Journal of Artificial Intelligence
ISSN: 2229-3965 & E-ISSN: 2229-3973, Volume 3, Issue 2, 2012, pp.-74-84.
Available online at http://www.bioinfo.in/contents.php?id=71

FUZZY LOGIC AND NEURO-FUZZY MODELING

NIKAM S.R.*, NIKUMBH P.J. AND KULKARNI S.P.

RAIT College of Enggineering, Nerul, Navi Mumbai, India.
*Corresponding Author: Email- sneha.nikam@gmail.com

Received: February 21, 2012; Accepted: March 15, 2012

Abstract- Fuzzy logic and fuzzy systems have recently been receiving a lot of attention, both from the media and scientific community, yet the basic techniques were originally developed in the mid-sixties. Fuzzy logic provides a formalism for implementing expert or heuristic rules on computers, and while this is the main goal in the field of expert or knowledge-based systems, fuzzy systems have had considerably more success and have been sold in automobiles, cameras, washing machines, rice cookers, etc. This report will describe the theory behind basic fuzzy logic and investigate how fuzzy systems work. This leads naturally on to neuro fuzzy systems which attempt to fuse the best points of neural and fuzzy networks into a single system. Throughout this report, the potential limitations of this method will be described as this provides the reader with a greater understanding of how the techniques can be applied.
Keywords- Fuzzy logic, Neural networks, fuzzy modeling, neuro-fuzzy systems, neuro-fuzzy modeling, ANFIS.

Citation: Nikam R.S., Nikumbh P.J. and Kulkarni S.P. (2012) Fuzzy Logic and Neuro-Fuzzy Modeling. Journal of Artificial Intelligence, ISSN: 2229-3965 & E-ISSN: 2229-3973, Volume 3, Issue 2, pp.-74-84.

Introduction

Neuro Fuzzy (NF) computing is a popular framework for solving complex problems. If we have knowledge expressed in linguistic rules, we can build a FIS, and if we have data, or can learn from a simulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of ANN.

Hayashi et al. showed that a feed forward neural network could approximate any fuzzy rule based system and any feed forward neural network may be approximated by a rule based fuzzy inference system . Fusion of Artificial Neural Networks (ANN) and Fuzzy Inference Systems (FIS) have attracted the growing interest of researchers in various scientific and engineering areas due to the growing need of adaptive intelligent systems to solve the real world problems. A neural network learns from scratch by adjusting the interconnections between layers. Fuzzy inference system is a popular computing framework based on the concept of fuzzy set theory, fuzzy *if-then* rules, and fuzzy reasoning. The advantages of a combination of neural networks and fuzzy inference systems are obvious. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore

it is natural to consider building an integrated system combining the concepts. The arrangement of this article is as follows:

In part 2, an introduction to the basic concepts of fuzzy sets, fuzzy reasoning, fuzzy if-then rules are given.

In part 3, Fuzzy inference System is Describe.

In part 4, is devoted to the Neuro-Fuzzy systems.

In part 5, a number of design techniques for fuzzy and neural controllers is described.

In part 6, concludes the paper by pointing current problems and future directions.

Fuzzy Sets, Fuzzy Rules, Fuzzy Reasoning

This section provides a concise introduction to and a summary of the basic concepts central to the study of fuzzy sets.

A. Fuzzy sets

Fuzzy logic starts with the concept of a fuzzy set. A *fuzzy set* is a set without a crisp, clearly defined boundary. It can contain el ments with only a partial degree of membership.

A classical set is a container that wholly includes or wholly excludes any given element. Another version of this law is: "Of any subject one thing must be either asserted or denied"

That is, the transition from "belonging to a set" to "not belonging to a set" is gradual, and this smooth transition is characterized by membership function that give fuzzy sets flexibility in modeling commonly used linguistic expressions.

B. Membership Function

A *membership function* (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the *universe of discourse*, a fancy name for a simple concept.

The output-axis is a number known as the membership value between 0 and 1. The curve is known as a *membership function* and is often given the designation of µ.

The simplest membership functions are formed using straight lines. Of these, the simplest is the *triangular* membership function, and it has the function name trimf. This function is nothing more than a collection of three points forming a triangle. The *trapezoidal* membership function, trapmf, has a flat top and really is just a truncated triangle curve. These straight line membership functions have the advantage of simplicity.

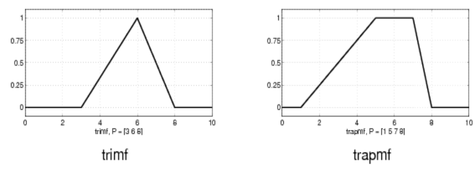

Fig. 2.1- Triangular & Trapezoidal MF

Two membership functions are built on the Gaussian distribution curve: a simple Gaussian curve and a two-sided composite of two different Gaussian curves. The two functions are gaussmf and gauss2mf. The generalized bell membership function is specified by three parameters and has the function name gbellmf.

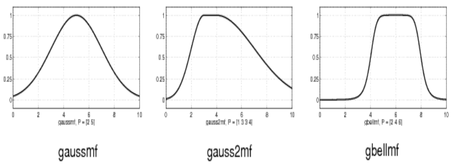

Fig. 2.2- Gaussian & Generalized Bell MF

Also we can define the sigmoidal membership function, which is either open left or right. Asymmetric and closed (i.e. not open to the left or right) membership functions can be synthesized using two sigmoidal functions, so in addition to the basic sigmf, you also have the difference between two sigmoidal functions, dsigmf, and the product of two sigmoidal functions psigmf.

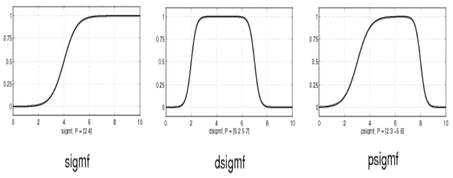

Fig. 2.3- Sigmoidal MF

C. Fuzzy If-Then Rules

A fuzzy if-then rule (fuzzy rule, fzzy implication or fuzzy conditional statement) assumes the form

If x is A then y is B,

Where A and B are linguistic values defined by fuzzy sets on universe of discourse X and Y, respectively. Often "x is A" is called the antecedent or premise while "y is B" is called the consequence or conclusion. Examples of fuzzy if-then rules in our daily linguistic expressions are as follows:

- If the road is slippery the driving is dangerous.
- If a tomato is red then it is ripe.

