Hybrid Techniques on Fuzzy Neural and Neural Fuzzy Systems
--Neuro-Fuzzy Classification and Function Approximation

Haitao Lu, Di Wu, and V. Miraftab (Student member IEEE)

Abstract-- In this paper, we will study different existing techniques for the integration of fuzzy logic systems and neural networks. First, we will explain fuzzy methods in neural networks called neural fuzzy systems, and then neural methods in fuzzy systems called neural fuzzy systems. To illustrate the performance and applicability of proposed fuzzy neural hybrid model, some experimental examples of these two cases are carried out.

Index terms-- fuzzy neural, neural fuzzy, neuro-fuzzy, fuzzy-neuro, BPFLS, classification, NEFCLASS, function approximation, ANFIS

1. INTRODUCTION

Although the structure of fuzzy logic systems (FLS) and artificial neural networks (ANN) are different, they share a rather complementary nature as far as strengths and weaknesses are concerned. In this survey, we will explain different approaches to integrate these two types of systems to take benefit of merits of both FLS and ANN at the same time.

The advantages of FLS over ANN in essence can be summarized as follows:
1) A FLS can be easily interpreted or modified.
2) The knowledge presentation is Linguistic, which models human way of thinking.
3) It can be easily and efficiently verified or optimized.
4) It can use fuzzy numbers and relations, which are more general than crisp numbers and relations.
5) The parameters of a FLS are associated with membership functions for physically meaningful quantities. It gives the designer the chance to obtain very good initial values for them, whereas in many other training algorithms such as feed-forward neural networks, the parameters have no physical meaning and usually must be chosen randomly.

The advantages of ANN over FLS can be summarized as follows:
1) ANN Trains itself by learning from data-pairs.
2) No prior knowledge is required.
3) It is adaptable to changes in the system.

To integrate fuzzy logic and neural networks, one can assume two different approaches: 1) Incorporating fuzzy logic into neural network models, and 2) Using neural networks learning methods into fuzzy systems to make them more adaptive. The first kinds of systems are called fuzzy logic-based neural network models or fuzzy neural networks, whereas the second types of systems are called neural fuzzy systems.

Historically, as reflected in classical mathematics, we commonly seek precise and “crisp” descriptions of things or events. This precision is accomplished by expressing phenomena in numerical values. However, due to fuzziness, classical mathematics can encounter substantial difficulties. G. Cantor, a German mathematician, founded set theory in 1874. One of the important methods used by Cantor in creating sets is the comprehension principle. Cantor set theory had given rise to some restrictions on the use of mathematics. In fact, according to Cantor’s claim, the objects that form the set are definite and distinct from each other. Thus, the property used to form the set must be crisp: for any object, it must be precise whether the property is satisfied or not. In order to enable mathematics to describe fuzzy phenomena, it is important to reform Cantor set concept to define a new kind of set called a fuzzy set. Starting with Zadeh’s fuzzy set theory in 1965, tremendous research has demonstrated success of the fuzzy sets and systems in both theoretical and practical areas.

On the other hand, neural networks are deliberately constructed to make use of some organizational principles resembling those of human brain. A human brain consists of millions of neurons of many different types. The history of neural networks goes to 1943, when McCulloch and Pitts proposed a neuron model to model the biological neurons in brain.

Fuzzy logic and neural network are natural complementary tools in building intelligent systems. While fuzzy logic performs an inference mechanism under cognitive uncertainty, artificial neural networks offer exciting advantages, such as learning, adaptation.

Probably Lee and Lee were the first to introduce fuzzy sets into neural networks, where author generalized the McCulloch-Pitts model by using intermediate values between zero and one [1-2]. A survey paper [3] in 1990 discussed the fusion of neural nets and fuzzy logic. However, very little research on fuzzy neural nets was done by them with the exception of Keller and Hunt [4] and Yamakawa's [5-8] fuzzy neuron. Keller and Hunt proposed adding fuzzy membership function to the perceptron. Yamakawa's initial fuzzy neuron
was of type FNN$_1$. A learning algorithm was applied to the weights.

Ishibuchi et al. [9-10] presented the FNN$_1$ with learning on the real weights performed by $\alpha$-cuts on the signals. Gupta et al. [11-14] presented various models for FNN$_1$, FNN$_2$, and FNN$_3$. Hayashi et al. [15] also discussed these models, presented applications and proposed learning algorithms.

Modern neuro-fuzzy systems are usually represented as a multilayer feedforward neural network. The well-known ANFIS model by Jang [16] implements a Sugeno-like fuzzy system in a network structure, and applies a mixture of backpropagation and least mean square procedure to train the system.

