3

Adaptation of Fuzzy Inference System Using Neural Learning

A. Abraham

Computer Science Department, Oklahoma State University, USA
ajith.abraham@ieee.org, http://ajith.softcomputing.net

The integration of neural networks and fuzzy inference systems could be formulated into three main categories: cooperative, concurrent and integrated neuro-fuzzy models. We present three different types of cooperative neuro-fuzzy models namely fuzzy associative memories, fuzzy rule extraction using self-organizing maps and systems capable of learning fuzzy set parameters. Different Mamdani and Takagi-Sugeno type integrated neuro-fuzzy systems are further introduced with a focus on some of the salient features and advantages of the different types of integrated neuro-fuzzy models that have been evolved during the last decade. Some discussions and conclusions are also provided towards the end of the chapter.

3.1 Introduction

Hayashi et al. [21] showed that a feedforward neural network could approximate any fuzzy rule based system and any feedforward neural network may be approximated by a rule based fuzzy inference system [31]. 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 [5, 6, 10, 11, 12, 13, 17, 19, 20, 22, 33, 37]. 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 [12, 32]. 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 [37]. While the learning capability is an advantage from the viewpoint of fuzzy inference system, the automatic formation of linguistic rule base will be advantage from the viewpoint of neural network. There are

A. Abraham: *Adaptation of Fuzzy Inference System Using Neural Learning*, StudFuzz **181**, 53–83 (2005)
www.springerlink.com
© Springer-Verlag Berlin Heidelberg 2005

several works related to the integration of neural networks and fuzzy inference systems [1, 2, 3, 15, 17, 23, 28, 30, 32, 34, 43, 45, 49, 52].

3.2 Cooperative Neuro-Fuzzy Systems

In the simplest way, a cooperative model 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. Fuzzy Associative Memories (FAM) by Kosko [29], fuzzy rule extraction using self organizing maps by Pedrycz et al. [46] and the systems capable of learning of fuzzy set parameters by Nomura et al. [44] are some good examples of cooperative neuro-fuzzy systems.

3.2.1 Fuzzy Associative Memories

Kosko interprets a fuzzy rule as an association between antecedent and consequent parts [29]. If a fuzzy set is seen as a point in the unit hypercube and rules are associations, then it is possible to use neural associative memories to store fuzzy rules. A neural associative memory can be represented by its connection matrix. Associative recall is equivalent to multiplying a key factor with this matrix. The weights store the correlations between the features of the key k and the information part i. Due to the restricted capacity of associative memories and because of the combination of multiple connection matrices into a single matrix is not recommended due to severe loss of information, it is necessary to store each fuzzy rule in a single FAM. Rules with n conjunctively combined variables in their antecedents can be represented by n FAMs, where each stores a single rule. The FAMs are completed by aggregating all the individual outputs (maximum operator in the case of Mamdani fuzzy system) and a defuzzification component.

Learning could be incorporated in FAM, as learning the weights associated with FAMs output or to create FAMs completely by learning. A neural network-learning algorithm determines the rule weights for the fuzzy rules. Such factors are often interpreted as the influence of a rule and are multiplied with the rule outputs. Rule weights can be replaced equivalently by modifying the membership functions. However, this could result in misinterpretation of fuzzy sets and identical linguistic values might be represented differently in different rules. Kosko suggests a form of adaptive vector quantization technique to learn the FAMs. This approach is termed as differential competitive learning and is very similar to the learning in self-organizing maps.

Figure 3.1 depicts a cooperative neuro-fuzzy model where the neural network learning mechanism is used to determine the fuzzy rules, parameters of fuzzy sets, rule weights etc. Kosko's adaptive FAM is a cooperative neuro-fuzzy model because it uses a learning technique to determine the rules and

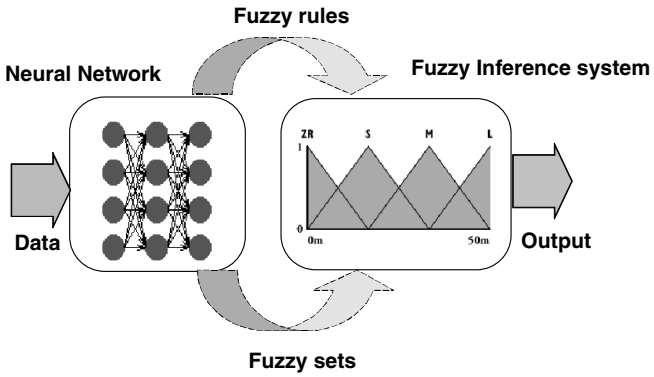

Fig. 3.1. Cooperative neuro-fuzzy model

its weights. The main disadvantage of FAM is the weighting of rules. Just because certain rules, does not have much influence does not mean that they are very unimportant. Hence, the reliability of FAMs for certain applications is questionable. Due to implementation simplicity, FAMs are used in many applications.

3.2.2 Fuzzy Rule Extraction Using Self Organizing Maps

Pedryz et al. [46] used self-organizing maps with a planar competition layer to cluster training data, and they provide means to interpret the learning results. The learning results could show whether two input vectors are similar to each other or belong to the same class. However, in the case of high-dimensional input vectors, the structure of the learning problem can rarely be detected in the two dimensional map. A procedure is provided for interpreting the learning results using linguistic variables.

After the learning process, the weight matrix W represents the weight of each feature of the input patterns to the output. Such a matrix defines a map for a single feature only. For each feature of the input patterns, fuzzy sets are specified by a linguistic description B (one fuzzy set for each variable). They are applied to the weight matrix W to obtain a number of transformed matrices. Each combination of linguistic terms is a possible description of a pattern subset or cluster. To check a linguistic description B for validity, the transformed maps are intersected and a matrix D is obtained. Matrix D determines the compatibility of the learning result with the linguistic description B. $D^{(B)}$ is a fuzzy relation, and $D^{(B)}$ is interpreted as the degree of support of B. By describing $D^{(B)}$ by its α-cuts D^B_α one obtains subsets of output nodes, whose degree of membership is at least α such that the confidence of all patterns X_α belong to the class described by Bvanishes with decreasing α. Each B is a valid description of a cluster if $D^{(B)}$ has a non-empty α-cut D^B_α. If the features are separated into input and output features according to

the application considered, then each B represents a linguistic rule, and by examining each combination of linguistic values, a complete fuzzy rule base can be created. This method also shows which patterns belong to a fuzzy rule, because they are not contained in any subset X_α. An important advantage when compared to FAMs is that the rules are not weighted. The problem is with the determination of the number of output neurons and the α values for each learning problem. Compared to FAM, since the form of the membership function determines a crucial role in the performance the data could be better exploited. Since Kosko's learning procedure does not take into account of the neighborhood relation between the output neurons, perfect topological mapping from the input patterns to the output patterns might not be obtained sometimes. Thus, the FAM learning procedure is more dependent on the sequence of the training data than Pedryz et al. procedure. The structure of the feature space is initially determined and then the linguistic descriptions best matching the learning results by using the available fuzzy partitions are obtained. If a large number of patterns fit none of the descriptions, this may be due to an insufficient choice of membership functions and they can be determined anew. Hence for learning the fuzzy rules this approach is preferable compared to FAM. Performance of this method still depends on the learning rate and the neighborhood size for weight modification, which is problem dependant and could be determined heuristically. Fuzzy c-means algorithm also has been explored to determine the learning rate and neighborhood size by Bezdek et al. [9].

