FUZZY MODELING BASED ON GENERALIZED NEURAL NETWORKS
AND FUZZY CLUSTERING OBJECTIVE FUNCTIONS*

Chuen-Tsai Sun

Jyh-Shing Jang

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720

Keywords: Fuzzy Modeling, Neural Networks, Fuzzy Clustering

Abstract

Fuzzy modeling is the task of identifying the structure and parameters of a fuzzy if-then rule base. In this paper we propose a new approach to formulate fuzzy if-then rules based on clustering objective functions. The membership functions are then calibrated with a new technique named generalized neural networks to achieve a desired input/output mapping. The learning procedure is basically a gradient descent algorithm. We also employ Kalman filter algorithm to improve the overall performance.

To explain this new approach, first we introduce the concept of generalized neural networks (GNN’s) and derive a supervised learning procedure based on a gradient descent algorithm to update the parameters in a GNN. Next, we discuss the use of Kalman filter algorithm to minimize the square error. From the simulation results, it is summarized that the adaptively adjusted inference system performs well on a given surface structure. To cope with the issues of large number of input variables, we introduce a hill-climbing algorithm to partition the feature space and to find a suitable initial state for the GNN phase to fine-tune the parameters. This structure identification algorithm is based on the idea of finding a set of local input-output relations (rules) to describe a system. It has found many practical applications not only in control fields but also in artificial intelligence and operations research, such as estimation, inference, and prediction. We can perform fuzzy modeling by extracting knowledge from human experts and by transforming the expertise into rules and membership functions. The resulted system can then be tuned with monitoring of performance by trial and error. However, depending on human introspection and experience results in some severe problems:

1. The mechanisms underlying human perception are still largely under investigation.
2. Even when human experts exist, their knowledge is often incomplete and episodic rather than systematic.
3. No formal way to transform experiences or knowledge of human experts to rule bases of a fuzzy inference system.
4. The lack of adaptability or learning ability to tune the membership functions so as to minimize error measures.

Introduction

The concept of system modeling is closely related to pattern classification, case-based reasoning, and learning from examples. It plays an important role in rule-based control, data compression, pattern recognition, expert systems, and multiple-objective decision processes. It is known that conventional approaches of system modeling, which are based on mathematical tools, e.g., differential equations, perform poorly in dealing with complex and uncertain systems. The reason is that in many cases, it is very difficult to find a global functional or analytical structure for a nonlinear system.

On the contrary, a fuzzy inference system uses fuzzy if-then rules to describe a system. Figure 1 gives a typical example of a fuzzy rule. Consequently, it can express the qualitative aspect of human reasoning without using any precise mathematical model of the system. Based on pattern matching against input signals, a number of fuzzy rules are triggered parallelly with various values of firing strength. Individually invoked actions are combined together by a defuzzification mechanism to give a single output. The inference process is called fuzzy reasoning, as illustrated in Figure 2.

Figure 1: A typical fuzzy control rule.

Figure 2: Fuzzy reasoning.
As a result, researchers have been trying to automatize the modeling process based on numerical training data. The task can be divided into two parts: structure identification and parameter identification. The former is related to finding a suitable number of rules and a proper partition of the feature space. The latter concerns the adjustment of the membership functions.

In this paper we will explore this research frontier and propose a general modeling scheme for fuzzy rule-based architectures. Our approach is a two-phase design in which we first present a method for parameter identification assuming a given structure. This method is based on generalized neural networks. Then, after validating its effectiveness, we provide another algorithm for structure identification and connect them together. Finally, we add the function of feature selection to the modeling scheme to make it complete.

Generalized Neural Networks

A generalized neural network (GNN), see Figure 3, is a multilayer feed-forward network in which each node performs a particular function (node function) based on incoming signals and a set of parameters pertaining to this node. When the set of parameters is empty, we use a circle to denote the node, otherwise we use a square. The types of node functions may vary from node to node, and the choice of node functions depends on the overall function that a GNN is designed to carry out. Note that the links in a GNN only serve the purpose of transmitting signals between nodes; no weights are associated with links.

![Figure 3: A generalized neural network (GNN).](image)

The parameter set of a GNN is the union of the node parameter sets. In order to minimize the output error measure of a GNN, these parameters are updated according to a given training data set and a learning algorithm described below.

Cybenko [3] showed that a continuous NN with two hidden layers and any fixed continuous sigmoidal nonlinear can approximate any continuous function arbitrarily well on a compact set. Therefore a GNN node can always be replaced by an ordinary NN with the same input/output characteristics. In this context, GNN's can be considered as a super set of NN's.

