Neuro–Fuzzy Rule Generation: Survey in Soft Computing Framework

Sushmita Mitra, *Member, IEEE*, and Yoichi Hayashi, *Senior Member, IEEE*

***Abstract*—The present article is a novel attempt in providing an exhaustive survey of neuro–fuzzy rule generation algorithms. Rule generation from artificial neural networks is gaining in popularity in recent times due to its capability of providing some insight to the user about the symbolic knowledge embedded within the network. Fuzzy sets are an aid in providing this information in a more human comprehensible or natural form, and can handle uncertainties at various levels. The neuro–fuzzy approach, symbiotically combining the merits of connectionist and fuzzy approaches, constitutes a key component of soft computing at this stage. To date, there has been no detailed and integrated categorization of the various neuro–fuzzy models used for rule generation. We propose to bring these together under a unified soft computing framework. Moreover, we include both rule extraction and rule refinement in the broader perspective of rule generation. Rules learned and generated for fuzzy reasoning and fuzzy control are also considered from this wider viewpoint. Models are grouped on the basis of their level of neuro–fuzzy synthesis. Use of other soft computing tools like genetic algorithms and rough sets are emphasized. Rule generation from fuzzy knowledge-based networks, which initially encode some crude domain knowledge, are found to result in more refined rules. Finally, real-life application to medical diagnosis is provided.**

***Index Terms*—Knowledge-based networks, neuro–fuzzy computing, rule extraction, rule generation, soft computing.**

I. Introduction

ARTIFICIAL neural networks (ANN's) attempt to replicate the *computational* power (low-level arithmetic processing ability) of biological neural networks and, thereby, hopefully endow machines with some of the (higher-level) *cognitive abilities* that biological organizms possess (due in part, perhaps, to their low-level computational prowess). However, an impediment to a more widespread acceptance of ANN's is the absence of a capability to explain to the user, in a human-comprehensible form, how the network arrives at a particular decision. Neither can one say something about the *knowledge* encoded within the *blackbox*. Recently, there has been widespread activity aimed at redressing this situation by extracting the embedded knowledge in trained ANN's in the form of symbolic rules [1]–[9]. This serves to identify the attributes that, either individually or in a combination, are the most significant determinants of the decision or classification. Often an ANN solution with good generalization does not necessarily imply involvement of hidden units with distinct *meaning*. Hence any individual unit cannot essentially be associated with a single concept or feature of the problem domain. This is typical of connectionist approaches, where all information is stored in a distributed manner among the neurons and their associated connectivity.

Generally, ANN's consider a fixed topology of neurons connected by links in a predefined manner. These connection weights are usually initialized by small random values. *Knowledge-based networks* [10], [11] constitute a special class of ANN's that consider crude domain knowledge to generate the initial network architecture, which is later refined in the presence of training data. Recently, there have been some attempts in improving the efficiency of neural computation by using knowledge-based nets. This helps in reducing the searching space and time while the network traces the optimal solution. In such situations, one can extract causal factors and functional dependencies from the data domain for initial encoding of the ANN [5], [12] and later extract refined rules from the trained network.

Andrews *et al.* [6] have provided a classification scheme for connectionist rule extraction algorithms. They take into consideration

- expressive power of the rules: 1) propositional or Boolean logic, i.e., crisp or nonfuzzy, and 2) nonconventional logic, i.e., probabilistic or fuzzy;
- *translucency* of view taken in the algorithm about the underlying ANN units: 1) decompositional approach (more analytical), where each internal element of the *transparent* network is examined and 2) pedagogical or *blackbox* approach, where one observes only the input–output behavior of the *opaque* network;
- extent to which the underlying ANN incorporates specialized training regimes, i.e., *portability*;
- quality of the rules: 1) accuracy, i.e., generalization to test cases; 2) fidelity, i.e., whether they can mimic the behavior of the ANN from which they were generated; 3) consistency, i.e., whether they produce the same classification of test instances over different training instances; and 4) comprehensibility, in terms of the size of the rule set and the number of antecedents per rule;
- algorithmic complexity of the technique.

Taha and Ghosh [13] have considered additional issues related to rule extraction. These include the granularity of explanation, modifiability, theory refinement capability (to handle incompleteness, inconsistency, and/or inaccuracy of initial domain knowledge), stability/robustness to corruption in data/knowledge, and scalability for large datasets/rulebases.

S. Mitra is with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 035, India (e-mail: sushmita@isical.ac.in).

Y. Hayashi is with the Department of Computer Science, Meiji University, Tama-ku, Kawasaki 214-8571, Japan (e-mail: hayashiy@cs.meiji.ac.jp).

Unfortunately, most of the available literature on rule generation do not provide such rigorous assessment on their pros and cons. There is also a preponderance of *specific purpose* techniques, that are designed to work with a particular ANN architecture. This limits the scope of comparing the various techniques in a single framework. Unless specified otherwise, all methods surveyed in this article will deal with the analytical or decompositional approach.