3.2.3 Systems Capable of Learning Fuzzy Set Parameters

Nomura et al. [44] proposed a supervised learning technique to fine-tune the fuzzy sets of an existing Sugeno type fuzzy system. The learning algorithm uses a gradient descent procedure that uses an error measure E (difference between the actual and target outputs) to fine-tune the parameters of the membership functions (MF). The procedure is very similar to the delta rule for multilayer perceptrons. The learning takes place in an offline mode. For the input vector, the resulting error E is calculated and based on that the consequent parts (a real value) are updated. Then the same patterns are propagated again and only the parameters of the MFs are updated. This is done to take the changes in the consequents into account when the antecedents are modified. A severe drawback of this approach is that the representation of the linguistic values of the input variables depends on the rules they appear in. Initially identical linguistic terms are represented by identical membership functions. During the learning process, they may be developed differently, so that identical linguistic terms are represented by different fuzzy sets. The proposed approach is applicable only to Sugeno type fuzzy inference system. Using a similar approach, Miyoshi et al. [38] adapted fuzzy T-norm and T-conorm operators while Yager et al. [53] adapted the defuzzification operator using a supervised learning algorithm.

3.3 Concurrent Neuro-Fuzzy System

In a concurrent model, 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 3.2 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.

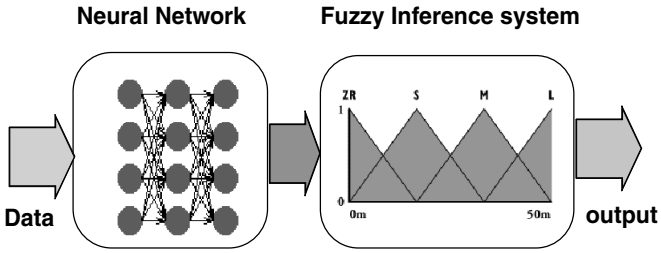

Fig. 3.2. Concurrent neuro-fuzzy model

3.4 Integrated Neuro-Fuzzy Systems

In an integrated model, 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. Due to the homogenous structure of neural network, it is hard to extract structured knowledge from either the weights or the configuration of the network. The weights of the neural network represent the coefficients of the hyper-plane that partition the input space into two regions with different output values. If we can visualize this hyper-plane structure from the training data then the subsequent

learning procedures in a neural network can be reduced. 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 3.1. 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 3.1 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. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural network like architecture. However the conventional neural network learning algorithms (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. In Sects. 3.4.1 and 3.4.2, we will discuss how to model integrated neuro-fuzzy systems implementing Mamdani [36] and Takagi-Sugeno FIS [47].

3.4.1 Mamdani Integrated Neuro-Fuzzy Systems

A Mamdani neuro-fuzzy system uses a supervised learning technique (back-propagation learning) to learn the parameters of the membership functions. Architecture of Mamdani neuro-fuzzy system is illustrated in Fig. 3.3. The detailed function of each layer is as follows:

Layer-1*(input layer):* No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. The link weight in layer 1 is unity.

Layer-2 *(fuzzification layer):* Each node in this layer corresponds to one linguistic label (excellent, good, etc.) to one of the input variables in layer 1. In other words, the output link represent the membership value, which specifies the degree to which an input value belongs to a fuzzy set, is calculated in layer 2. A clustering algorithm will decide the initial number and type of membership functions to be allocated to each of the input variable. The final shapes of the MFs will be fine tuned during network learning.

Layer-3 *(rule antecedent layer):* A node in this layer represents the antecedent part of a rule. Usually a T-norm operator is used in this node. The

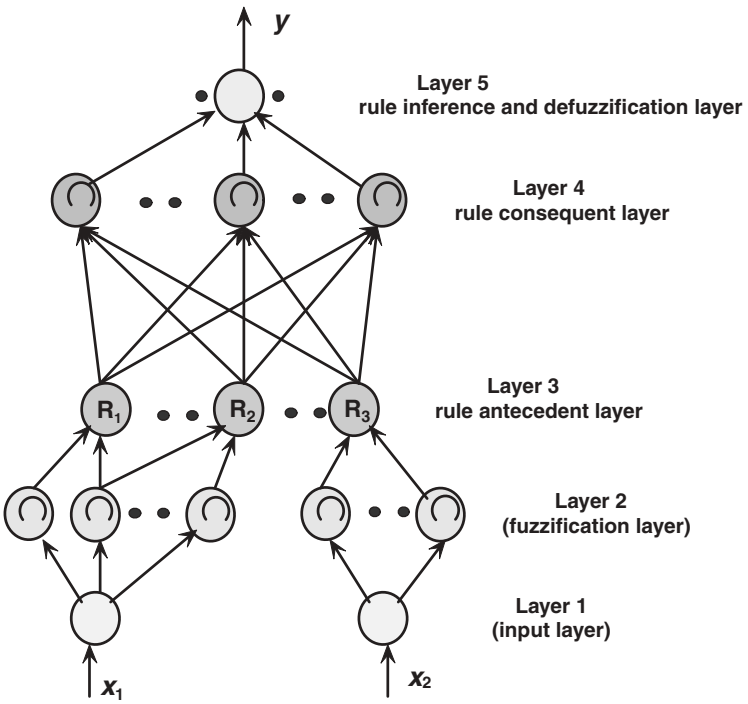

Fig. 3.3. Mamdani neuro-fuzzy system

output of a layer 3 node represents the firing strength of the corresponding fuzzy rule.

Layer-4 *(rule consequent layer):* This node basically has two tasks. To combine the incoming rule antecedents and determine the degree to which they belong to the output linguistic label (high, medium, low, etc.). The number of nodes in this layer will be equal to the number of rules.

Layer-5 *(combination and defuzzification layer):* This node does the combination of all the rules consequents using a T-conorm operator and finally computes the crisp output after defuzzification.

3.4.2 Takagi-Sugeno Integrated Neuro-Fuzzy System

Takagi-Sugeno neuro-fuzzy systems make use of a mixture of backpropagation to learn the membership functions and least mean square estimation to determine the coefficients of the linear combinations in the rule's conclusions. A step in the learning procedure got two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and

in this epoch, backpropagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated. The detailed functioning of each layer (as depicted in 3.4) is as follows:

Layers 1, 2 and 3 functions the same way as Mamdani FIS.

Layer 4 *(rule strength normalization):* Every node in this layer calculates the ratio of the i-th rule's firing strength to the sum of all rules firing strength

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1, 2.. \tag{3.1}$$

Layer-5 *(rule consequent layer):* Every node i in this layer is with a node function

$$\overline{w_i} f_i = \overline{w_i} \left(p_i \, x_1 + q_i \, x_2 + r_i\right) \tag{3.2}$$

where $\overline{w_i}$ is the output of layer 4, and $\{p_i, q_i, r_i\}$ is the parameter set. A well-established way is to determine the consequent parameters using the least means squares algorithm.

Layer-6 *(rule inference layer)* The single node in this layer computes the overall output as the summation of all incoming signals

$$\textit{Overall output} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{3.3}$$

In the following sections, we briefly discuss the different integrated neuro-fuzzy models that make use of the complementarities of neural networks and fuzzy inference systems implementing a Mamdani or Takagi Sugeno fuzzy inference system. Some of the major works in this area are GARIC [8], FALCON [32], ANFIS [24], NEFCON [40], NEFCLASS [41], NEFPROX [43], FUN [48], SONFIN [16], FINEST [50], EFuNN [26], dmEFuNN [27], EvoNF [4], and many others [25, 39, 54].

3.4.3 Adaptive Network Based Fuzzy Inference System (ANFIS)

ANFIS is perhaps the first integrated hybrid neuro-fuzzy model [24] and the architecture is very similar to Fig. 3.4. A modified version of ANFIS as shown in Fig. 3.5 is capable of implementing the Tsukamoto fuzzy inference system [24, 51] as depicted in Fig. 3.6. In the Tsukamoto FIS, the overall output is the weighted average of each rule's crisp output induced by the rule's firing strength (the product or minimum of the degrees of match with the premise part) and output membership functions. The output membership functions used in this scheme must be monotonically non-decreasing. The first hidden layer is for fuzzification of the input variables and T-norm operators are deployed in the second hidden layer to compute the rule antecedent part. The third hidden layer normalizes the rule strengths followed by the fourth hidden layer where the consequent parameters of the rule are determined. Output layer computes the overall input as the summation of all incoming signals.

