the gradient descent and the least-squares method for an effective search of optimal parameters; both on-line and off-line learning paradigms were developed and reported in [3]. Moreover, other advanced techniques in nonlinear regression and optimization, such as the Gauss-Newton method, the Levenberg-Marquardt method [8, 10], and the extended Kalman filter algorithm [14, 12] can also be applied here directly.

The original ANFIS C codes and several examples can be retrieved via anonymous ftp in `user/ai/areas/fuzzy/systems/anfis` at

`ftp.cs.cmu.edu` (CMU Artificial Intelligence Repository). For MATLAB users, ANFIS is also available in the Fuzzy Logic Toolbox used with MATLAB [4].

Following the concept of ANFIS, we have also proposed the CANFIS (Coactive ANFIS) architecture [11, 7] that has multiple outputs and nonlinear output equations. Details of ANFIS/CANFIS and their applications can be found in [7].

3. Input Selection

As mentioned earlier, a real-world modeling problem usually involves tens (or even hundreds) of potential inputs to the model under construction. Therefore we need to have a heuristic way to quickly determine the priorities of these potential inputs and use them accordingly. In this section, we propose a quick and straightfoward way to do input selection for ANFIS modeling.

As described in the previous section, ANFIS is a network structure that facilitates systematical computation of gradient vectors, the derivative of the output error with respective to each modifiable parameters. In particular, ANFIS employs an efficient hybrid learning method that combines gradient descent and the least-squares method. The least-squares method is, actually, the major driving force that leads to fast training, while the gradient descent serves to slowly change the underlying membership functions that generates the basis functions for the least-squares method. As a result, ANFIS can usually generate satisfactory results right after the first epoch of training, that is, only after the first application of the least-squares method. Since the least-squares method is computationally efficient, we can construct ANFIS models for various combinations of inputs, train them with a single application of the least-squares method, and then choose the one with the best performance and proceed for further training.

The proposed input selection method is based on the assumption that the ANFIS model with the smallest RMSE (root mean squared error) after one epoch of training, has a greater potential of achieving a lower RMSE when given more epochs of training. This assumption is not absolutely true, but it is heuristically reasonable.

For instance, if we have a modeling problem with 10 candidate inputs and we want to find the most influential 3 inputs as the inputs to ANFIS, we can construct $C_3^{10} = 120$ ANFIS models (each with different combination of 3 inputs), and train them with a single pass of the least-squares method. The ANFIS model with the smallest training error is then selected for further training using the hybrid learning rule to tune the membership functions as well. Note that one-epoch training of 120 ANFIS models in fact involves less computation than 120-epoch training of a single ANFIS model, therefore the input selection procedure is not really as computation intensive as it looks.

Another reason for input selection is to facilitate the input-space grid partitioning for ANFIS; this is further explained in Section 4, where ANFIS is used for automobile MPG (miles per gallon) prediction.

For certain types of problems, the candidate inputs are divided into groups and, due to physical properties of the target system, one or several members of each group has to be in the set of final inputs to the model under consideration. These physical properties allow us to build less potential ANFIS models initially. One such example is the nonlinear system identification problem discussed in Section 5.

4. Automobile MPG Prediction

This section describes the use of the proposed input selection method for ANFIS modeling, with application to nonlinear regression. We shall use automobile MPG (miles per gallon) prediction as a case study, in which an automobile's fuel consumption in terms of MPG is predicted by ANFIS based on several given characteristics, such as number of cylinders, weight, model years, and so on.

The automobile MPG prediction problem is a typical nonlinear regression problem where several attributes (input variables) are used to predict another continuous attribute (output variable). In this case, the six input attributes includes profile information about the automobiles:

No. of cylinders:	multi-valued discrete
Displacement:	continuous
Horsepower:	continuous
Weight:	continuous
Acceleration:	continuous
Model year:	multi-valued discrete

The attribute to be predicted in terms of the above six (6) input attributes is the fuel consumption in MPG. Table 1 is a list of seven instances selected at random from the data set. After removing instances with missing values, the data set was reduced to 392 entries. Our task was then to use this data set and ANFIS to construct a fuzzy inference system that could best predict the MPG of an automobile given its six profile attributes.