Suppose a given GNN has L layers and the k-th layer has \( #(k) \) nodes. We denote the node in the i-th position of the k-th layer by \((k,i)\), and its node function (or node output) by \( O^k_{(k,i)} \). Since a node output depends on its incoming signals and its parameter set, we have

\[
O^k_{(k,i)} = O^k_{(k,i)}(O^k_{(k,1)}, \ldots, O^{k-1}_{(k-1,i)}, a, b, c, \ldots)
\]

where \( a, b, c, \ldots \) are parameters pertaining to this node.

Assuming the given training data set has \( P \) entries, we define the error measure (or energy function) on p-th entry of training data as the square of error:

\[
E_p = \sum_{m=1}^{#(k)} (T_{m,p} - O^k_{(k,i)})^2
\]

where \( T_{m,p} \) is the m-th component of p-th target output vector, and \( O^k_{(k,i)} \) is the m-th component of the output vector produced by the presentation of the p-th input vector. (For brevity, we omit the subscript \( p \) in \( O^k_{(k,i)} \)). Hence, the overall error measure is \( E = \sum_{p=1}^{P} E_p \). Now we introduce a learning procedure that implements gradient descent in \( E \) over the parameter space.

For the output-layer node at \((L,i)\), we can calculate \( \frac{\partial E_p}{\partial O^L_{(L,i)}} \) readily from Equation 2:

\[
\frac{\partial E_p}{\partial O^L_{(L,i)}} = -2(T_{i,p} - O^L_{(L,i)})
\]

For the internal node at \((k,i)\), we can apply the chain rule to write \( \frac{\partial E_p}{\partial O^k_{(k,i)}} \) as

\[
\frac{\partial E_p}{\partial O^k_{(k,i)}} = \sum_{n=1}^{#(k)} \frac{\partial E_p}{\partial O^n_{(k+1,i)}} \frac{\partial O^n_{(k+1,i)}}{\partial O^k_{(k,i)}}
\]

which is then calculated by Equations 3 and 4. Now if \( \alpha \) is a parameter of the given GNN, we have

\[
\frac{\partial E_p}{\partial \alpha} = \sum_{i \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}
\]

where \( S \) is the set of nodes whose outputs depend on \( \alpha \). The derivative of the overall error measure \( E \) to \( \alpha \) is then calculated as:

\[
\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}
\]

Accordingly, the update amount for the generic parameter \( \alpha \) is

\[
\Delta \alpha = -\eta \frac{\partial E}{\partial \alpha}
\]

where \( \eta \) is a learning rate and can be further expressed as

\[
\eta = \frac{k}{\sqrt{\sum_{i=0}^{P} \left( \frac{\partial E_p}{\partial \alpha} \right)^2}}
\]

where \( k \) is the step size of the gradient descent.

GNN-based Fuzzy Inference System

An example of fuzzy if then rule used in a fuzzy inference system is

\[
\text{If pressure is high and temperature is low, then volume is small.}
\]

where pressure and temperature are input variables, volume is an output variable, high, low and small are linguistic terms characterized by appropriate membership functions. Each fuzzy rule represents a local description of the system's behavior.

Several types of reasoning methods used in fuzzy models have been proposed in the past years. Here we adopt the one proposed by Takagi and Sugeno [9]. The firing strength of each fuzzy rule is calculated as the product of the membership values in the premise part; and the final output is obtained as the weighted average of each rule's consequence.

For the purpose of illustration, we assume the system to be modeled has two input variables \( x_1 \) and \( x_2 \), one output variable

2925
y, and three rules. The basic configuration of the GNN-based fuzzy inference system is shown in Figure 4, where nodes in the same layer have the same type of node functions explained below.

Layer 1: Each node in this layer is a square node with node function

\[ O_i = \mu_A(x_i) = \frac{1}{1 + e^{-(x_i-a)/b}} \]  

(10)

where \( x_i \) is one of the input variables, \( \{ a, b, c \} \) is the parameter set, and \( A \) is the linguistic term associated with this node function. As the values of \( a, b \) and \( c \) change, this bell-shaped node function varies accordingly, thus exhibiting various concepts of corresponding linguistic terms. Parameters in this layer are usually called premise parameters.

Layer 2: Each node in this layer is a circle node labeled II which multiplies the incoming signals and sends the product out. Each node output corresponds to the firing strength of a rule.