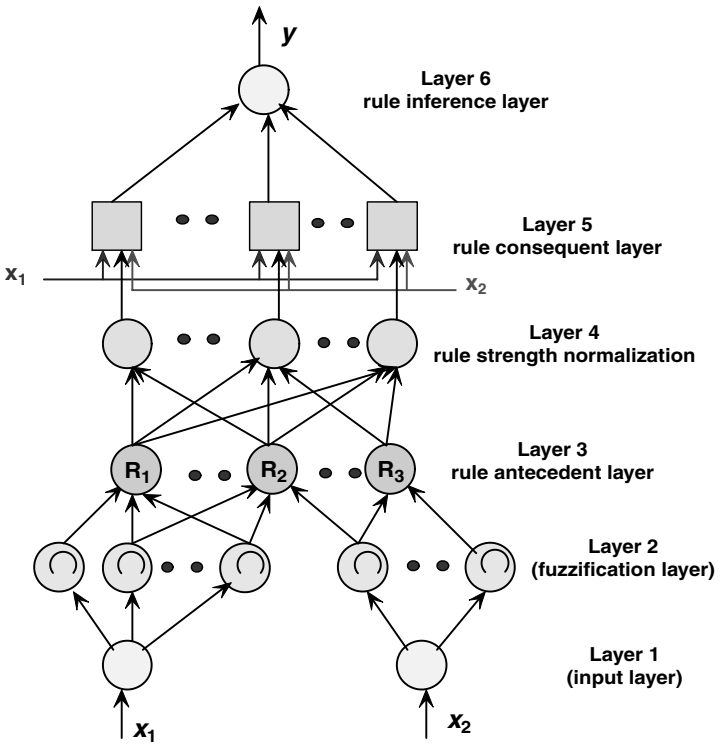

Fig. 3.4. Takagi Sugeno neuro-fuzzy system

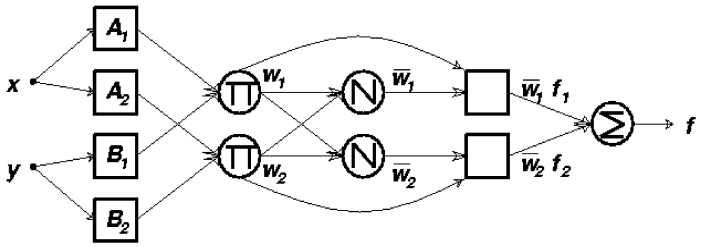

Fig. 3.5. Architecture of ANFIS implementing Tsukamoto fuzzy inference system

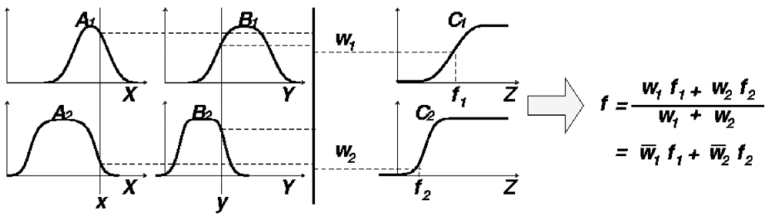

Fig. 3.6. Tsukamoto fuzzy reasoning

In ANFIS, the adaptation (learning) process is only concerned with parameter level adaptation within fixed structures. For large-scale problems, it will be too 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. However, the learning procedure of ANFIS does not provide the means to apply constraints that restrict the kind of modifications applied to the membership functions. When using Gaussian membership functions, operationally ANFIS can be compared with a radial basis function network.

3.4.4 Fuzzy Adaptive Learning Control Network (FALCON)

FALCON [32] has a five-layered architecture as shown in Fig. 3.7 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

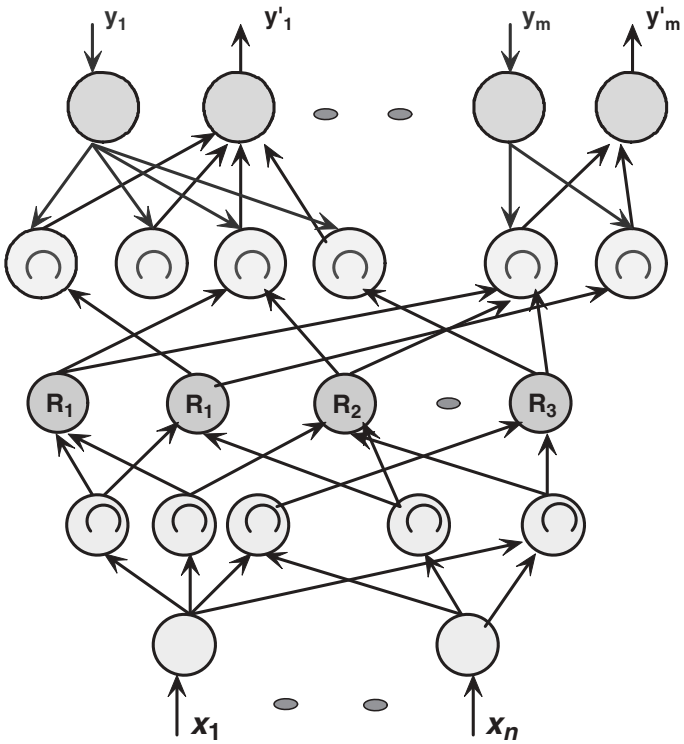

Fig. 3.7. Architecture of FALCON

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 backpropagation algorithm is used for the supervised learning. Hence FALCON algorithm provides a framework for structure and parameter adaptation for designing neuro-fuzzy systems [32].

3.4.5 Generalized Approximate Reasoning Based Intelligent Control (GARIC)

GARIC [8] is an extended version of Berenji's Approximate Reasoning based Intelligent Control (ARIC) that implements a fuzzy controller by using several specialized feedforward neural networks [7]. Like ARIC, it consists of an Action state Evaluation Network (AEN) and an Action Selection Network (ASN). The AEN is an adaptive critic that evaluates the actions of the ASN. The ASN does not use any weighted connections, but the learning process modifies parameters stored within the units of the network. Architecture of the GARIC-ASN is depicted in Fig. 3.8. ASN of GARIC is feedforward network with

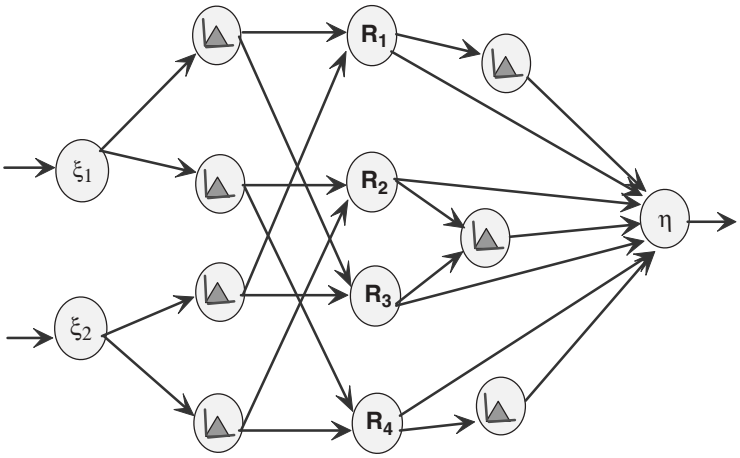

Fig. 3.8. ASN of GARIC

five layers. 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. The learning algorithm of the AEN of GARIC is equivalent to that of its predecessor ARIC. However, the ASN learning procedure is different from the procedure used in ARIC. 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 interpretation of GARIC is improved compared to GARIC. The relatively complex learning procedure and the architecture of GARIC can be seen as a main disadvantage of GARIC.