Before training a fuzzy inference system, we divide the data set into training and test sets. The training set

Table 1. *Samples of the MPG training data set. (The last column is used for reference only and not for prediction.) The data set is available from the UCI Repository of Machine Learning Databases and Domain Theories (FTP address:* `ftp://ics.uci.edu/pub/machine-learning-databases/auto-mpg`*). More historical information about the data set can be found there.*

No. of Cylinders	Displacement	Horse Power (HP)	Weight	Acceleration	Year	MPG	Car name
8	307	130	3504	12	70	18	Chevrolet Chevelle Malibu
6	198	95	2833	15.5	70	22	Plymouth Duster
4	90	75	2108	15.5	74	24	Fiat 128
8	260	110	4060	19	77	17	Oldsmobile Cutlass Supreme
4	89	62	2050	17.3	81	37.7	Toyota Tercel
4	107	75	2205	14.5	82	36	Honda Accord
4	120	79	2625	18.6	82	28	Ford Ranger

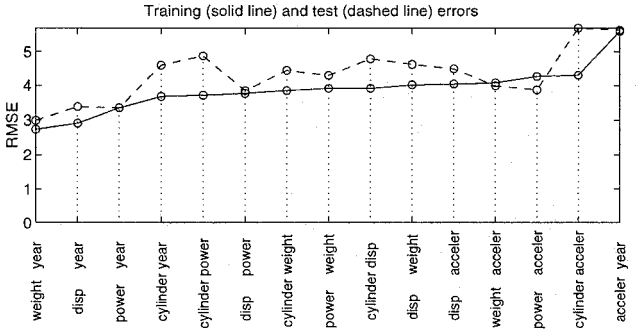

Figure 2. *Fifteen two-input fuzzy models for automobile MPG prediction.*

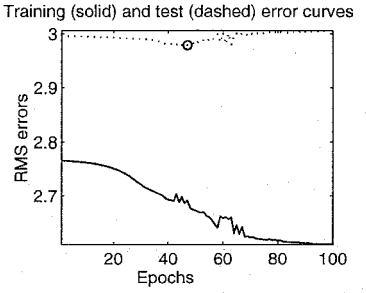

Figure 3. *Error curves obtained by training a fuzzy inference system to predict MPG.*

is used to train (or tune) a fuzzy model, while the test set is used to determine when training should be terminated. to prevent overfitting. The 392 instances are randomly divided into training and test sets of equal size (196).

Grid partitioning is the most frequently used input partitioning method for ANFIS. However, for a problem with six inputs, grid partitioning leads to at least $2^6 = 64$ rules, which results in $(6+1) \times 64 = 448$ linear parameters if we want to stick to the first-order Sugeno fuzzy model. This implies that we have too many fitting parameters and the resultant model is not reliable for unforeseen inputs. To deal with this, we can either select certain inputs that have more prediction power instead of using all the inputs, or choose tree or scatter partitioning [6, 7] instead. Here we consider only input dimension reduction and apply the input selection method described in Section 3.

If we only want to select the two most relevant inputs as predictors, we can cycle through all the inputs and build $C_2^6 = 15$ ANFIS models. As described in Section 3, the performance of an ANFIS model after the first epoch is usually a good index of how well the model will perform after further training. Based on this heuristic observation, we built 15 fuzzy models each with a single epoch of ANFIS training; it took about 16 seconds on a 486-DX100 PC with 16 MB RAM. The results are shown in Figure 2 with two curves representing training and test RMSE (root-mean-squared errors). We reordered these 15 models according to their training errors. Obviously, the best model takes "weight" and "model year" as the input variables, which is quite reasonable. In this case, both error curves are more or less consistent; this implies that the training and test data were evenly distributed across the original data set. In particular, we will end up with the same model if we pick the one with the smallest test error. Note that Figure 2 is based only on one epoch of training; more reliable results can be

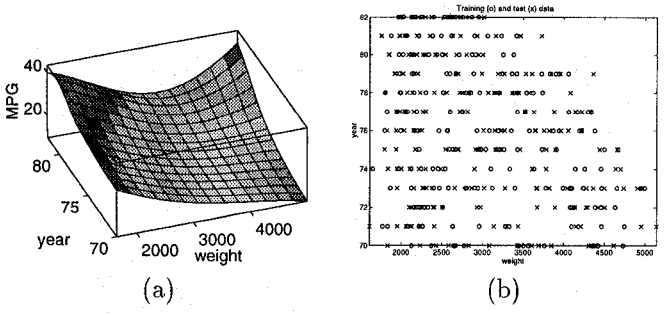

Figure 4. *Membership functions in chaotic time series prediction: (a) ANFIS surface for MPG prediction; (b) training and checking data distribution.*

obtained if more training epochs are allotted to each of the 15 models.