Before we can employ fuzzy if-then rules to model and analyze a system, we first have to formalize what is meant by the expression "if x is A then y is B", which is sometimes abbreviated as A $\rightarrow$ B. in essence, the expression describes a relation between two variables x and y; this suggests that a fuzzy if-then rule be defined as a binary fuzzy relation R on the product space X × Y. a binary relation R is an extension of the classical Cartesian product, where each element (x,y) € X × Y is associated with a membership grade denoted by $\mu_R(x,y)$.

Interpreting an if-then rule involves distinct parts: first evaluating the antecedent (which involves *fuzzifying* the input and applying any necessary *fuzzy operators*) and second applying that result to the consequent (known as *implication*). In the case of two-valued or binary logic, if-then rules do not present much difficulty. If the premise is true, then the conclusion is true. If you relax the restrictions of two-valued logic and let the antecedent be a fuzzy statement, how does this reflect on the conclusion? The answer is a simple one. if the antecedent is true to some degree of membership, then the consequent is also true to that same degree.

D. Fuzzy Reasoning

Fuzzy reasoning also known as approximate reasoning is an inference procedure used to derive conclusion from a set of fuzzy if-then rules and one or more conditions. Before introducing fuzzy reasoning, we will discuss the compositional rule of inference.

The compositional rule of inference is a generalization of the following notion. Suppose that we have a curve y = f(x) that regulates the relation between x and y. when we are given x=a, then from y=f(x) we can infer that y = b = f(a)which is shown below fig.

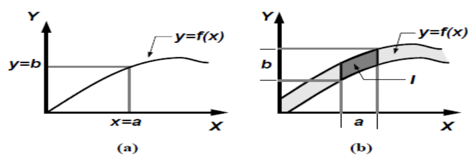

Fig. 2.4- Compositional Rule (a) a & b are points (b) a & b are intervals

Using the compositional rule of inference, we can formulize an inference procedure, called fuzzy reasoning, upon a set of fuzzy if-

then rules. The basic rule of inference in traditional two-valued logic is modus ponens, according to which we can infer the truth of a proposition B from the truth of A and the implication A $\rightarrow$ B. for instance, if A is identified with “the tomato is red” and B with “the tomato is ripe”, then if it is true that “the tomato is red”, it is also true that “the tomato is ripe”.
Premise 1 (fact)- x is A,
Premise 2 (rule)- if x is A then y is B,
Consequence(conclusion)- y is B.
However, in much of human reasoning, modus ponens is employed in an approximate manner.
Using this compositional rule we can perform fuzzy reasoning. The various types of reasoning are:

- Based on max-min composition
- Single rule with single antecedent
- Single rule with two antecedents
- Multiple rules with two antecedents

Fuzzy Inference Systems

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all of the pieces that are described in the previous sections: Membership Functions, Fuzzy set theory, and If-Then Rules and Fuzzy reasoning. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems.
Basically a fuzzy inference system is composed of five functional blocks:

- 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 reasoning mechanism which performs the inference procedure upon the rules and a given condition to derive a reasonable output;
- 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 interface into a crisp output.

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

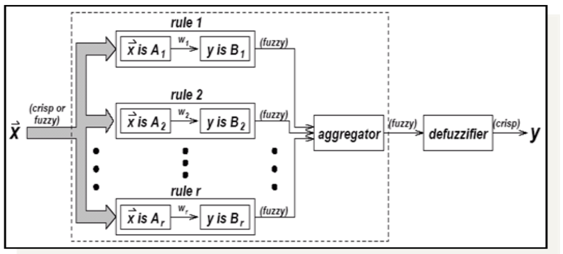

Fig.3.1- Block diagram for fuzzy inference system

The dashed line indicates a basic fuzzy inference system with fuzzy output and the defuzzification block serves the purpose of transforming a fuzzy output into a crisp one. With crisp inputs and outputs, a fuzzy inference system implements a non-linear mapping from its input space to output space. This mapping is done by a no of fuzzy if-then rules, each of which describes the local behavior of the mapping.
Now, we will first introduce three types of fuzzy inference systems that have been widely employed in various applications. The difference between these three fuzzy inference systems lie in the consequents of their fuzzy rules, and their aggregation and defuzzification procedures.

A. Mamdani Fuzzy Model

The Mamdani Fuzzy Model was proposed to control a steam e gine and boiler combination by a set of linguistic control rules. Following fig. shows how a two-rule fuzzy inference system of the Mamdani type derives the overall output z when subjected to two crisp inputs x and y.

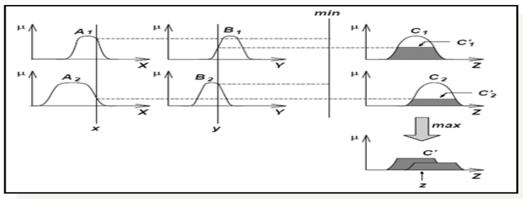

Fig. 3.2- Max-Min composition

If we adopt product and max as our choice for the fuzzy AND and OR operators and use max-product composition instead of max-min composition, then the resulting fuzzy reasoning is shown below:

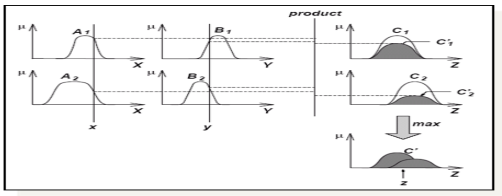

Fig. 3.3- Max-Product composition

Since the plant takes only crisp values as inputs, we have defuzzifier to convert a fuzzy set to a crisp value. The most frequently used defuzzification strategy is the centroid of area, which is define as

$$Z_{COA} = \frac{\int_Z \mu_A(z)z\,dz}{\int_Z \mu_A(z)\,dz}$$

where, $\mu_A(z)$ is the aggregated output MF.
Other defuzzification strategies arise for specific applications, which include bisector of area, mean of maximum, largest of maximum, and smallest of maximum and so on.