Both fuzzy systems and ANN's are soft computing approaches to modeling expert behavior. The goal is to mimic the actions of an expert who solves complex problems. In other words, instead of investigating the problem in detail, one observes how an expert successfully tackles the problem and obtains knowledge by instruction and/or learning [14]. A learning process can be part of knowledge acquisition. In the absence of an expert or sufficient time or data, one can resort to reinforcement learning instead of supervised learning. If one has knowledge expressed as linguistic rules, one can build a fuzzy system. On the other hand, if one has data or can learn from a simulation or the real task, ANN's are more appropriate. The merits of both neural and fuzzy systems can be integrated in a neuro–fuzzy approach [4]. The focus of this article will be on neuro–fuzzy rule generation.

The term *rule generation* encompasses both rule extraction and rule refinement. Note that *rule extraction* here refers to extracting knowledge from the ANN, using the network parameters in the process. *Rule refinement*, on the other hand, pertains to extracting refined knowledge from the ANN that was initialized using crude domain knowledge. Rules learned and interpolated for fuzzy reasoning and fuzzy control can also be considered under rule generation. It covers, in a wider sense, the extraction of domain knowledge (say, for the initial encoding of an ANN) using nonconnectionist tools like fuzzy sets and rough sets. Unlike Tickle *et al.* [5], [6] who deal with rule extraction for nonfuzzy connectionist models (using propositional logic) only, we provide here a broader and exhaustive survey of neuro–fuzzy rule generation. Both feedforward and recurrent neural networks are considered. Although the focus is on neuro–fuzzy models, we also briefly deal with other fuzzy, neural, genetic algorithms, and rough set-based approaches to rule generation. We concentrate on categorizing the different neuro–fuzzy approaches, based on their level of integration, in a unified *soft computing* framework.

In general, the primary input to a connectionist rule generation algorithm is a representation of the trained ANN, in terms of its nodes and links, and sometimes the data set. One interprets one or more hidden and output units into rules, which may later be combined and simplified to arrive at a more comprehensible rule set. These rules can also provide new insights into the application domain. The use of ANN helps in 1) incorporating parallelism and 2) tackling optimization problems in the data domain. The models are usually suitable in data-rich environments and seem to be capable of overcoming the problem of the *knowledge acquisition bottleneck* faced by knowledge engineers while designing the knowledge base of traditional expert systems. The trained link weights and node activation of the ANN are used to automatically generate the rules, either for later use in a traditional expert system, refining the initial domain knowledge, or providing justification/explanation in the case of an inferred decision. This automates and also speeds up the knowledge acquisition process. Such models help in minimizing human interaction and associated inherent bias during the phase of knowledge base formation and also reduce the possibility of generating contradictory rules. Fuzzy neural networks [4] can be used for the same purpose, and can also handle uncertainty at various stages.

The present article is the first of its kind to provide a detailed categorization of neuro–fuzzy rule generation algorithms based on their level of synthesis. Section II provides an overview of neuro–fuzzy hybridization, which is the oldest and most well-known methodology in soft computing. An exhaustive survey of rule generation in the fuzzy, neural, and neuro–fuzzy framework is presented in Section III, along with some hybridization involving genetic algorithms. This is followed in Section IV by a survey of rule generation in knowledge-based networks using neuro–fuzzy hybridization, rough sets and genetic algorithms. Section V provides a case study of a neuro–fuzzy rule generation algorithm with application to medical diagnosis. Section VI concludes the article.

II. Neuro-Fuzzy and Soft Computing

This section focuses on different aspects of neuro–fuzzy computing, keeping in mind the rich literature currently available in this field. Finally, the concept of soft computing is introduced.

A. Need for Neuro–Fuzzy Integration

Both neural networks and fuzzy systems are dynamic, parallel processing systems that estimate input–output functions. They estimate a function without any mathematical model and *learn from experience* with sample data. A fuzzy system adaptively infers and modifies its fuzzy associations from representative numerical samples. Neural networks, on the other hand, can *blindly* generate and refine fuzzy rules from training data [15]. Fuzzy sets are considered to be advantageous in the logical field, and in handling higher order processing easily. The higher flexibility is a characteristic feature of neural nets produced by learning and, hence, this suits data-driven processing better [16]. Hayashi and Buckley [17] proved that 1) any rule-based fuzzy system may be approximated by a neural net and 2) any neural net (feedforward, multilayered) may be approximated by a rule-based fuzzy system. This kind of equivalence between fuzzy rule-based systems and neural networks is also studied in [18]–[21]. Jang and Sun [22] have shown that fuzzy systems are functionally equivalent to a class of radial basis function (RBF) networks, based on the similarity between the local receptive fields of the network and the membership functions of the fuzzy system.