In this paper, we will first explain different types of fuzzy neural networks, which include 3 types of fuzzy neurons. Next, we will switch to neural fuzzy systems and their different types. Finally, we will show some examples to give a better view of these kinds of systems.

II. FUZZY NEURAL NETWORKS

In this section, we introduce different basic models of fuzzy neurons from which fuzzy neural networks can be built. A fuzzy neuron functions in a similar way as the classical McCulloch-Pitts neuron does except that it can reflect the fuzzy nature of a neuron and has the ability to cope with fuzzy information. In literature, three types of fuzzy neurons are explained:

1) Fuzzy neuron of type I, which has crisp inputs and fuzzy weights.
2) Fuzzy neuron of type II, which has fuzzy inputs, outputs and fuzzy weights.
3) Fuzzy neuron of type III, which is a fuzzy neuron described by fuzzy logic equations (rules).

Fig. 1 shows a fuzzy neuron of type I denoted by $N$, where includes $n$ crisp inputs, and fuzzy weights $A_r$. The result of each weighting operation is the membership value of the corresponding input. Then the membership values will be aggregated to give an output in the interval $[0,1]$, which can be considered as the level of confidence.

Fig. 2 shows a fuzzy neuron of type II denoted by $N$, where includes $n$ fuzzy inputs, and fuzzy weights $A_r$. The resulting output is also a fuzzy set. Usually in problems with this type of structure, the system is expected to learn the fuzzy weights $A_r$ from some training data $(X_i, Y_k)$.

\begin{equation}
\text{IF } X_1 \text{ AND } X_2 \text{ AND } \cdots \text{ AND } X_n, \text{ THEN } Y
\end{equation}

$R$ in Fig. 3 is the following fuzzy relation in equation (2).

\begin{equation}
R = X_1 \times X_2 \times \cdots \times X_n \times Y
\end{equation}

resulting output is also a fuzzy set. Usually in problems with this type of structure, the system is expected to learn the fuzzy weights $A_r$ from some training data $(X_i, Y_k)$.

A fuzzy neural network of type III is very useful for rule extraction from training data in fuzzy expert systems [19].

III. NEURAL FUZZY NETWORKS

In this section, we discuss different application of neural networks to be used in fuzzy logic systems. The basic structure is a FLS. Neural networks can help in optimal design of fuzzy logic systems in different ways:

1) NN can help in identifying/extracting the rules.
2) NN can tune the parameters of a FLS.
3) NN can be used in series with a FLS.
4) NN can determine the membership functions.

The neuro-fuzzy systems are mainly categorized into two different types: 1) Co-operative neuro-fuzzy systems. 2) Hybrid neuro-fuzzy systems. In co-operative neural fuzzy
systems, neural network is used to tune the parameters of an FLS. The process of tuning can be done either online or offline. Moreover, the neural network in this approach works independently of the FLS, and there is no parallelism. The co-operative neural fuzzy systems can be categorized into 4 different types:

1) Neural network is used to determine the fuzzy sets. The schematic of this type of co-operative neural fuzzy systems is shown in Fig. 4.

2) Neural network is used to determine the fuzzy rules. The schematic of this type of co-operative neural fuzzy system is shown in Fig. 5.

3) Neural network is used to tune the membership functions of a fuzzy logic system. The schematic of this type of neural fuzzy system is shown in Fig. 6.

4) Neural network is used to determine the fuzzy rule weights. This system will make an adaptive FLS, where the degree of the rule can change when the system changes. The schematic for this type of neural fuzzy system is shown in Fig. 7.

In hybrid neural fuzzy networks, the neural network and fuzzy logic system are used in parallel. NN and FLS are used in parallel. In this structure, the learning algorithms of neural networks are used. The parallelism can be viewed as a mapping from FLS into a NN. Therefore, each phase of the fuzzy system corresponds to a layer of neural network. The most common architecture of a hybrid neural fuzzy system considers a five-layer neural network as shown in Fig. 8.
Layer 1 in Fig. 8 represents the fuzzification, which consists of a set of linguistic rules. Layer 2 is the rule node, where contains one node per fuzzy if/then rule. Layer 3 is the normalization node, where the firing strengths of fuzzy rules are normalized. Layer 4 is the consequent layer, where the values of the consequent are multiplied by the normalized firing strengths. Finally, the layer 5 does the summation job to calculate the output. An example of these systems is ANFIS for which we will bring an example.