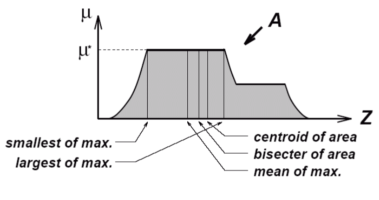

Fig 3.4- Defuzzification Strategies

B. Sugeno Fuzzy Model

The sugeno fuzzy model also known as TSK fuzzy model was proposed to develop a systematic approach to generating fuzzy rules from a given input-output data set. A typical fuzzy rule in a sugeno fuzzy model has the form

If x is A and y is B the z = f(x,y)

Where A and B are fuzzy sets in the antecedent and z = f(x,y) is a crisp function in the consequent. when f(x,y) is a first order polynomial, the resulting fuzzy inference system is called a first order sugeno fuzzy model. When f is constant, then we have zero-order sugeno fuzzy model, which can be viewed as a special case of the mamdani fuzzy inference system.

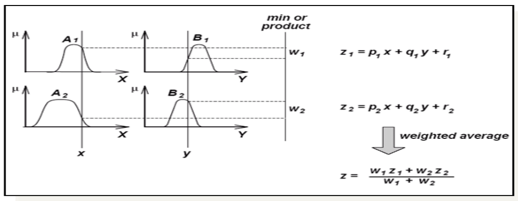
min or product
A_1
B_1
A_2
B_2
w_1
w_2
$z_1 = p_1x + q_1y + r_1$
$z_2 = p_2x + q_2y + r_2$
weighted average
$z = \frac{w_1z_1 + w_2z_2}{w_1 + w_2}$

Fig. 3.5- First Order Sugeno Fuzzy Model

figure 3.5 shows the reasoning procedure for a first-order sugeno fuzzy model. The aggregator and defuzzifier blocks in fig 3.1 are replaced by the operation of weighted average, thus avoiding the time consuming procedure of defuzzification. Sometimes the weighted average operator is replaced with the weighted sum operator. However, this simplification could lead to the loss of MF linguistic meanings unless the sum of firing strength is close to unity.

C. Tsukamoto Fuzzy Model

In the Tsukamoto fuzzy model, the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonical MF as shown in fig. The inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's output.

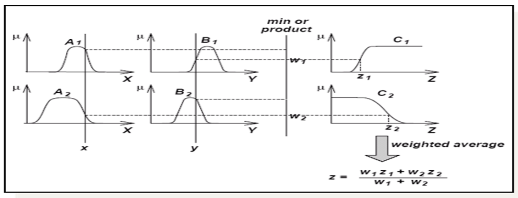

Fig. 3.6- Tsukamoto Fuzzy Model

Since each rule infers a crisp output, the Tsukamoto fuzzy model aggregates each rule's output by the method of weighted average and also avoids the time consuming process of defuzzification.

Neuro-Fuzzy Systems

Neuro Fuzzy (NF) computing is a popular framework for solving complex problems. If we have knowledge expressed in linguistic rules, we can build a FIS, and if we have data, or can learn from a simulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of ANN.

The process for constructing a fuzzy inference system is usually called fuzzy modeling, which has following features:

- Due to the rule structure of a fuzzy inference system, it is easy to incorporate human expertise about the target system directly into the modeling process. Fuzzy modeling takes advantage of domain knowledge that might not be easily or directly employed in other modeling approaches.
- When the input-output data of a system to be modeled is available, conventional system identification techniques can be used for fuzzy modeling.

The term neuro-fuzzy modeling refers to the way of applying various learning techniques developed in the neural network literature to fuzzy inference systems. Now, we present cooperative NF system and concurrent NF system followed by the different fused NF models.

A. Cooperative And Concurrent Neuro-Fuzzy Systems

In the simplest way, a cooperative model [1][2][3]can be considered as a preprocessor wherein artificial neural network (ANN) learning mechanism determines the fuzzy inference system (FIS) membership functions or fuzzy rules from the training data. Once the FIS parameters are determined, ANN goes to the background. The rule based is usually determined by a clustering approach or fuzzy clustering algorithms. Membership functions are usually approximated by neural network from the training data.

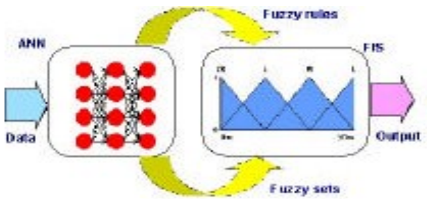

Fig. 4.1- Cooperative NF model

In a concurrent model[1][2], neural network assists the fuzzy system continuously (or vice versa) to determine the required parameters especially if the input variables of the controller cannot be measured directly. Such combinations do not optimize the fuzzy system but only aids to improve the performance of the overall system. Learning takes place only in the neural network and the fuzzy system remains unchanged during this phase. In some cases the fuzzy outputs might not be directly applicable to the process. In that case neural network can act as a postprocessor of fuzzy outputs. Figure 4.2 depicts a concurrent neuro-fuzzy model where in the input data is fed to a neural network and the output of the neural network is further processed by the fuzzy system.

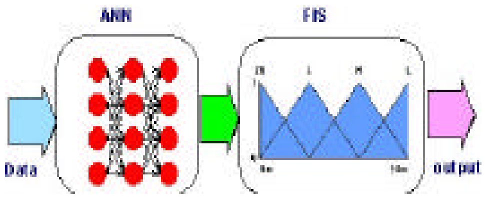

Fig. 4.2- Concurrent NF model

B. Fused Neuro-Fuzzy Systems

In an integrated model[1][2][3], neural network learning algorithms are used to determine the parameters of fuzzy inference systems. Integrated neuro-fuzzy systems share data structures and knowledge representations. A fuzzy inference system can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of *if-then* rules and corresponding membership functions depends heavily on the a priori knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a fuzzy inference system. There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate. On the other hand, neural network learning mechanism does not rely on human expertise. However, in reality, the a priori knowledge is usually obtained from human experts, it is most appropriate to express the knowledge as a set of fuzzy if-then rules, and it is very difficult to encode into a neural network.