3.4.6 Neuro-Fuzzy Controller (NEFCON)

The learning algorithm defined for NEFCON is able to learn fuzzy sets as well as fuzzy rules implementing a Mamdani type FIS [40]. This method can be considered as an extension to GARIC that also use reinforcement learning but need a previously defined rule base. Figure 3.9 illustrates the basic NEFCON architecture with 2 inputs and five fuzzy rules [40]. The inner nodes $R_1, \ldots, R_5$ represent the rules, the nodes ξ_1, ξ_2, and η the input and output values, and μ_r, V_r 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. 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. The first phase is designed to learn the rule base and the second phase optimizes the rules by shifting or modifying the fuzzy sets of the rules. Two methods are available for learning the rule base. Incremental rule learning is used when the correct out put 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. However it can be applied to

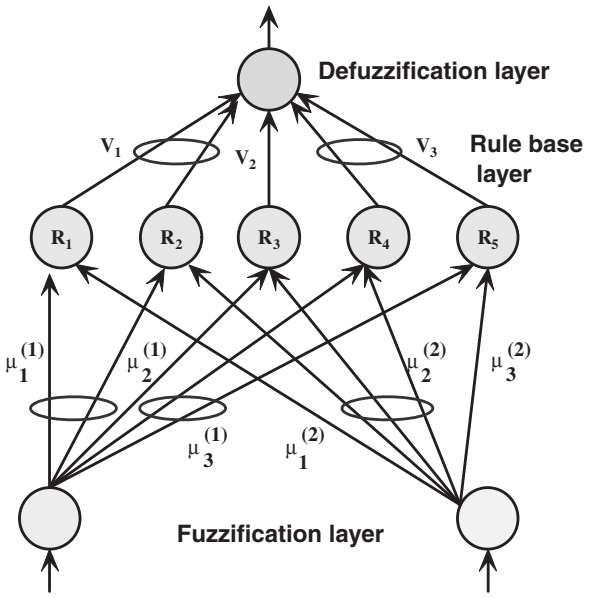

Fig. 3.9. Architecture of NEFCON

unknown processes without difficulty, and there is no need to know or to guess an optimal output value. Both phases use a fuzzy error E, which describes the quality of the current system state, to learn or to optimize the rule base. To obtain a good rule base it must be ensured that the state space of the process is sufficiently covered during the learning process. 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 Backpropagation (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.

3.4.7 Neuro-Fuzzy Classification (NEFCLASS)

NEFCLASS is used to derive fuzzy rules from a set of data that can be separated in different crisp classes [41]. The rule base of a NEFCLASS system approximates an unknown function ϕ that represents the classification problem and maps an input pattern x to its class C_i:

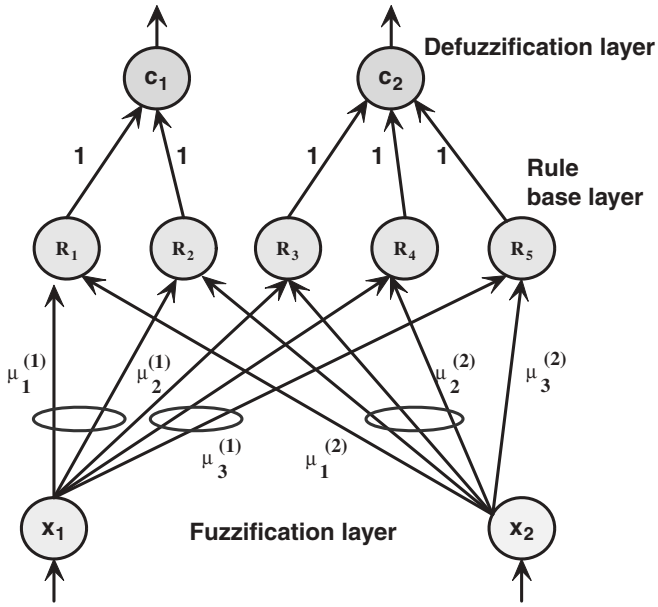

Fig. 3.10. Architecture of NEFCLASS

Because of the propagation procedures used in NEFCLASS the rule base actually does not approximate ϕ but a function ϕ'. We obtain $\phi(x)$ from the equality $\phi(x) = \phi(\phi(x))$, where ϕ reflects the interpretation of the classification result obtained from a NEFCLASS system [42]. Figure 3.10 illustrates the NEFCLASS system that maps patterns with two features into two distinct classes by using five linguistic rules. The NEFCLASS very much resemble the NEFCON system except the slight variation in the learning algorithm and the interpretation of the rules. As in NEFCON system, in NEFCLASS identical linguistic values of an input variable are represented by the same fuzzy set. As classification is the primary task of NEFCLASS, there should be two rules with identical antecedents and each rule unit must be connected to only one output unit. The weights between rule layer and the output layer only connect the units. A NEFCLASS system can be built from partial knowledge about the patterns, and can then be refined by learning, or it can be created from scratch by learning. A user must define a number of initial fuzzy sets that partition the domains of the input features, and specify a value for k, i.e. the maximum number of rule nodes that may be created in the hidden layer. NEFCLASS makes use of triangular membership functions and the learning algorithm of the membership functions uses an error measure 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. Being a classification system, we are not much interested in the exact output values. In addition, we take a winner-takes-all interpretation for the output, and we are mainly

interested in the correct classification result. The incremental rule learning in NEFCLASS is much less expensive than decremental rule learning in NEFCON. It is possible to build up a rule base in a single sweep through the training set. Even for higher dimensional problems, the rule base is completed after at most three cycles. Compared to neural networks, NEFCLASS uses a much simpler learning strategy. There is no vector quantization involved in finding the rules (clusters, and there is no gradient information needed to train the membership functions. Some other advantages are interpretability, possibility of initialization (incorporating prior knowledge) and its simplicity.

3.4.8 Neuro-Fuzzy Function Approximation (NEFPROX)

NEFPROX system is based on plain supervised learning (fixed learning problem) and it is used for function approximation [43]. It is a modified version of the NEFCON model without the reinforcement learning. NEFPROX (Fig. 3.11) is very much similar to NEFCON and NEFCLASS except the fact that NEFCON have only a single output node and NEFCLASS systems do not use membership functions on the conclusion side. We can initialize the NEFPROX system if we already know suitable rules or else the system is capable to incrementally learn all rules. NEFPROX architecture is as shown in Fig. 11. While ANFIS is capable to implement only Sugeno models with differentiable functions, NEFPROX can learn common Mamdani type of fuzzy

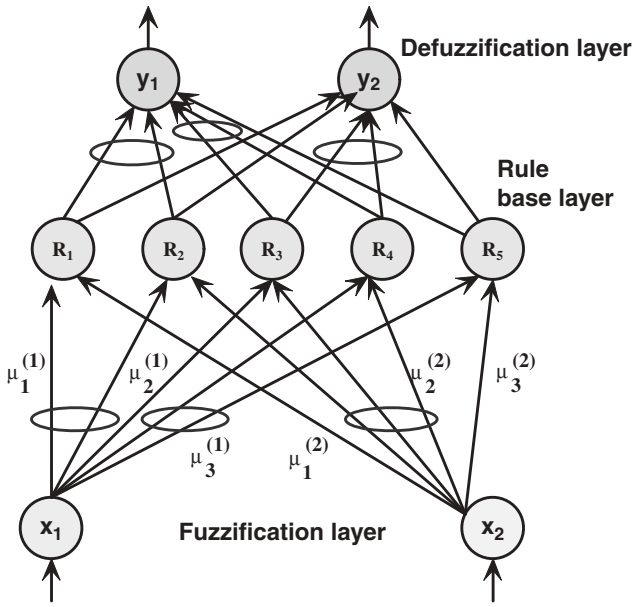

Fig. 3.11. Architecture of NEFPROX

system from data. Further NEFPROX is much faster compared to ANFIS to yield results.