Once we have selected the model with "weight" and "model year" as inputs, we can refine its performance via extended ANFIS training. Figure 3 shows the error curves for 100 epochs of training. The training error decreases all the way, but the test error, after decreasing initially, reaches a plateau, oscillates a little bit, and then increases. Usually we use the test error as a true measure of the model's performance; therefore the best model we can achieve occurs when the test error is minimal. This corresponds to the circle in Figure 3; though further training beyond this point decreases the training error, it will degrade the performance of the fuzzy inference system on unforeseen inputs.

As a comparison, we now look at the result of linear regression, where the model is expressed as

$$\begin{aligned} MPG &= a_0 + a_1 * \text{cyl} + a_2 * \text{disp} + a_3 * \text{hp} \\ &\quad + a_4 * \text{weight} + a_5 * \text{accel} + a_6 * \text{year}, \end{aligned}$$

with a_0, a_1, $\cdots$, a_6 being seven modifiable linear parameters. The optimum values of these linear parameters were obtained directly by the least-squares method; the training and test errors are 3.45 and 3.44, respectively. In contract, after 100 epochs of training, the minimal test error is 2.98, at which the training error is 2.61. It is worth noting that the linear model takes all six inputs into consideration, but the error measures are still high since MPG prediction is nonlinear. On the other hand, our input selection technique of choosing the two most relevant inputs can result in a nonlinear mapping with lower error measures.

Figure 4 (a) is a three-dimensional surface of the fuzzy model with the smallest test error. This is a smooth nonlinear surface, but it raises a legitimate question: why does the surface increase toward the right upper corner? This is an apparently spurious result that states that heavy old cars have higher MPG ratings. The anomaly can be explained by the scatter plot of the data distribution in Figure 4 (b), in which it is obvious that the lack of data (due to the tendency of automobile manufacturers to begin building small compact cars instead of big heavy ones during mid 70s) is responsible. In other words, our trained fuzzy inference system is good at interpolation, but not at extrapolation. Without input selection, it is hard to visualize the data qualify the scope of its validity before interpreting ANFIS output correctly.

5. Nonlinear System Identification

This section applies ANFIS to nonlinear system identification, using the well-known Box and Jenkins gas furnace data [1] as the training data set. This is a time-series data set for a gas furnace process with gas flow rate $u(t)$ as the furnace input and CO_2 concentration $y(t)$ as the furnace output. We want to extract a dynamic process model to predict $y(t)$ using ten candidate inputs to ANFIS: $y(t-1)$, $y(t-2)$, $y(t-3)$, $y(t-4)$, $u(t-1)$, $u(t-2)$, $u(t-3)$, $u(t-4)$, $u(t-5)$, and $u(t-6)$. The original data set contains 296 $[u(t), y(t)]$ data pairs; converting the data so that each training data point consists of $[y(t-1), \cdots, y(t-4), u(t-1), \cdots, u(t-6); y(t)]$ (the last one is the desired output) reduces the number of effective data points to 290. We use the first 145 data points as the training set, the remaining 145 as the test set.

Since we have ten candidate input variables for ANFIS, it is reasonable to do input selection first to rate variable priorities and reduce the input dimension. For dynamic system modeling, the inputs selected for ANFIS must contain elements from both the set of historical furnace outputs $\{y(t-1), y(t-2), y(t-3), y(t-4)\}$ and the set of historical furnace inputs $\{u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6)\}$. For simplicity, we assume that there are two inputs for ANFIS, one is from the historical furnace outputs, the other from the historical furnace inputs. In other words, we have to build 24 ($= 4 \times 6$) ANFIS models with various input combinations, and then choose the one with the smallest training error for further parameter-level fine-tuning. We could have chosen the ANFIS with the smallest test error, but this would have led to "indirect training on test data". The input selection procedure took about 40 seconds on a 486-DX100 PC with 16 MB RAM. Figure 5 shows the performance of these 24 ANFIS models, they are listed according to their training errors. Note that each ANFIS has four rules, and

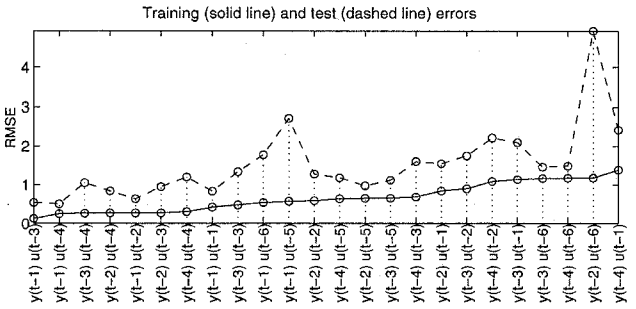

Figure 5. *Input selection for Box-Jenkins data.* .

the training took only one epoch each to identify linear parameters. If computing power is not a problem, we could then assign more training epochs to each ANFIS.