Table 4.1- Comparison between neural networks and fuzzy inference systems

Artificial Neural Network	Fuzzy Inference System
Difficult to use prior rule knowledge	Prior rule-base can be incorporated
Learning from scratch	Cannot learn (linguistic knowledge)
Black box	Interpretable (if-then rules)
Complicated learning algorithms	Simple interpretation and implementation
Difficult to extract knowledge	Knowledge must be available

Table 4.1 summarizes the comparison between neural networks and fuzzy inference system. To a large extent, the drawbacks pertaining to these two approaches seem complementary. Therefore, it seems natural to consider building an integrated system combining the concepts of FIS and ANN modeling..
This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. Some of the major woks in this area are GARIC, FALCON , ANFIS , NEFCON , FUN , SONFIN ,FINEST , EFuNN , dmEFuNN, evolutionary design of neuro fuzzy systems, and many others.

C. Fuzzy Adaptive learning Control Network (FALCON)

FALCON [1][2][9] has a five-layered architecture and implements a Mamdani type FIS. There are two linguistic nodes for each output variable. One is for training data (desired output) and the other is for the actual output of FALCON. The first hidden layer is responsible for the fuzzification of each input variable. Each node can be a single node representing a simple membership function (MF) or composed of multilayer nodes that compute a complex MF. The Second hidden layer defines the preconditions of the rule followed by rule consequents in the third hidden layer. FALCON uses a hybrid-learning algorithm comprising of unsupervised learning and a gradient descent learning to optimally adjust the parameters to produce the desired outputs. The hybrid learning occurs in two different phases. In the initial phase, the centers and width of the membership functions are determined by self-organized learning techniques analogous to statistical clustering techniques. Once the initial parameters are determined, it is easy to formulate the rule antecedents. A competitive learning algorithm is used to determine the correct rule consequent links of each rule node. After the fuzzy rule base is established, the whole network structure is established. The network then enters the second learning phase to adjust the parameters of the (input and output) membership functions optimally. The back propagation algorithm is used for the supervised learning. Hence FALCON algorithm provides a framework for structure and parameter adaptation for designing neuro-fuzzy systems.

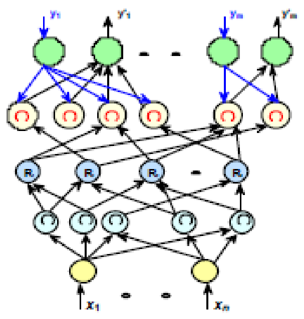

Fig. 4.3- Architecture of FALCON

D. Generalized Approximate Reasoning based Intelligent Control (GARIC)

GARIC [1][2][9] is an extended version of Berenji's Approximate Reasoning based Intelligent Control (ARIC) that implements a fuzzy controller by using several specialized feed forward neural networks. Like ARIC, it consists of an Action state Evaluation Network (AEN) and an Action Selection Network (ASN). Architecture of the GARICASN is depicted in Fig.4.4. ASN of GARIC is feedforward network with ASN of GARIC is feed forward network with five layers .

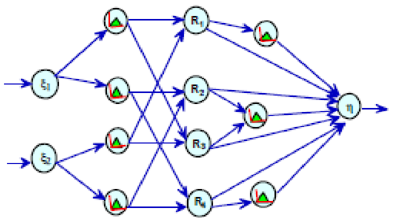

Fig. 4.4- ASN of GARIC

The first hidden layer stores the linguistic values of all the input variables. Each input unit is only connected to those units of the first hidden layer, which represent its associated linguistic values. The second hidden layer represents the fuzzy rules nodes, which determine the degree of fulfillment of a rule using a *softmin* operation. The third hidden layer represents the linguistic values of the control output variable *η*. Conclusions of the rule are computed depending on the strength of the rule antecedents computed by the rule node layer. GARIC makes use of local mean-of-maximum method for computing the rule outputs. This method needs a crisp output value from each rule. Therefore, the conclusions must be defuzzified before they are accumulated to the final output value of the controller. GARIC uses a mixture of gradient descent and reinforcement learning to fine-tune the node parameters. The hybrid learning stops if the output of the AEN ceases to change. The relatively complex learning procedure and the architecture of GARIC can be seen as a main disadvantage of GARIC.

E. Neuro-Fuzzy Control (NEFCON)

The learning algorithm defined for NEFCON[1][2][9] is able to learn fuzzy sets as well as fuzzy rules implementing a Mamdani type FIS [1][2]. This method can be considered as an extension to GARIC that also use reinforcement learning but need a previously defined rule base. Figure 4.5 illustrates the basic NEFCON architecture with 2 inputs and five fuzzy rules [1][2]. The inner nodes *R*1*, . . . , R*5 represent the rules, the nodes *ξ*1*, ξ*2, and *η* the input and output values, and *μr, Vr* the fuzzy sets describing the antecedents and consequents. In contrast to neural networks, the connections in NEFCON are weighted with fuzzy sets instead of real numbers. Rules with the same antecedent use so-called shared weights, which are represented by ellipses drawn around the connections as shown in the figure 4.5. They ensure the integrity of the rule base. The knowledge base of the fuzzy system is implicitly given by the network structure. The input units assume the task of fuzzification interface, the inference logic is represented by the propagation functions, and the output unit is the defuzzification interface. The learning process of the NEFCON model can be divided into two main phases. Incremental rule learning is used when the correct output is not known and rules are created based on estimated output values. As the learning progresses, more rules are added according to the requirement. For decremental rule learning, initially rules are created due to fuzzy partitions of process variables and unnecessary rules are eliminated in the course of learning. Decremental rule learning is less efficient compared to incremental approach.