Layer 3: Each 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.

Layer 4: Each node in this layer is a square node labeled \( D \) with node function

\[ O^4 = w_i \ast (d \ast x_1 + e \ast x_2 + f) \]  

(11)

where \( w_i \) is the output from layer 3, and \( \{ d, e, f \} \) is the parameter set. Parameters in this layer are usually called consequence parameters.

Layer 5: There is only one circle node labeled \( S \) that sums all incoming signals.

Thus we have constructed a fuzzy inference system with a generalized neural network. Now the proposed learning procedure can be applied to tune the parameters according to given training data. To speed up the tuning process, we employ the Kalman filter algorithm described below.

Kalman Filter Algorithm

From Figure 4, it is observed that given the values of the premise parameters and \( P \) training data, we can form \( P \) linear equations in terms of the consequence parameters. For simplicity, let's assume we have \( m \) consequence parameters and the resulting linear equations can be expressed in the following matrix form:

\[ AX = B \]  

(12)

where the elements of \( X \) are consequence parameters. Several approaches have been developed to solve this kind of over-constrained simultaneous linear equations, and one of the most concise [1] is

\[ X^* = (A^T A)^{-1} A^T B \]  

(13)

where \( A^T \) is the transpose of \( A \), and \( (A^T A)^{-1} \) is called the pseudo-inverse of \( A \) if \( A^T A \) is non-singular.

In many cases, the row vectors of matrix \( A \) (and the corresponding elements in \( B \)) are obtained sequentially, hence it is desirable to compute the least-square estimate of \( X \) in Equation 12 recursively. Let the \( i \)-th row vector of matrix \( A \) defined in Equation 12 be \( a_i \), and the \( i \)-th element of \( B \) be \( b_i \), then \( X \) can be calculated recursively using the following formula:

\[
\begin{align*}
X_{i+1} &= X_i + S_{i+1} a_i^T (b_i - a_i x_i) \\
S_{i+1} &= S_i - \frac{S_i a_i^T S_i}{1 + a_i^T S_i a_i}, \quad i = 0, 1, \ldots, P - 1
\end{align*}
\]  

(14)

with initial conditions

\[ X_0 = 0 \text{ and } S_0 = \gamma I \]  

(15)

where \( \gamma \) is a positive big number, \( I \) is the identity matrix of dimension \( m \times m \).

The least-squares estimate of \( X \) can be interpreted as a Kalman filter [4] for the process

\[ X(k + 1) = X(k) \]  

(16)

\[ Y(k) = A(k) X(k) + \text{noise} \]  

(17)

where \( X(k) = X_k, Y(k) = b_k \) and \( A(k) = a_k \). Therefore the formula in Equation 14 is usually referred to as a Kalman filter algorithm.

Simulation Results

We use one example to show the validity of the proposed parameter identification mechanism. The training data set is calculated from a nonlinear function,

\[ z = z(x, y) = \frac{\sin(x)}{x} \cdot \frac{\sin(y)}{y} \]  

(18)

see the target function in Figure 5(a). The inference system has 2 inputs, 1 output, and 16 rules. We use 121 training data which are sampled uniformly from \([-10, 10] \times [-10, 10]\) of the input space of the original system. Initially, the feature (input) space is uniformly partitioned into 16 regions, each corresponding to a rule.

The initial values of the premise parameters are set in such a way that the membership functions along \( X \) and \( Y \) axes satisfy \( \epsilon \) completeness [6] (\( \epsilon = 0.5 \) in our case), \( \rho \) normality and \( \rho \) convexity [5]. Though these initial membership functions are set heuristically and subjectively, they do provide an easy interpretation parallel to human thinking. The initial and final membership functions are shown in Figure 5(c). Once the premise parameters are set, the consequence parameters are found by Kalman filter algorithm, hence the initial values of consequence parameters are irrelevant. The 3-D diagram of the training data is shown as the target surface in Figure 5(a). Other identified surfaces at different epoch numbers are also shown in the figure. In order to evaluate the performance of the GNN, we define an
average percentage error (APE) as

\[ APE = \frac{\sum_{i=1}^{P} |T(i) - O(i)|}{\sum_{i=1}^{T} |T(i)|} \cdot 100\% \]  

where \( P \) is the number of training data, \( T(i) \) and \( O(i) \) are the \( i \)-th desired output and the calculated output, respectively. Though the final values of parameters are not the same as those used in the original fuzzy inference system, the final surface after 200 epochs is close to the target surface with APE at 0.4640%, see Figure 5(b), a fairly good result.