3.4.9 Fuzzy Inference Environment Software with Tuning (FINEST)

FINEST 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 [50]. 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 also these parameters. In the same way, the tuning mechanism can 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.

FINEST make use of a backpropagation algorithm for the fine-tuning of the parameters. Figure 3.12 shows the layered architecture of FINEST and the calculation process of the fuzzy inference. The input values (x_i) 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. 3.12, the function and_i, I_i 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 and_i, I_i, $comb$ and membership functions of each fuzzy predicate are defined with some parameters.

Backpropagation 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. Parameterization of the inference procedure is very much essential for proper application of the tuning algorithm.

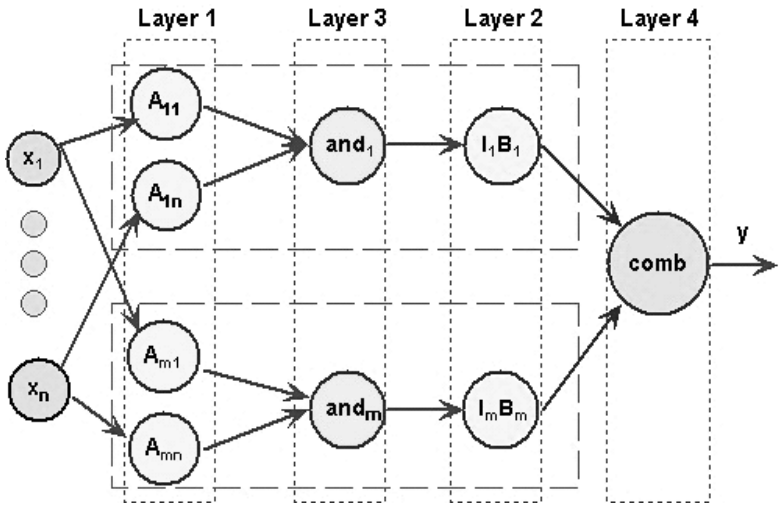

Fig. 3.12. Architecture of FINEST

3.4.10 Self Constructing Neural Fuzzy Inference Network (SONFIN)

SONFIN 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 [16]. 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. Figure 3.13 illustrates the 6-layer structure of SONFIN.

Learning progresses concurrently in two stages for the construction of SONFIN. The *structure* learning includes both the precondition and consequent structure identification of a fuzzy *if-then* rule. The parameter learning is based on supervised learning algorithms, the parameters of the linear equations in the consequent parts are adjusted by either LMS or RLS algorithms

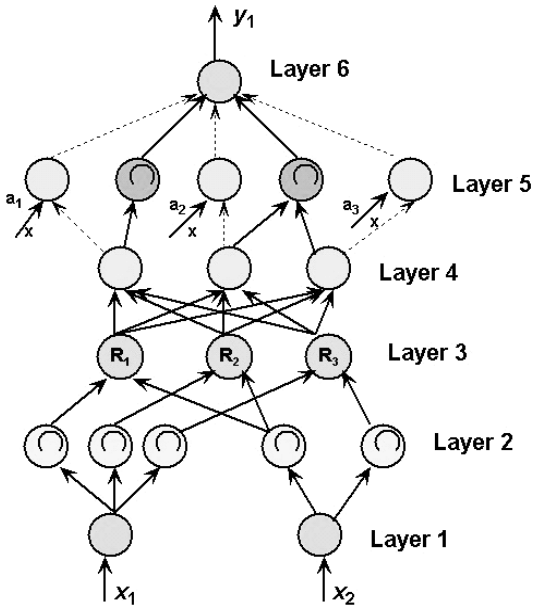

Fig. 3.13. Six layered architecture of SONFIN

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:

- **Input-output space partitioning**

The way the input space is partitioned determines the number of rules extracted from the training data as well as the number of fuzzy sets on the universal of discourse of each input variable. For each incoming pattern x the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. The center and width of the corresponding membership functions (of the newly formed fuzzy rules) are assigned according to the first-neighbor heuristic. For each rule generated, the next step is to decompose the multidimensional membership function to corresponding $1 - D$ membership function for each input variable. For the output space partitioning, almost a similar measure is adopted. Performance of SONFIN can be enhanced by incorporating a transformation matrix R into the structure, which accommodates all the *a priori* knowledge of the data set.

- **Construction of fuzzy rule base**

Generation of new input cluster corresponds to the generation of a new fuzzy rule, with its precondition part constructed by the learning algorithm in

process. At the same time we have to decide the consequent part of the generated rule. This is done using a algorithm based on the fact that different preconditions of rules may be mapped to the same consequent fuzzy set. Since only the center of each output membership function is used for defuzzification, the consequent part of each rule may simply be regarded as a singleton. Compared to the general fuzzy rule based models with singleton output where each rule has its own singleton value, fewer parameters are needed in the consequent part of the SONFIN, especially for complicated systems with a large number of rules.

- **Optimal consequent structure identification**

TSK model can model a sophisticated system with a few rules. In SONFIN, instead of using the linear combination of all input variables as the consequent part, only the most significant input variables are used as the consequent terms of the SONFIN. The significant terms will be chosen and added to the network incrementally any time when the parameter learning cannot improve the network output accuracy anymore during the online learning process. The consequent structure identification scheme in SONFIN is a kind of node growing method in ANNs. When the effect of the parameter learning diminished (output error is not decreasing), additional terms are added to the consequent part.

- **Parameter identification**

After the network structure is adjusted according to the current training pattern, the network then enters the parameter identification phase to adjust the parameters of the network optimally based on the same training pattern. Parameter learning is performed on the whole network after structure learning, no matter whether the nodes (links) are newly added or are existent originally. Backpropagation algorithm is used for this supervised learning. SONFIN is perhaps one of the most computational expensive among all neuro-fuzzy models. The network is adaptable to the users specification of required accuracy.

3.4.11 Fuzzy Net (FUN)

In FUN in order to enable an unequivocal translation of fuzzy rules and membership functions into the network, special neurons have been defined, which, through their activation functions, can evaluate logic expressions [48]. 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 14 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

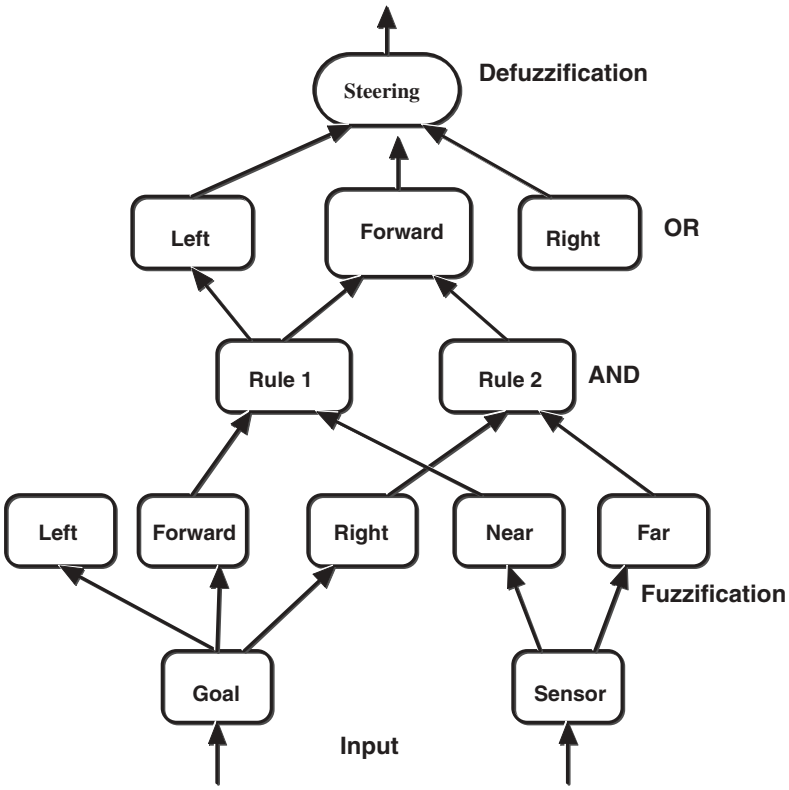

Fig. 3.14. Architecture of the FUN showing the implementation of a sample rule

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. FUN network is depicted in Fig. 3.14.