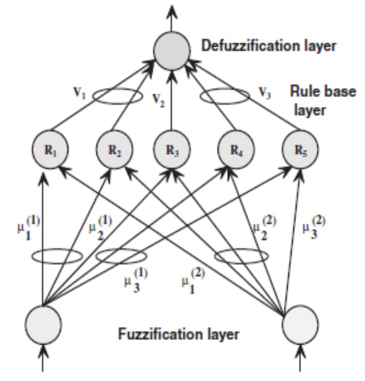

Fig. 4.5- Architecture of NEFCON

Due to the complexity of the calculations required, the decremental learning rule can only be used, if there are only a few input variables with not too many fuzzy sets. For larger systems, the incremental learning rule will be optimal. Prior knowledge whenever available could be incorporated to reduce the complexity of the learning. Membership functions of the rule base are modified according to the Fuzzy Error Back propagation (FEBP) algorithm. The FEBP algorithm can adapt the membership functions, and can be applied only if there is already a rule base of fuzzy rules. The idea of the learning algorithm is identical: increase the influence of a rule if its action goes in the right direction (rewarding), and decrease its influence if a rule behaves counter productively (punishing). If there is absolutely no knowledge about initial membership function, a uniform fuzzy partition of the variables should be used.

F. Fuzzy Inference Environment Software with Tuning (FINEST)

FINEST[1][2][9] is designed to tune the fuzzy inference itself. FINEST is capable of two kinds of tuning process, the tuning of fuzzy predicates, combination functions and the tuning of an implication function [1]. The three important features of the system are:

- The generalized modus ponens is improved in the following four ways:

1. aggregation operators that have synergy and cancellation nature
2. a parameterized implication function
3. a combination function, which can reduce fuzziness
4. backward chaining based on generalized modus ponens.

- Aggregation operators with synergy and cancellation nature are defined using some parameters, indicating the strength of the synergic affect, the area influenced by the effect, etc., and the tuning mechanism is designed to tune these parameters also tune the implication function and combination function.
- The software environment and the algorithms are designed for carrying out forward and backward chaining based on the improved generalized modus ponens and for tuning various parameters of a system.

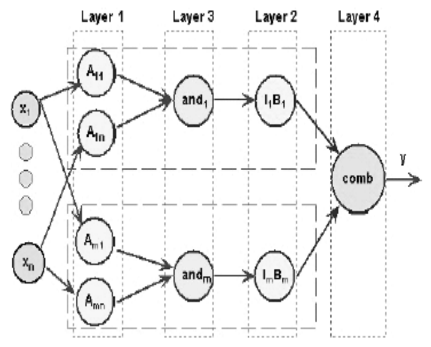

Fig. 4.6- Architecture of FINEST

FINEST make use of a back propagation algorithm for the fine-tuning of the parameters. Figure4.6 shows the layered architecture of FINEST and the calculation process of the fuzzy inference. The input values (*xi*) are the facts and the output value (*y*) is the conclusion of the fuzzy inference. Layer 1 is a fuzzification layer and layer 2 aggregates the truth-values of the conditions of Rule *i*. Layer 3 deduces the conclusion from Rule *I* and the combination of all the rules is done in Layer 4. Referring to Fig. 4.6, the function *andi*, *Ii* and *comb* respectively represent the function characterizing the aggregation operator of rule *i*, the implication function of rule *i*, and the global combination function. The functions *andi*, *Ii*, *comb* and membership functions of each fuzzy predicate are defined with some parameters. Back propagation method is used to tune the network parameters. It is possible to tune any parameter, which appears in the nodes of the network representing the calculation process of the fuzzy data if the derivative function with respect to the parameters is given. Thus, FINEST framework provides a mechanism based on the improved generalized modus ponens for fine tuning of fuzzy predicates and combination functions and tuning of the implication function.

G. Self Constructing Neural Fuzzy Inference Network (SONFIN)

SONFIN [1][2][9] implements a Takagi-Sugeno type fuzzy inference system. Fuzzy rules are created and adapted as online learning proceeds via a simultaneous structure and parameter identification. In the structure identification of the precondition part, the input space is partitioned in a flexible way according to an aligned clustering based algorithm. As to the structure identification of the consequent part, only a singleton value selected by a clustering method is assigned to each rule initially. Afterwards, some additional significant terms (input variables) selected via a projection-based correlation measure for each rule will be added to the consequent part (forming a linear equation of input variables) incrementally as learning proceeds. For parameter identification, the consequent parameters are tuned optimally by either Least Mean Squares [LMS] or Recursive Least Squares [RLS] algorithms and the precondition parameters are tuned by back propagation algorithm. To enhance knowledge representation ability of SONFIN, a linear transformation for each input variable can be incorporated into the network so that much fewer rules are needed or higher accuracy can be achieved. Proper linear transformations are also learned dynamically in the parameter identification phase of SONFIN.

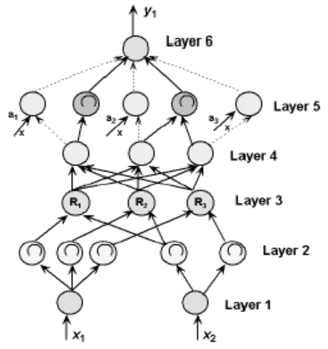

Fig. 4.7- illustrates the 6-layer structure of SONFIN

and the parameters in the precondition part are adjusted by the backpropagation algorithm. SONFIN can be used for normal operation at anytime during the learning process without repeated training on the input-output pattern when online operation is required. In SONFIN rule base is dynamically created as the learning progresses by performing the following learning processes:

H. Fuzzy Net (FUN)

In FUN[1][2][9] in order to enable an unequivocal translation of fuzzy rules and membership functions into the network, special neurons have been defined, through their activation functions, can evaluate logic expressions. The network consists of an input, an output and three hidden layers. The neurons of each layer have different activation functions representing the different stages in the calculation of fuzzy inference. The activation function can be individually chosen for problems. The network is initialized with a fuzzy rule base and the corresponding membership functions. Figure 4.8 illustrates the FUN network. The input variables are stored in the input neurons. The neurons in the first hidden layer contain the membership functions and this performs a fuzzification of the input values. In the second hidden layer, the conjunctions (fuzzy-AND) are calculated. Membership functions of the output variables are stored in the third hidden layer. Their activation function is a fuzzy-OR. Finally, the output neurons contain the output variables and have a defuzzification activation function.

The rules and the membership functions are used to construct an initial FUN network. The rule base can then be optimized by changing the structure of the net or the data in the neurons. To learn the rules, the connections between the rules and the fuzzy values are changed. To learn the membership functions, the data of the nodes in the first and three hidden layers are changed. FUN can be trained with the standard neural network training strategies such as reinforcement or supervised learning.