Structure Identification

As we have experimentally verified the validity of the parameter identification mechanism using GNN, we now try to tackle our next goal: structure identification. One of the practical needs of identifying rule base structures is based on the consideration of computational complexity. When the number of input variables is large, the partition method we employed before, such as in the previous example, is no longer applicable. Assume we have \( n \) variables and we uniformly partition the feature space associated with one variable into \( p \) linguistic terms, it will result in \( p^n \) rules in total, just as we have \( 4^3 = 16 \) rules in Example 1. Since the architectural and computational complexity is proportional to the number of rules, structure identification becomes necessary when the number of variables increases. We use fuzzy clustering to accomplish this goal.

In terms of fuzzy rule-base modeling, clustering can be viewed as finding a set of fuzzy points, each representing a rule, which as a whole cover the given training set. The more rules involved, the finer coverage is likely to achieve, see Figure 6.

![Figure 5: Example 1: Parameter Identification, (a) target surface and identified surfaces at different stages, (b) average percentage error, (c) initial and final membership functions.](image)

![Figure 6: Fuzzy points covering a crisp curve. Various numbers of fuzzy points result in different degrees of information granularity.](image)

![Figure 7: A fuzzy guillotine partition of a two-dimensional feature space.](image)

Most existing fuzzy clustering algorithms aim at optimizing nonlinear objectives of the resulting clusters. Dynamic programming is usually used to implement the algorithms and, in general, the clustering process is time-consuming. However, since our goal is just to find a satisfiable initial state for the GNN phase to fine-tune the parameters, it makes no sense to spend a lot of time in optimizing the cluster criteria, no matter what the objective functions might be. Consequently, we adopt a computationally efficient hill-climbing method in identifying the proper structure.

We use guillotine cuts in partitioning the feature space. By a guillotine cut, we mean a cut which is made entirely across the subspace to be partitioned, see Figure 7 for an example. Each of the regions so produced can then be subjected to independent guillotine cutting. At the beginning of the \( i \)-th iteration step, the feature space is partitioned into \( i \) regions. Now another guillotine cut is applied to one of the regions to further partition the entire space into \( i+1 \) regions. The objective functions discussed in the next section are used to determine which region is chosen for the cutting and where the cut should be. Once the cut is made, the hill-climbing sequence proceeds to its next step. This process continues until a given number of clusters are generated. Sugeno and Kang [8] used a similar method to partition the input space.

Further, at step \( i \), a fuzzy set is defined for each cluster with the following membership function which will be used in the
Objective Functions

As analyzed by Bezdek [2], various clustering algorithms can suggest radically different substructures in the same data set. To achieve a meaningful structure for a fuzzy rule base, we have to select appropriate objective functions for clustering. In our approach, we use two objective functions, one (JD) is a density measure, the other one (JT) is a typicality measure.

JD was proposed by Ruspini [7],
\[
JD = \sum_{i=1}^{P} \sum_{j=1}^{C} \left( (\mu_{ij} - \mu)^2 - d_{jk}^2 \right)
\]

(21)

where \( P \) is the number of training data, \( C \) is the number of clusters, \( \mu_{ij} \) is the membership of the \( k \)th point (\( z_k \)) in the \( i \)th cluster, \( \mu \) is the mean of all \( \mu_{ij} \), \( d_{jk} \) is the distance (or the measure of dissimilarity) between sampling points \( j \) and \( k \), and \( \mu_j \) is the membership of point \( j \) in cluster \( i \). As pointed out by Ruspini, JD is a measure of cluster quality based on local density, because JD will be small when the terms in Equation 21 are individually small: in turn, this will occur when close pairs of points have nearly equal fuzzy memberships in the \( C \) clusters.

JT is a variation of the least square functional proposed by Bezdek [2],
\[
JT = \sum_{k=1}^{P} \sum_{i=1}^{C} \mu_{i} d_{ik}^2
\]

(22)

where \( d_{ik} \) is the distance from point \( k \) to the center (or prototype) \( \bar{v}_i \) of cluster \( i \), as defined in the previous section. We call JT a typicality measure because it will be small when points in a cluster adhere tightly (have small \( d_{ik}^2 \)'s) to their cluster centers \( \bar{v}_i \).