IV. NEURO-FUZZY CLASSIFICATION

Fuzzy RuleNet is a neuro-fuzzy approach that is suitable for classification. The architecture is shown in Fig. 9. It is a 3-layer feedforward network, where the hidden nodes represent fuzzy rules. Each hidden node is connected to exactly one of the output nodes, which represent classes. The learning algorithm places overlapping hyperboxes in the input space, to encompass patterns. Over each hyperbox a multidimensional membership function is defined, such that each hyperbox represents a multidimensional fuzzy set, and can be interpreted as a fuzzy rule. We can control the degree of overlapping of the hyperboxes.

To interpret the fuzzy rules, they are projected onto the single dimensions, such that triangular or trapezoidal membership functions are obtained. Thus the learning algorithm creates a rule base and membership functions at the same time.

NEFCLASS (Neuro-fuzzy classification) is a neuro-fuzzy model that is derived from the generic fuzzy perceptron. NEFCLASS is a neuro-fuzzy model based on a generic 3-layer fuzzy perceptron, and it is used for classifying data (patterns). A NEFCLASS system is trained with a set of example patterns, where each pattern belongs to one of a number of distinct classes (crisp classification). A single input pattern is a vector of floating point numbers.

NEFCLASS finds fuzzy rules by scanning the data, and later optimizes these rules by learning the parameters of the fuzzy sets that are used to partition the domain of the input variables (features of the patterns). The idea of the learning algorithm is to create a rule-base first, and then to refine it by modifying the initially given membership functions (usually fuzzy partitions where the membership degrees of each value and up to 1).

Finding for each pattern in the training set, a rule that best classifies it will create the rule base.

If a rule with an identical antecedent is not already in the rule base, it will be added. After all patterns are processed once, the rule base is complete. It is then possible to evaluate the performance of each rule, and delete some of them (if there are too many) to keep only the best rules.

The learning algorithm of the membership functions uses the output error that tells, whether the degree of fulfillment of a rule has to be higher or lower.

This information is used to change the input fuzzy sets by shifting the membership functions, and making their supports larger or smaller. By changing only the fuzzy set that delivered the smallest membership degree for the current pattern, the changes are kept as small as possible. It is easy to define constraints for the learning procedure, e.g. that fuzzy sets must not pass each other, or that they must intersect at 0.5, etc. Constraints like there help to obtain an interpretable rule base, but may cause a loss of performance in classification.

The learning process is visualized in Figure 11. Fig. 11(a) shows the situation after the rule-learning algorithm has terminated. The predefined fuzzy partitioning on both input variables defines a grid in the input space. Overlapping hyper-boxes, where the Cartesian product of the supports of fuzzy sets forms each hyper-box, creates this grid. Each
hyper-box represents the support of an n-dimensional fuzzy set. During rule learning, hyper-boxes are selected due to the distribution of the patterns.

![Image](image1.png)

Figure 11. Visualization of a possible NEFCLASS learning process (a) after rule creation, and (b) after fuzzy set tuning

Each hyper-box is mapped to the class of the pattern which caused its selection (i.e. the rule conclusion is determined). After all patterns are processed, the mapping of hyper-boxes to classes is re-evaluated and changed wherever necessary. After this process, only the “best” hyper-boxes (fuzzy rules) are kept.

There are usually some patterns, which are not classified after rule learning, because their hyper-box (rule) was not included in the set of $k_{\text{max}}$ best rules. There are usually also some misclassifications. It is the task of the fuzzy set learning algorithm to improve this situation. By modifying the membership functions, the predefined grid is distorted. These results of simulation are shown in figure 12. Because the learning algorithm for the fuzzy sets is constrained (e.g. fuzzy set must not pass a neighbor), it is possible, that some changes to the form of the hyper-boxes are not applicable. This is one reason that some classification errors can remain. Another reason can be a too small number of fuzzy rules. This can also lead to undesired forms of membership functions (e.g. too much overlapping). Considering the resulting situation in figure 12, it is probably better to accept four instead of three rules to avoid the extremely wide support of the leftmost fuzzy set over feature x.

From the viewpoint of the NEFCLASS architecture and the flow of data, the fuzzy sets are trained by backpropagation algorithm: the error is propagated from the output units towards the input units and is used to change the membership function parameters, but there is no gradient information involved. The adapt of a NEFCLASS system is restricted, because of the initially given input fuzzy sets, which define the form and maximal number of clusters, and by the constraints that do not admit certain changes in the fuzzy sets.