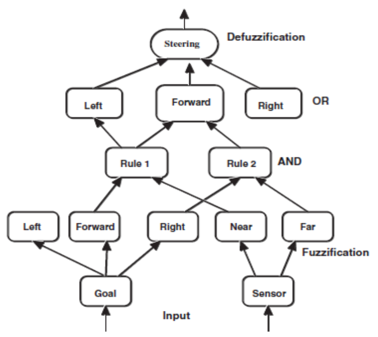

Fig. 4.8- Architecture of the FUN showing the implementation of a sample rule

I. Evolving Fuzzy Neural Networks (EFuNN)

EFuNNs [1][2][9] and dmEFuNNs [1][2][9] are based on the ECOS (Evolving Connectionist Systems) framework for adaptive intelligent systems formed because of evolution and incremental, hybrid (supervised/unsupervised), online learning. They can accommodate new input data, including new features, new classes, etc. through local element tuning. In EFuNNs all nodes are created during learning. EFuNN has a five-layer architecture as shown in Figure 4.9. The input layer is a buffer layer representing the input variables. The second layer of nodes represents fuzzy quantification of each input variable space. Each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantization of this variable. The nodes representing membership functions (triangular, Gaussian, etc) can be modified during learning. The third layer contains rule nodes that evolve through hybrid supervised/unsupervised learning.

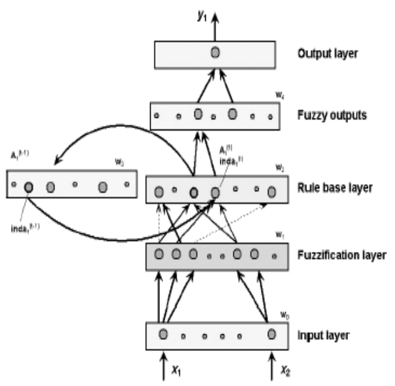

Fig. 4.9- Architecture of EFuNN

The rule nodes represent prototypes of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node *r* is defined by two vectors of connection weights: *W*1(*r*) and *W*2(*r*), the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the input problem space. The fourth layer of neurons represents fuzzy quantification for the output variables. The fifth layer represents the real values for the output variables. In the case of "one-of-n" EFuNNs, the maximum activation of the rule node is propagated to the next level. In the case of "*many-of-n*" mode, all the activation values of rule nodes that are above an activation threshold are propagated further in the connectionist structure.

J. Dynamic Evolving Fuzzy Neural Networks (dmEFuNNs)

Dynamic Evolving Fuzzy Neural Networks (dmEFuNN) model[1] [2][9] is developed with the idea that not just the winning rule node's activation is propagated but a group of rule nodes is dynamically selected for every new input vector and their activation values are used to calculate the dynamical parameters of the output function. While EFuNN make use of the weighted fuzzy rules of Mamdani type, dmEFuNN uses the Takagi-Sugeno fuzzy rules. The architecture is depicted in Figure 4.10

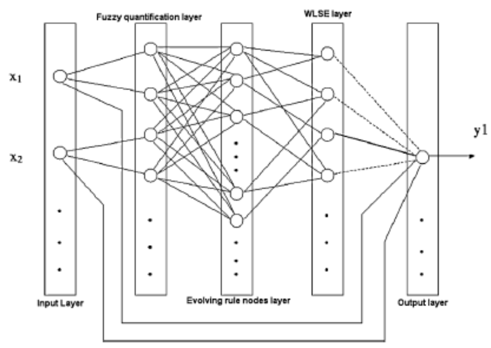

Fig. 4.10- Architecture of dmEFuNN

The first, second and third layers of dmEFuNN have exactly the same structures and functions as the EFuNN. The fourth layer, the fuzzy inference layer, selects *m* rule nodes from the third layer which have the closest fuzzy normalized local distance to the fuzzy input vector, and then, a Takagi Sugeno fuzzy rule will be formed using the weighted least square estimator. The last layer calculates the output of dmEFuNN. The number m of activated nodes used to calculate the output values for a dmEFuNN is not less than the number of the input nodes plus one. Like the EFuNNs, the dmEFuNNs can be used for both offline learning and online learning thus optimizing global generalization error, or a local generalization error. In dmEFuNNs, for a new input vector (for which the output vector is not known), a subspace consisted of *m* rule nodes are found and a first order Takagi Sugeno fuzzy rule is formed using the least square estimator method. This rule is used to calculate the dmEFuNN output value. In this way, a dmEFuNN acts as a universal function approximator using *m* linear functions in a small *m* dimensional node subspace. The accuracy of approximation depends on the size of the node subspaces, the smaller the subspace is, the higher the accuracy. It means that if there are sufficient training data vectors and sufficient rule nodes are created, a satisfying accuracy can be obtained.

K. Evolutionary and Neural Learning of Fuzzy Inference System (EvoNF)

In an integrated neuro-fuzzy model there is no guarantee that the neural network learning algorithm converges and the tuning of fuzzy inference system will be successful. Natural intelligence is a product of evolution. Therefore, by mimicking biological evolution, we could also simulate high-level intelligence. Evolutionary computation works by simulating a population of individuals, evaluating their performance, and evolving the population a number of times until the required solution is obtained. The drawbacks pertaining to neural networks and fuzzy inference systems seem complementary and evolutionary computation could be used to optimize the integration to produce the best possible synergetic behavior to form a single system. Adaptation of fuzzy inference systems using evolutionary computation techniques has been widely explored. EvoNF is an adaptive framework based on evolutionary computation and neural learning wherein the membership functions, rule base and fuzzy operators are adapted according to the problem. The evolutionary search of MFs, rule base, fuzzy operators etc. would progress on different time scales to adapt the fuzzy inference system according to the problem environment. Membership functions and fuzzy operators would be further fine-tuned using a neural learning technique. Optimal neural learning parameters will be decided during the evolutionary search process.