Rule: *IF (Goal IS forward AND Sensor IS near) OR (goal IS right AND sensor IS far) THEN steering = forward*

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.

- **Learning of the rules and membership functions**

The rules are represented in the net through the connections between the layers. The learning of the rules is implemented as a stochastic search in the rule space: a randomly chosen connection is changed and the new network performance is verified with a cost function. If the performance is worse, the change is undone, otherwise it is kept and some other changes are tested, until the desired output is achieved. As the learning algorithm should preserve the semantic of the rules, it has to be controlled in such a way that no two values of the same variable appear in the same rule. This is achieved by swapping connections between the values of the same variable. FUN uses a mixture of gradient descent and stochastic search for updating the membership functions. A maximum change in a random direction is initially assigned to all Membership function Descriptors (MFDs). In a random fashion one MFD of one linguistic variable is selected, and the network performance is tested with this MFD altered according to the allowable change for this MFD. If the network performs better according to the given cost function, the new value is accepted and next time another change is tried in the same direction. Contrary if the network performs worse, the change is reversed. To guarantee convergence, the changes are reduced after each training step and shrink asymptotically towards zero according to the learning rate. As evident, FUN system is initialized by specifying a fixed number of rules and a fixed number of initial fuzzy sets for each variable and the network learns through a stochastic procedure that randomly changes parameters of membership functions and connections within the network structure Since no formal neural network learning technique is used it is questionable to call FUN a neuro-fuzzy system.

3.4.12 Evolving Fuzzy Neural Networks (EFuNN)

EFuNNs [26] and dmEFuNNs [27] 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, and etc. through local element tuning.

In EFuNNs all nodes are created during learning. EFuNN has a five-layer architecture as shown in Fig. 3.15. 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. 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

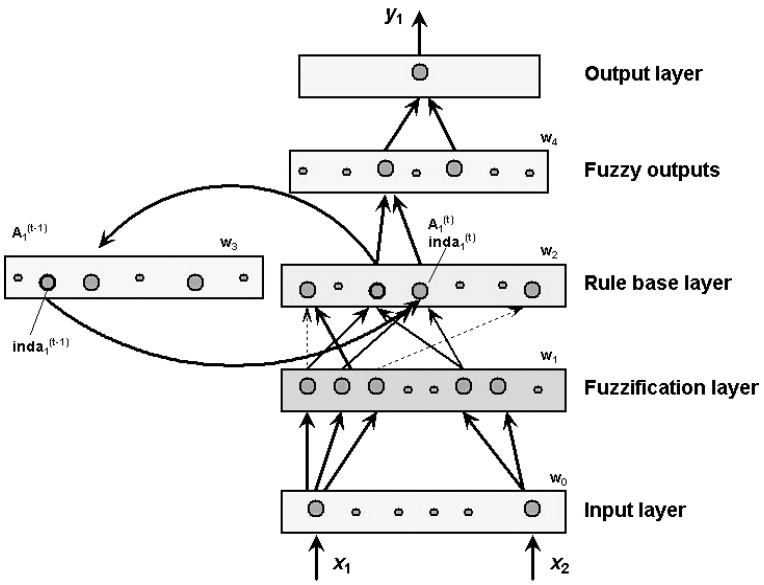

Fig. 3.15. Architecture of EFuNN

$W_2(r)$, 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.

3.4.13 Dynamic Evolving Fuzzy Neural Networks (dmEFuNNs)

Dynamic Evolving Fuzzy Neural Networks (dmEFuNN) model 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 Fig. 3.16.

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 normalised local distance to the fuzzy input vector, and then, a TakagiSugeno

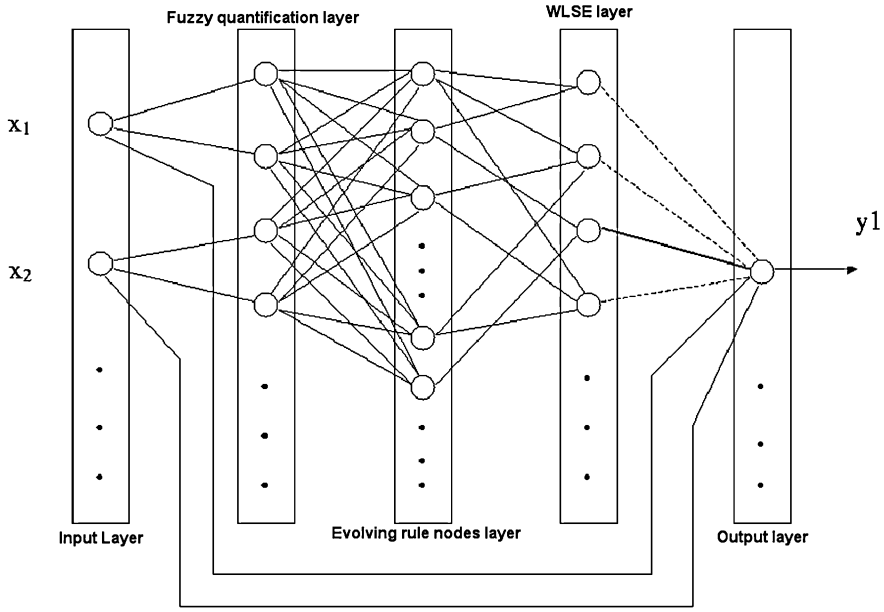

Fig. 3.16. Architecture of dmEFuNN

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 optimising 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 TakagiSugeno 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.

3.5 Discussions

As evident, both cooperative and concurrent models are not fully interpretable due to the presence of neural network (black box concept). Whereas an integrated neuro-fuzzy model is interpretable and capable of learning in a supervised mode (or even reinforcement learning like NEFCON). In FALCON,

GARIC, ANFIS, NEFCON, SONFIN, FINEST and FUN the learning process is only concerned with parameter level adaptation within fixed structures. For large-scale problems, it will be too complicated to determine the optimal premise-consequent structures, rule numbers etc. User has to provide the architecture details (type and quantity of MF's for input and output variables), type of fuzzy operators etc. FINEST provides a mechanism based on the improved generalized modus ponens for fine tuning of fuzzy predicates and combination functions and tuning of an implication function. An important feature of EFuNN and dmEFuNN is the one pass (epoch) training, which is highly capable of online learning. Table 3.2 provides a comparative performance of some neuro fuzzy systems for predicting the Mackey-Glass chaotic time series [35]. Due to unavailability of source codes, we are unable to provide a comparison with all the models. Training was done using 500 data sets and the considered NF models were tested with another 500 data sets [1].

Table 3.2. Performance of neuro-fuzzy systems

System	Epochs	Test RMSE
ANFIS	75	0.0017
NEFPROX	216	0.0332
EFuNN	1	0.0140
dmEFuNN	1	0.0042
SONFIN	1	0.0180

Among NF models ANFIS has the lowest Root Mean Square Error (RMSE) and NEPROX the highest. This is probably due to Takagi-Sugeno rules implementation in ANFIS compared to the Mamdani-type fuzzy system in NEFPROX. However, NEFPROX outperformed ANFIS in terms of computational time. Due to fewer numbers of rules SONFIN, EFuNN and dmEFuNN are also able to perform faster than ANFIS. Hence, there is a tradeoff between interpretability and accuracy. Takagi Sugeno type inference systems are more accurate but require more computational effort. While Mamdani type inference, systems are more interpretable and required less computational load but often with a compromise on accuracy.