Other neuro-fuzzy models that are suitable for classification purposes are Fuzzy ART [17] and fuzzy min-max neural networks [18]. The latter one is similar to Fuzzy RuleNet. However, there are differences in the learning strategy and the interpretation of the learning result.

After the learning process the resulting NEFCLASS system can be used for classifying new, previously unknown data.

![Image](image2.png)

Figure 12. Classification results

One advantage of using NEFCLASS for data classification is that the system can be interpreted in form of fuzzy rules like

$$R_x: \text{IF } x_1 \text{ is } A_{j1}^{(1)} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn}^{(n)} \text{ THEN } (x_1, x_2, \ldots, x_n) \in C_j$$

where $A_{ji}^{(i)}$ are linguistic terms (like small, large etc.) represented by fuzzy sets. $x_1, x_2, \ldots, x_n$ are the input features of the pattern x, $C_j$ is a (crisp) subset of $\mathbb{IR}^n$ that represents the class $j$.

V. NEURO-FUZZY FUNCTION APPROXIMATION

In this section we consider the problem of approximating an unknown continuous function by a fuzzy system, where the function is partly specified by a set of data.
sample. This is a supervised learning problem, because the error of the approximation is defined by the difference between the actual output of the fuzzy system, and the target output given in the training data.

One of the first neuro-fuzzy systems for function approximation is the ANFIS model [16]. ANFIS, which stands for Adaptive Network-based Fuzzy Inference System, or semantically equivalently, Adaptive Neuro-Fuzzy Inference System, is referred to as a class of adaptive networks that act as a fundamental framework for adaptive fuzzy inference systems. We will primarily describe the ANFIS architecture and its learning algorithms for the Sugeno fuzzy model, with an application example of trajectory of the non-linear function.

A. ANFIS Architecture

For simplicity, we assume the fuzzy inference system under consideration has two inputs \(x\) and \(y\), and one output \(z\). For a first-order Sugeno fuzzy model, a typical rule set with two fuzzy if-then rules can be expressed as

**Rule 1**: If \(x\) is \(A_1\) and \(y\) is \(B_1\),
then \(f_1 = p_1 x + q_1 y + r_1\).

**Rule 2**: If \(x\) is \(A_2\) and \(y\) is \(B_2\),
then \(f_2 = p_2 x + q_2 y + r_2\).

Figure 13(a) illustrates the reasoning mechanism for this Sugeno model. The corresponding equivalent ANFIS architecture is as shown in Figure 13(b), where nodes of the same layer have similar functions, as described below. (Here we denote the output node \(i\) in layer \(l\) as \(O_{i,l}\).

**Layer 1**: Every node \(i\) in this layer is an adaptive node with a node output defined by
\[
O_{i,1} = \mu_A(x), \quad i = 1, 2,
\]
\[
O_{i,1} = \mu_B(y), \quad i = 3, 4,
\]
where \(x\) (or \(y\)) is the input to the node and \(A_i\) (or \(B_{i-2}\)) is a fuzzy set associated with this node. In other words, outputs of this layer are the membership functions for \(A_i\) and \(B_i\) can be any appropriate parameterized membership functions. For example, \(A_i\) can be characterized by the generalized bell function:
\[
\mu_A(x) = \frac{1}{1 + \left(\frac{x - c_i}{a_i}\right)^b}
\]
where \(\{a_i, b, c_i\}\) is the parameter set. Parameters in this layer are referred to as **premise parameters**.

**Layer 2**: Every node in this layer is a fixed node labeled \(\Pi\), which multiplies the incoming signals and outputs the product. For instance,
\[
O_{i,2} = \omega_{i,1} \mu_A(x) \times \mu_B(y), \quad i = 1, 2.
\]
Each node output represents the firing strength of a rule. (In fact, any other T-norm operators that perform fuzzy AND can be used as the node function in this layer.)

**Layer 3**: Every node in this layer is a fixed node labeled \(\Sigma\). The \(i\)-th node calculates the ratio of the \(i\)-th rule's firing strength to the sum of all rules' firing strengths:
\[
O_{i,3} = \frac{\omega_{i,2}}{\sum_{i} \omega_{i,2}}
\]
For convenience, outputs of this layer will be called **normalized firing strengths**.