Density and typicality are important measures because they are closely related to two important characteristics of linguistic terms: support and core, respectively. Support is the range of non-zero membership values (\( \mu > 0 \)), whereas core is the range of full membership (\( \mu = 1 \)). In general, we want a linguistic term to have a strong support (high density, or small \( JD \)) and a representative core (good prototype, or small \( JT \)). Thus, it is reasonable to choose \( JD + JT \) to be our objective function. In other words, for each possible guillotine cut, we calculate \( JD + JT \) of the resulting partition. Then, we select the partition with the least \( JD + JT \) value as our next hypothesis to continue the hill-climbing process.

Now we use a second example to show the combined effect of structure identification (fuzzy clustering) and parameter identification (GNN). In this example we have 3 input variables and 1 output variable. We use 216 training data entries which are calculated from the following function:

\[
w = w(x, y, z) = (8.5 \exp(-3) + 5 \tanh(\frac{2}{10} - \frac{3}{10}z))
\]

(23)

The simulation results with 10, 20, and 30 rules are shown in Figure 8.

Figure 8: Example 8: Structure and Parameter Identification.
The upper, middle, and lower curves are the error percentages of simulation (300 epochs each) with 10, 20, 30 rules, respectively.

Feature Selection

The above discussion is based on one assumption: the input variables are of equal importance. However, in applications of automatic control, pattern classification, or multi-criteria decision making, this assumption usually is not true. Further, a general modeling scheme should include two parts: feature selection and cluster analysis. Since the proposed model serves the second goal pretty well, i.e., it successfully identifies the structure and parameters of fuzzy clusters (rules), we want to enhance it to make the modeling mechanism complete. Here, we use the concept of weight of importance in fuzzy pattern matching to realize feature selection.

Assume each input variable \( x_i \) is associated with a weight of importance \( w_i \in [0,1] \). We generalize Equation 10 to a weighted degree of match, denoted by \( s_i \),
\[
s_i = 1 - w_i \cdot \mu(x_i)
\]

(24)

Note that the less important a variable is (\( w_i \) small), the less role \( s_i \) plays in the multiplication at the next layer in the GNN. On the other hand, if a variable is of full importance, the above equation reduces to Equation 10.

Correspondingly, we add one layer, the \( 0 \)th layer, to the GNN structure, see Figure 9. Each node in layer 0 is either an input node (a circle) or a weight node (a square labeled \( W \)). The weight of importance is the only parameter of a weight node. The initial value of weight is defaulted at 1.0 but can be assigned to any value in \( [0,1] \) by users.

To show the ability of the enhanced GNN for feature selection, we use a training data set in which there are three input variables and the output is independent of the third input variable. The changes of weights are shown in Figure 10. After 49 training epochs, the weights were stabilized to values 1.0, 0.65, and 0.0, respectively. In terms of feature selection, the proposed mechanism isolated the third input variable successfully. Once a weight of importance is stabilized below a certain threshold value, the corresponding variable is considered unimportant in the system to be modeled. Thus, the variable can be neglected and a simplified structure/parameter identification process can be resumed to find a even better solution.
Concluding remarks

We conclude this paper with Figure 11. We proposed a general modeling scheme for fuzzy rule-based inference system, which can be used in automatic control, pattern recognition, decision analysis, and many other fields where human expertise is either unavailable or episodic. The modeling scheme is a two-phase design. Structure identification is realized by fuzzy clustering based on guillotine cuts and objective functions. The two measures, density and typicality, we chose to evaluate clusters have a sound theoretical background in fuzzy sets. Parameter identification, on the other hand, is implemented with generalized neural networks. Besides employing Kalman filters to improve the overall performance, we also introduced the concept of weight of importance to achieve the goal of feature selection.

Since the proposed learning algorithm is a gradient descent procedure, sometimes it could get stuck around local minima. However, even the global minimum has not been found, the final average percentage error is still acceptable, as shown in the examples. As long as we can identify a fairly good surface structure as the initial state, the learning process has a good possibility to adjust parameters without being trapped in local minima.

In this paper, multiplication is used as the fuzzy conjunction operator simply for convenience of differentiation. We can use any other t-norms to serve the same purpose.

We are now experimenting the scheme on more complicated cases with more variables or sparse sampling data. A challenging direction of future work is to let the modeling scheme find the proper number of rules needed for the targeted system. To achieve this goal, the information in the GNN learning process should be feedbacked to the clustering module, and a set of delicate performance indices should be designed.

References