As the problem become, more complicated manual definition of NF architecture/parameters becomes complicated. The following questions remain unanswered:

- For input/output variables, what are the optimal number of membership functions and shape?
- What is the optimal structure (rule base) and fuzzy operators?
- What are the optimal learning parameters?
- Which fuzzy inference system (example. Takagi-Sugeno, Mamdani etc.) will work the best for a given problem?

3.5.1 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 [14]. EvoNF [4] 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. Figure 3.17 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. For example, if there

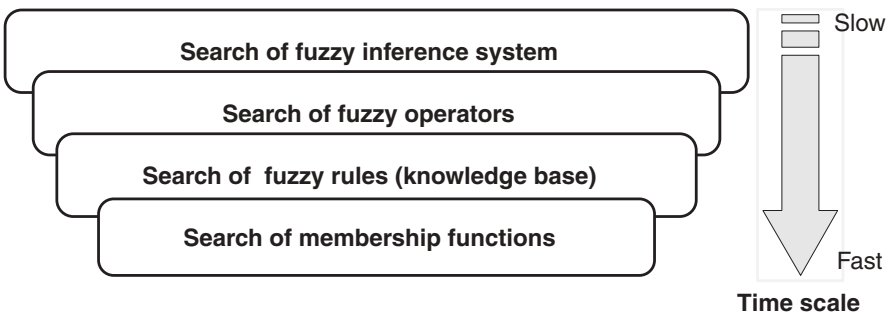

Fig. 3.17. Interaction of evolutionary search mechanisms in the adaptation of fuzzy inference system

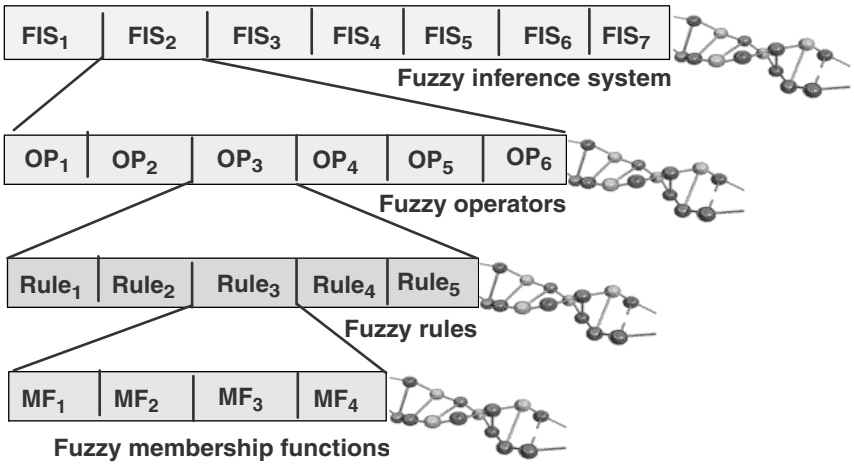

Fig. 3.18. Chromosome representation of the adaptive fuzzy inference system using evolutionary computation and neural learning

is more prior knowledge about the fuzzy rule base than the fuzzy operators then it is better to implement the search for fuzzy rule base at a higher level. The problem representation (genetic coding) is illustrated in Fig. 3.18. Please refer [4] for more technical details.

Automatic adaptation of membership functions is popularly known as self tuning and the genome encodes parameters of trapezoidal, triangle, logistic, hyperbolic-tangent, Gaussian membership functions etc.

Evolutionary search of fuzzy rules can be carried out using three approaches. In the first method, (Michigan approach) the fuzzy knowledge base is adapted because of antagonistic roles of competition and cooperation of fuzzy rules [14]. Each genotype represents a single fuzzy rule and the entire population represents a solution. A classifier rule triggers whenever its condition part matches the current input, in which case the proposed action is sent to the process to be controlled. The global search algorithm will generate new classifier rules based on the rule strengths acquired during the entire process. The fuzzy behavior is created by an activation sequence of mutually collaborating fuzzy rules. The entire knowledge base is build up by a cooperation of competing multiple fuzzy rules.

The second method (Pittsburgh approach) evolves a population of knowledge bases rather than individual fuzzy rules [14]. Genetic operators serve to provide a new combination of rules and new rules. In some cases, variable length rule bases are used; employing modified genetic operators for dealing with these variable length and position independent genomes. The disadvantage is the increased complexity of search space and additional computational burden especially for online learning.

The third method (iterative rule learning approach) is very much similar to the first method with each chromosome representing a single rule, but contrary to the Michigan approach, only the best individual is considered to form part of the solution, discarding the remaining chromosomes in the population. The evolutionary learning process builds up the complete rule base through a iterative learning process [18].

3.6 Conclusions

In this chapter, we presented the different ways to learn fuzzy inference systems using neural network learning techniques. As a guideline, for neuro-fuzzy 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. For online learning, global optimization procedures might sound computational expensive. Fortunately, evolutionary algorithms work with a population of independent solutions, which makes it easy to distribute the computational load among several processors using parallel algorithms.

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.

3.7 Acknowledgements

The author wishes to thank the anonymous reviewers for their constructive comments which helped to improve the presentation of the chapter.

References

1. A. Abraham, Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Springer-Verlag Germany, Jose Mira and Alberto Prieto (Eds.), Granada, Spain, pp. 269–276, 2001. 54, 76
2. A. Abraham, Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications, Abraham A., Jain L. and Kacprzyk J. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chap. 1, pp. 1–35, 2002. 54
3. A. Abraham and M.R. Khan, Neuro-Fuzzy Paradigms for Intelligent Energy Management, Innovations in Intelligent Systems: Design, Management and Applications, Abraham A., Jain L. and Jan van der Zwaag B. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chap. 12, pp. 285–314, 2003. 54
4. A. Abraham, EvoNF: A Framework for Optimization of Fuzzy Inference Systems Using Neural Network Learning and Evolutionary Computation, The 17th IEEE International Symposium on Intelligent Control, ISIC'02, IEEE Press, pp. 327–332, 2002. 60, 77, 78
5. P. Andlinger and E.R. Reichl, Fuzzy-Neunet: A Non Standard Neural Network, In Prieto et al., pp. 173–180, 1991. 53
6. M. Arao, T. Fukuda and K. Shimokima, Flexible Intelligent System based on Fuzzy Neural Networks and Reinforcement Learning, In proceedings of IEEE International Conference on Fuzzy Systems, Vol 5(1), pp. 69–70, 1995. 53
7. H.R. Berenji and P. Khedkar, Fuzzy Rules for Guiding Reinforcement Learning, In International. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'92), pp. 511–514, 1992. 63
8. H.R. Berenji and P. Khedkar, Learning and Tuning Fuzzy Logic Controllers through Reinforcements, IEEE Transactions on Neural Networks, Vol (3), pp. 724–740, 1992. 60, 63
9. J.C. Bezdek and S.K. Pal, Fuzzy Models for Pattern Recognition, IEEE Press, New York, 1992. 56
10. M. Brown, K. Bossley and D. Mills, High Dimensional Neurofuzzy Systems: Overcoming the Course of Dimensionality, In Proceedings. IEEE International. Conference on Fuzzy Systems, pp. 2139–2146, 1995. 53
11. J.J. Buckley and Y. Hayashi, Hybrid neural nets can be fuzzy controllers and fuzzy expert systems, Fuzzy Sets and Systems, 60: pp. 135–142, 1993. 53
12. H. Bunke and A. Kandel, Neuro-Fuzzy Pattern Recognition, World Scientific Publishing CO, Singapore, 2000. 53
13. G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps, IEEE Transactions Neural Networks, 3(5), pp. 698–712, 1992. 53
14. O. Cordón F. Herrera, F. Hoffmann and L. Magdalena, Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientific Publishing Company, Singapore, 2001. 77, 78
15. F. De Souza, M.M.R. Vellasco, M.A.C. Pacheco, The Hierarchical Neuro-Fuzzy BSP Model: An Application in Electric Load Forecasting, Connectionist Models of Neurons, Learning Processes and Artificial Intelligence, Jose Mira et al (Editors), LNCS 2084, Springer Verlag Germany, 2001. 54