**Layer 4**: Every node \(i\) in this layer is an adaptive node with a node function
\[
O_{i,4} = \omega_{i,3} \mu_A(x) \times \mu_B(y) = \omega_{i,3} \omega_{i,2} (p_i x + q_i y + r_i),
\]
where \(\omega_{i,3}\) 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 fixed node labeled \(\Sigma\), which computes the overall output as the summation of all incoming signals:
\[
O_{5,1} = \text{overall output} = \sum_i \omega_{i,4} f_i = \sum_i \omega_i f_i
\]
Thus we have constructed an adaptive network that has exactly the same function as a Sugeno fuzzy model. Note that the structure of this adaptive network is not unique; we can easily combine layers 3 and 4 to obtain an equivalent network with only four layers. Similarly, we can perform weight
normalization at the last layer; Figure 14 illustrates an ANFIS of this type.

Figure 14 Another ANFIS architecture for the two-input two-rule Sugeno fuzzy model.

Figure 15(a) is an ANFIS architecture that is equivalent to a two-input first order Sugeno fuzzy model with nine rules, where each input is assumed to have three associated MF's. Figure 15(b) illustrates how the 2-D space is partitioned into nice overlapping fuzzy regions, each of which is governed by fuzzy if-then rules. In other words, the premise part of a rule defines a fuzzy region, while the consequent part specifies the output within this region.

B. Hybrid Learning Algorithm

From the ANFIS architecture shown in Figure 13(b), we observe that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters. In symbols, the output $f$ in Figure 13(b) can be rewritten as

$$f = \frac{\omega_1}{\omega_1 + \omega_2} f_1 + \frac{\omega_2}{\omega_1 + \omega_2} f_2$$

$$= \omega_1 f_1 + \omega_2 f_2$$

$$= (\omega_1 x) p_1 + (\omega_1 y) q_1 + (\omega_2 x) p_2 + (\omega_2 y) q_2$$

which is linear in the consequent parameters $p_1, q_1, r_1, p_2, q_2, r_2$.

More specifically, in the forward pass of the hybrid learning algorithms, node outputs go forward until layer 4 and the consequent parameters are identified by the least-squares method. In the backward pass, the error signal propagates backward and the premise parameters are updated by gradient descent.

However, in some cases, when the input-output data set is relatively small, these membership functions can be kept fixed throughout the training process, and only consequent parameters are adjusted.

C. Application and Numerical Example

Let us now demonstrate an application of an ANFIS for function approximation. In our example, an ANFIS is used to follow a trajectory of the non-linear function defined by the equation

$$y = \frac{\cos(2 \pi x_1)}{e^{x_2^2}}$$

First, we choose an appropriate architecture for the ANFIS. An ANFIS must have two inputs $x_1$ and $x_2$ and one output $y$.

We decide on the number of membership functions to be assigned to each input by choosing the smallest number of membership function that yield a 'satisfactory' performance. Thus, the experimental study may begin with two membership functions assigned to each input variable.
Let's implement an ANFIS in MATLAB. We will use one of most popular tools that is MatLab Fuzzy Logic Toolbox. The ANFIS training data includes 101 training samples. They are represented by a $101 \times 3$ matrix $[x_1, x_2, y_d]$, where $x_1$ and $x_2$ are input vectors, $y_d$ is a desired output vector. The first input vector $x_1$ starts at 0, increments by 0.1 and ends at 10. The second input vector, $x_2$, is created by taking the cosine of each element of vector $x_1$. Finally, each element of the desired output vector $y_d$ is determined by the function equation.

An actual trajectory of the function and the ANFIS's output after 1 and 100 epochs of training are depicted in Figure 16. Note that Figure 16(a) represents results after the least square estimator identified the rule consequent parameter for the first time. As we can see, the ANFIS's performance is not always adequate even after 100 epochs of training.

We can achieve some improvements in an ANFIS's performance by increasing the number of epochs, but better results are obtained when we assign three membership functions to each input variable. In this case, the ANFIS model will have nine rules, as shown in Figure 17.

Figure 17 shows that the ANFIS's performance improves significantly, and even after one epoch it's output quite accurately resembles the desired trajectory.

The ANFIS has a remarkable ability to generalize and converge rapidly. This is particularly important in on-line learning. As a result, Jang's model and its variants are finding numerous applications, especially in adaptive control.

**VI. SUMMARY AND CONCLUSIONS**

The integration of fuzzy logic systems and neural networks were studied. This integration will allow us overcome the limitations of each individual method. Fuzzy methods used in neural networks (Fuzzy-Neural) and Neural methods used in Fuzzy Systems (Neural-Fuzzy) were explained and the structure of different types of these methods were demonstrated. Finally, some examples were demonstrated to show the usefulness of these methods.

This area has become so wide and is rapidly growing with lots of applications in different engineering systems, because in most real-world engineering problems these tools are of best means of system identification.

**VII. REFERENCES**