In Figure 5, we can see that the ANFIS with $y(t-1)$ and $u(t-3)$ as inputs has the smallest training error, so it is reasonable to choose this ANFIS for further parameter tuning. Figure 6 shows the result of training this ANFIS for 100 epochs. In particular, Figure 6 (a) displayed the training and test error curves; the optimal ANFIS parameters were obtained at the time when the test error reached the minimum indicated by a small circle. Figure 6 (b) shows the data distribution; it demonstrates that the training and test data do not cover the same region. Better performance can be expected if they cover roughly the same region; this can be achieved by using other schemes to divide the original data set. (For instance, the training and test sets can be interleaved in the original data set.) Figure 6 (c) displays the desired curve and ANFIS prediction; the performance for time index from 1 to 145 is better since this is the domain from which the training data was extracted. Figure 6 (d) is the ANFIS surface; it is cut off at the maximum and minimum of the desired output.

6. Concluding Remarks

In this paper, we present a quick and staightforward way of input selection for neuro-fuzzy modeling using ANFIS (Adaptive Neuro-Fuzzy Inference Systems) [2, 3]. The proposed method was tested on two real-world problems: the nonlinear regression problem of automobile MPG (miles per gallon) prediction, and the nonlinear system identification using the Box and Jenkins gas furnace data [1]. For the automobile MPG prediction problem, we also compared the ANFIS approach with input selection to the common linear regression method, and found that a nonlinear ANFIS model with two inputs performed better than a linear

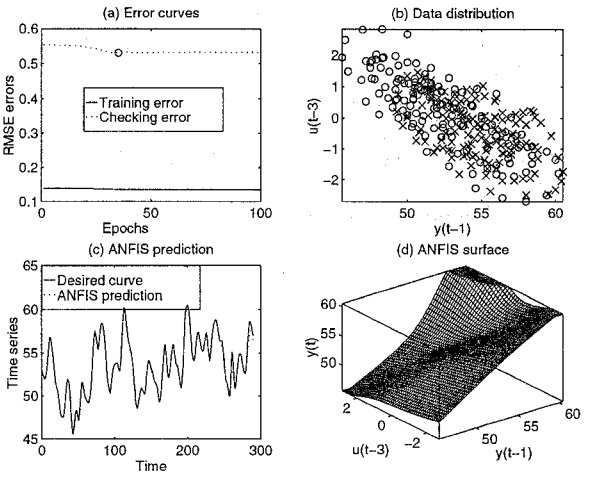

Figure 6. *ANFIS for Box-Jenkins data: (a) training and checking error curves; (b) training and checking data distribution; (c) desired system response and ANFIS prediction; (d) ANFIS surface.* .

model with six inputs.

In summary, input selection reduces training data dimension, which in turn allows grid partitioning for ANFIS modeling and effective data visualization for qualifying valid scopes of ANFIS. The proposed input selection method, though heuristic by nature, can provide an effective means to determining input priorities for ANFIS modeling.

References

[1] G. Box and G. Jenkins. *Time Series Analysis, Forcasting and Control*, pages 532–533. Holden Day, San Francisco, 1970.

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

[3] J.-S. R. Jang. ANFIS: Adaptive-Network-based Fuzzy Inference Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(03):665–685, May 1993.

[4] J.-S. R. Jang and N. Gulley. The Fuzzy Logic Toolbox for use with MATLAB. The MathWorks, Inc., Natick, Massachusetts, 1995.

[5] J.-S. R. Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159, Jan. 1993.

[6] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *The Proceedings of the IEEE*, 83(3):378–406, Mar. 1995.

[7] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence, 1996. To be published by Prentice Hall.

[8] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Apl. Math.*, 2:164–168, 1944.

[9] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[10] D. W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11:431–441, 1963.

[11] E. Mizutani and J.-S. R. Jang. Coactive neural fuzzy modeling. In *Proceedings of the International Conference on Neural Networks*, pages 760–765, Nov. 1995.

[12] D. W. Ruck, S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley. Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):686–691, 1992.

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

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

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

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

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

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