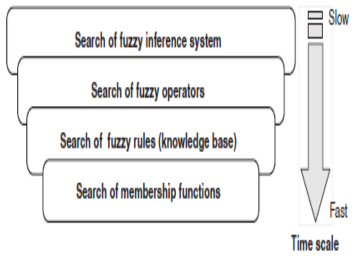
Search of fuzzy inference system
Search of fuzzy operators
Search of fuzzy rules (knowledge base)
Search of membership functions
Slow
Fast
Time scale

Fig. 4.11- Interaction of evolutionary search mechanisms in the adaptation of fuzzy inference system

Figure 4.11 illustrates the general interaction mechanism of the EvoNF framework with the evolutionary search of fuzzy inference system (Mamdani, Takagi -Sugeno etc.) evolving at the highest level on the slowest time scale. For each evolutionary search of fuzzy operators (best combination of T-norm and T-conorm, defuzzification strategy etc), the search for the fuzzy rule base progresses at a faster time scale in an environment decided by the problem. In a similar manner, evolutionary search of membership functions proceeds at a faster time scale (for every rule base) in the environment decided by the problem. Hierarchy of the different adaptation procedures will rely on the prior knowledge.

L. Adaptive Network Based Fuzzy Inference System (ANFIS)

In this section we will discuss the architecture and learning procedure of the adaptive network which is in fact a superset of all kinds of feed forward neural networks with supervised learning capability. As its name implies, adaptive network structure con-

sisting of nodes and directional links though which the nodes are connected. Also, part or all of the nodes are adaptive, which means each output of these nodes depends on the parameters of this node, and learning rule specifies how these parameters should be changed to minimize a prescribed error measure.
Since the basic learning rule is based the gradient method which s 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.

ANFIS Architecture
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 feed-forward type. Due to this restrictions, the adaptive networks applications are immediate and immense in various areas.in this section, we describe a class of adaptive network which are functionally equivalent to fuzzy inference systems.
For simplicity, 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 type:
Rule 1: If x is A1 and y is B1, then f1 = p1x+q1y + r1
Rule 2: If x is A2 and y is B2, then f2 = p2x + q2y +r2.
Then type-3 fuzzy reasoning is given in belo fig 4.

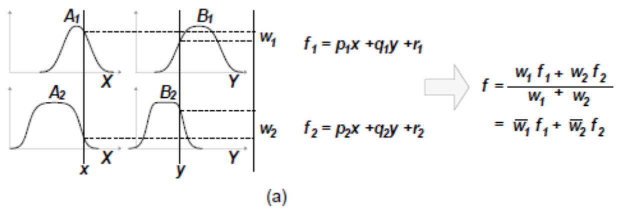

Fig. 4.13- ANFIS- type 3

And equivalent ANFIS architecture is,

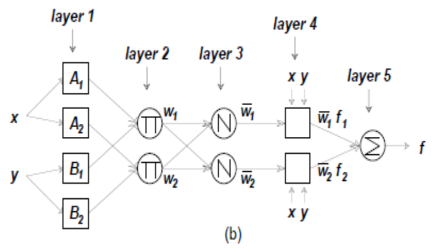

Fig. 4.14- Equivalent ANFIS architecture

Layer 1 Every node I in this layer is a square node with a node function
Where x is the input to node I, and Ai is the linguistic label (small,large, etc) associated with this node function. In other words, Oi1 is the membership fuction of Ai and it specifies the degree to which the given x satisfies the quantifier Ai. Usually μA (x) is choose as bell-shaped with maximum equal to 1 and minimum equal to 0, such as,
Where {ai, bi, ci} is parameter set. As the values of these parameters change, the bell-shaped fuctions vary accordingly. Parameters in this layer are reffered to as premise parameters.

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

Layer 2 Every node in this layer is a circle node label as II which multiplies the incoming signals and sends the product out.
i=1,2
Each node output represents the

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \qquad \mu_{A_i}(x) = \exp\left\{-\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}\right\}$$

firing strength of a rule. Also T-norm operators that perform generalized AND can be used as a node function.
Layer 3 Every node in this layer is a circle node label N. the i-th node calculates the ratio of the i-th rule's firing strength to the sum of all rules firing strength.

$$\omega_i = \mu_{A_i}(x) \times \mu_{B_i}(y)$$

Outputs of this layer $\varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}, i = 1,2$ is called as normalized firing strength.
Layer 4 Every node I in this layer is a square node with a node function

Where wi is the output of layer 3 and {pi, qi, ri} is the parameter set. Parameters in this layer will be referred as consequent parameters.

$$O_i^4 = \varpi_i f_i = \varpi_i (p_i x + q_i y + r_i)$$

Layer 5 the single node in this layer is a circle node labeled ∑ that computs the overall output as summation of all incoming signals

$$O_1^5 = overalloutput = \sum_i \varpi_i f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i}$$

Thus we have constructed adaptive network which is functionally equivalent to type-3 fuzzy inference system. For type-1 fuzzy inference systems the extension is quite straightforward and type-1 ANFIS is shown in below fig.
Where the output of each rule is induced jointly by the output membership function and the firing strength. For type-2 fuzzy inference systems, if we replace the centroid defuzzification operator with a discrete version which calculates the centroid of area, then type-3 ANFIS can still be constructed accordingly.

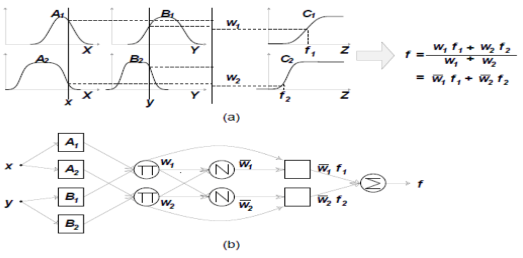

Fig 4.15- (a)Type 1 Fuzzy Reasoning (b) Equivalent ANFIS-type1

Below figure 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 subspaces, while the consequent part specifies the output within this fuzzy subspace.