16. J.C. Feng and L.C. Teng, An Online Self Constructing Neural Fuzzy Inference Network and its Applications, IEEE Transactions on Fuzzy Systems, Vol 6, No.1, pp. 12–32, 1998. 60, 69
17. R. Fuller, Introduction to Neuro-Fuzzy Systems, Studies in Fuzziness and Soft Computing, Springer Verlag, Germany, 2000. 53, 54
18. A. Gonzalez and F. Herrera, Multi-Stage Genetic Fuzzy Systems Based on the Iterative Rule Learning Approach, Mathware and Soft Computing Vol 4, pp. 233–249, 1997. 79
19. M.M. Gupta, Fuzzy Neural Computing Systems, In Proceedings of SPIE, Vol 1710, Science of Artificial Neural Networks, Vol 2, pp. 489–499, 1992. 53
20. S.K. Halgamuge and M. Glesner, Neural Networks in Designing Fuzzy Systems for Real World Applications, Fuzzy Sets and Systems, 65: pp. 1–12, 1994. 53
21. Y. Hayashi and J.J. Buckley, Approximations Between Fuzzy Expert Systems and Neural Networks, International Journal of Approximate Reasoning, Vol 10, pp. 63–73, 1994. 53
22. Y. Hayashi and A. Imura, Fuzzy Neural Expert System with Automated Extraction of Fuzzy If-Then Rules from a Trained Neural Network, In First International. Symposium on Uncertainty Modeling and Analysis, pp. 489–494, 1990. 53
23. J. Hollatz, Neuro-Fuzzy in Legal Reasoning, In Proceedings. IEEE International. Conference on Fuzzy Systems, pp. 655–662, 1995. 54
24. R. Jang, Neuro-Fuzzy Modeling: Architectures, Analyses and Applications, Ph.D. Thesis, University of California, Berkeley, 1992. 60
25. A. Kandel, Q.Y. Zhang and H. Bunke, A Genetic Fuzzy Neural Network for Pattern Recognition, In IEEE Transactions on Fuzzy Systems, pp. 75–78, 1997. 60
26. N. Kasabov, Evolving Fuzzy Neural Networks – Algorithms, Applications and Biological Motivation, In Yamakawa T and Matsumoto G (Eds), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, pp. 271–274, 1998. 60, 73
27. N. Kasabov and S. Qun, Dynamic Evolving Fuzzy Neural Networks with m-out-of-n Activation Nodes for On-line Adaptive Systems, Technical Report TR99/04, Department of information science, University of Otago, New Zealand, 1999. 60, 73
28. E. Khan and P. Venkatapuram, Neufuz: Neural Network Based Fuzzy Logic Design Algorithms, In Proceedings IEEE International Conference on Fuzzy Systems, pp. 647–654, 1993. 54
29. B. Kosko, Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, New Jersey, 1992. 54
30. W. Li, Optimization of a Fuzzy Controller Using Neural Network, In Proceedings IEEE International Conference on Fuzzy Systems, pp. 223–227, 1994. 54
31. X.H. Li and C.L.P. Chen, The Equivalance Between Fuzzy Logic Systems and Feedforward Neural Networks, IEEE Transactions on Neural Networks, Vol 11, No. 2, pp. 356–365, 2000. 53
32. C.T. Lin and C.S.G. Lee, Neural Network based Fuzzy Logic Control and Decision System, IEEE Transactions on Comput. (40(12): pp. 1320–1336, 1991. 53, 54, 60, 62, 63
33. C.T. Lin and C.S.G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice Hall Inc, USA, 1996. 53

34. A. Lotfi, Learning Fuzzy Inference Systems, Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Queensland, Australia, 1995. 54
35. M.C. Mackey and L. Glass, Oscillation and Chaos in Physiological Control Systems, Science Vol 197, pp. 287–289, 1977. 76
36. E.H. Mamdani and S. Assilian, An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Man-Machine Studies, Vol 7, No.1, pp. 1–13, 1975. 58
37. S. Mitra and Y. Hayashi, Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework', IEEE Transactions on Neural Networks, Vol II, No. 3. pp. 748–768, 2000. 53
38. T. Miyoshi, S. Tano, Y. Kato and T. Arnould, Operator Tuning in Fuzzy Production Rules Using Neural networks, In Proceedings of the IEEE International Conference on Fuzzy Systems, San Francisco, pp. 641–646, 1993. 56
39. M. Mizumoto and S. Yan, A New Approach of Neurofuzzy Learning Algorithm, Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms, Ruan D (Ed.), Kluwer Academic Publishers, pp. 109–129, 1997. 60
40. D. Nauck and R. Kruse, NEFCON-I: An X-Window Based Simulator for Neural Fuzzy Controllers. In Proceedings of the IEEE International Conference on Neural Networks, Orlando, pp. 1638–1643, 1994. 60, 64
41. D. Nauck and R. Kruse, NEFCLASS: A Neuro-Fuzzy Approach for the Classification of Data, In Proceedings of ACM Symposium on Applied Computing, George K et al (Eds.), Nashville, ACM Press, pp. 461–465, 1995. 60, 65
42. D. Nauck and R. Kruse, A Neuro-Fuzzy Method to Learn Fuzzy Classification Rules from Data. Fuzzy Sets and Systems, 89, pp. 277–288, 1997. 66
43. D. Nauck D. and R. Kruse, Neuro-Fuzzy Systems for Function Approximation, Fuzzy Sets and Systems, 101, pp. 261–271, 1999. 54, 60, 67
44. H. Nomura, I. Hayashi and N. Wakami, A Learning Method of Fuzzy Inference Systems by Descent Method, In Proceedings of the First IEEE International conference on Fuzzy Systems, San Diego, USA, pp. 203–210, 1992. 54, 56
45. S.K. Pal and S. Mitra, Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing, John Wiley & Sons, Inc, USA, 1999. 54
46. W. Pedrycz and H.C. Card, Linguistic Interpretation of Self Organizing Maps, In Prroceedings of the IEEE International Conference on Fuzzy Systems, San Diego, pp. 371–378, 1992. 54, 55
47. M. Sugeno, Industrial Applications of Fuzzy Control, Elsevier Science Pub Co., 1985. 58
48. S.M. Sulzberger, N.N. Tschicholg-Gurman, S.J. Vestli, FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks, In Proceedings of IEEE Conference on Neural Networks, San Francisco, pp. 312–316, 1993. 60, 71
49. H. Takagi, Fusion Technology of Fuzzy Theory and Neural Networks - Survey and Future Directions, In Proceedings 1st International Conference on Fuzzy Logic & Neural Networks, pp. 13–26, 1990. 54
50. S. Tano, T. Oyama and T. Arnould, Deep combination of Fuzzy Inference and Neural Network in Fuzzy Inference, Fuzzy Sets and Systems, 82(2) pp. 151–160, 1996. 60, 68
51. Y. Tsukamoto, An Approach to Fuzzy Reasoning Method, Gupta M.M. et al (Eds.), Advances in Fuzzy Set Theory and Applications, pp. 137–149, 1979. 60
52. R.R. Yager, On the Interface of Fuzzy Sets and Neural Networks, In International Workshop on Fuzzy System Applications, pp. 215–216, 1988. 54

53. R.R. Yager and D.P. Filev, Adaptive Defuzzification for Fuzzy System Modeling, In Proceedings of the Workshop of the North American Fuzzy Information Processing Society, pp. 135–142, 1992. 56
54. Q.Y. Zhang and A. Kandel, Compensatory Neuro-fuzzy Systems with Fast Learning Algorithms, IEEE Transactions on Neural Networks, Vol 9, No. 1, pp. 83–105, 1998. 60