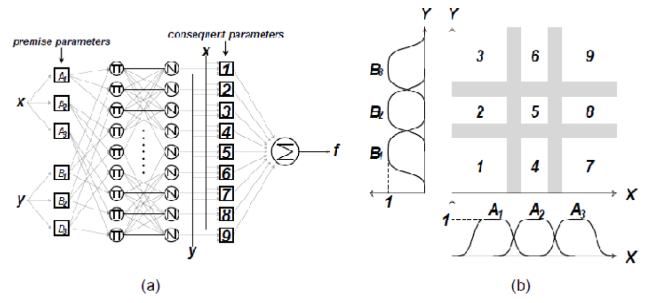

Fig. 4.16- (a) ANFIS type-3 with 2 input and 9 rules (b) Corresponding fuzzy Subspaces

Hybrid Learning Algorithm

From the type-3 anfis architecture it is observed that given values of premise parameters, the output can be expressed as a linear combinations of the consequent parameters. The output f in fig. 4 can be rewritten as,

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

=

= $(\varpi_1 x) p_1 + (\varpi_1 y) q_1 + (\varpi_1) r_1 + (\varpi_2 x) p_2 + (\varpi_2 y) q_2 + (\varpi_2) r_2$

Which is linear in consequent parameters (p1, q1, r1, p2, q2, and r2).

S = set of total parame- ters

S1 = set of premise pa- $\varpi_1 f_1 + \varpi_2 f_2$ rameters

S2 = set of consequent parameters

In forward pass of 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. Below table summarizes the activities in each pass.

Table 4.2- Summarizing the activities in each pass

	Forward pass	Backward pass
Premise parameters	Fixed	Gradient descent
Consequent parameters	Least squares estimates	Fixed
Signals	Node outputs	Error rates

The consequent parameters identified are optimal under the condition that premise parameters are fixed. Accordingly the hybrid approach is much faster than the strict gradient descent.

It should be noted that computation complexity of the least squares estimate is higher than that of the gradient descent. There are four methods to update the parameters,

i. Gradient descent only:- all parameters are updated by the gradient descent.
ii. 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 gradient descent takes over to update all parameters.
iii. Gradient descent and LSE:- this is proposed hybrid learning rule.
iv. Sequential LSE only:- the ANFIS is linearized w.r.t all parameters and the extended kalman filter algo is employed to update all parameters.

The choice of above methods should be based on the trade-off between computation complexity and resulting performance.

Conclusion and Future scope

We presented the different ways to learn fuzzy inference systems using neural network learning techniques. As a guideline, for neurofuzzy systems to be highly intelligent some of the major requirements are fast learning (memory based - efficient storage and retrieval capacities), on-line adaptability (accommodating new features like inputs, outputs, nodes, connections etc), achieve a global error rate and computationally inexpensive. The data acquisition and preprocessing training data is also quite important for the success of neuro-fuzzy systems. Many neuro-fuzzy models use supervised/unsupervised techniques to learn the different parameters of the inference system. The success of the learning process is not guaranteed, as the designed model might not be optimal. Empirical research has shown that gradient descent technique (most commonly used supervised learning algorithm) is trapped in local optima especially when the error surface is complicated. Global optimization procedures like evolutionary algorithms, simulated annealing, tabu search etc. might be useful for adaptive evolution of fuzzy if-then rules, shape and quantity of membership functions, fuzzy operators and other node functions, to prevent the network parameters being trapped in local optima due to reliance on gradient information by most of the supervised learning techniques. Sugeno-type fuzzy systems are high performers (less RMSE) but often requires complicated learning procedures and computational expensive. However, Mamdani-type fuzzy systems can be modeled using faster heuristics but with a compromise on the performance (accuracy). Hence there is always a compromise between performance and computational time.

ANFIS implements a Takagi-Sugeno fuzzy system and applies a mixture of back propagation and least mean squares procedure to train the system. The adaptation process is only concerned with parameter level adaptation within fixed structures. For large scale problems, it will be too much complicated to determine the optimal premise consequent structures, rule numbers etc. the structure of ANFIS ensures that each linguistic term is represented by only one fuzzy set. The learning procedure of ANFIS does not provide the means to apply constraints that restrict the kind of modification applied to membership functions. Due to the high flexibility of adaptive networks, the anfis can have number of variants, for instance, the membership functions can be changed to L-R representation which could be asymmetric, also we can replace Π nodes in layer 2 with parameterized T-norm and 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 proposed other adaptive fuzzy models for data classification and feature extraction purposes.

FUN system is initialized by specifying a fixed number of rules and a fixed number of initial fuzzy sets for each variable and there after uses a stochastic procedure that randomly changes parameters of membership functions and connections within the network structure. The learning process is driven by a cost function, which is evaluated after random modification.

NEFCON makes use of an incremental or decremental learning algorithm for learning the rule base and back propagation algorithm for learning the fuzzy sets. NEFCON system is capable of incorporating prior knowledge as well as learning from scratch. The performance of the system will very much depend on heuristic factors like learning rate, error measure etc.

FINEST provides a mechanism based on the improved generalized modus ponens for fine tuning of fuzzy predicates & combination functions and tuning of an implication function. FINEST uses a gradient descent technique to tune the various parameters. Parameterization of the inference procedure is very much essential for proper application of the tuning algorithm.

SONFIN learns from scratch and the rules are created and adapted as online learning proceeds via simultaneous structure and parameter identification. As the learning proceeds, rules will get modified incrementally.

References

[1] Ajith Abraham (2001) *Sixth international work conference on Artificial and Natural Neural Networks*, IWANN, Granada, 269-276.

[2] Ajith Abraham and Baikunth Nath (2000) *School of computing & information technology*, Monash University, Australia, technical report series, 1-55.

[3] Abraham A. (2005) *Computer Science Department*, Oklahoma State University.

[4] Heikki Koivo (2000) *ANFIS (Adaptive Neuro-Fuzzy Inference System)*.

[5] Jang R. (1992) *Neuro-Fuzzy Modelling: Architectures, Analyses and Applications*.

[6] Jyh-Shing Roger Jang (1993) *IEEE transaction on systems, Man, and Cybernetics*, 23(03):665-685.

[7] Jyh-Shing Roger Jang and Chuen-Tsai Sun (1995) *Proceedings of IEEE*, 83, 378-406.

[8] Jan Jantzen (1995) *An computational approach to intelligence*.

[9] Fernando J.V., Morgado Dias, Alexandre Mota, (2004) *5th WSEAS NNA International Conference on Neural Networks and Applications*, Udine, Italia.