Fuzzy systems can be broadly categorized into two families. The first includes linguistic models based on collections of IF–THEN rules, whose antecedents and consequents utilize fuzzy values. It uses fuzzy reasoning and the system behavior can be described in *natural* terms. The *Mamdani* model [23] falls in this group. The knowledge is represented as

$$R^i: \text{If } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \cdots \text{and } x_n \text{ is } A_n^i \text{ then } y^i \text{ is } B^i \quad (1)$$

where $R^i (i = 1, 2, \cdots, l)$ denotes the ith fuzzy rule, $x_j (j = 1, 2, \cdots, n)$ is the input, y^i is the output of the fuzzy rule R^i, and $A_1^i, A_2^i, \cdots, A_n^i, B^i (i = 1, 2, \cdots, l)$ are fuzzy membership functions usually associated with linguistic terms.

The second category, based on *Sugeno*-type systems [24], uses a rule structure that has fuzzy antecedent and *functional* consequent parts. This can be viewed as the expansion of piecewise linear partition represented as

$$R^i: \text{If } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \cdots \text{ and } x_n \text{ is } A_m^i, \text{ then } y^i = a_0^i + a_1^i x_1 + \cdots + a_n^i x_n. \quad (2)$$

The approach approximates a nonlinear system with a combination of several linear systems, by decomposing the whole input space into several partial fuzzy spaces and representing each output space with a linear equation. Such models are capable of representing both qualitative and quantitative information and allow relatively easier application of powerful learning techniques for their identification from data. They are capable of approximating any continuous real-valued function on a compact set to any degree of accuracy [25]. This type of knowledge representation does not allow the output variables to be described in linguistic terms and the parameter optimization is carried out iteratively using a nonlinear optimization method.

However, there is a tradeoff between readability and precision. If one is interested in a more precise solution, then one is usually not so bothered about its linguistic interpretability. Sugeno-type systems are more suitable in such cases. Otherwise, the choice is for Mamdani-type systems. Two primary tasks of fuzzy modeling are structure identification and parameter adjustment. The first determines the input–output space partition, antecedent and consequent variables of IF–THEN rules, number of such rules, and initial positions of membership functions. The second identifies a feasible set of parameters under the given structure.

Neural networks, like fuzzy systems, are excellent at developing human-made systems that can perform information processing in a manner similar to our brain. In fact, the concept of ANN's was inspired by *biological neural networks* (BNN's), which are inherently nonlinear, highly parallel, robust and fault tolerant. A BNN is capable of 1) adapting its synaptic weights to changes in the surrounding environment; 2) easily handling imprecise, fuzzy, noisy, and probabilistic information; and 3) generalizing to unknown tasks. ANN's attempt to mimic these characteristics, often using principles from nervous systems to solve complex problems in an efficient manner. Fuzzy logic is capable of modeling vagueness, handling uncertainty, and supporting human-type reasoning.

A neural network is widely regarded as a black box that reveals little about its predictions. Extraction of rules from neural nets enables humans to understand this prediction process in a better manner. Rules are a form of knowledge that human experts can easily verify, transmit, and expand. Representing rules in *natural* form aids in enhancing their comprehensibility for humans. This aspect is suitably handled using fuzzy set-theoretic concepts.

The relation between neural networks and linguistic knowledge is bidirectional [26]. Therefore 1) neural network-based classification systems can be trained by numerical data and linguistic knowledge and 2) fuzzy rule-based classification systems can be designed by linguistic knowledge and fuzzy rules extracted from neural networks.

Fuzzy logic and neural systems have very contrasting application requirements. For example, fuzzy systems are appropriate if sufficient expert knowledge about the process is available, while neural systems are useful if sufficient process data are available or measurable. Both approaches build nonlinear systems based on bounded continuous variables, the difference being that neural systems are treated in a numeric quantitative manner, whereas fuzzy systems are treated in a symbolic qualitative manner. Fuzzy systems, however, exhibit both symbolic and numeric features. For example, when treated as collections of objects encapsulated by linguistic labels they lend themselves to symbolic processing via rule-based operations, while by referring to the definitions of the linguistic labels their membership functions are also suitable for numeric processing. Therefore, the integration of neural and fuzzy systems leads to a symbiotic relationship in which fuzzy systems provide a powerful framework for expert knowledge representation, while neural networks provide learning capabilities and exceptional suitability for computationally efficient hardware implementations. The significance of this integration becomes even more apparent by considering their disparities. Neural networks do not provide a strong scheme for knowledge representation, while fuzzy logic controllers do not possess capabilities for automated learning.

Neuro-fuzzy computing [4], [25], [27]–[31], which is a judicious integration of the merits of neural and fuzzy approaches, enables one to build more intelligent decision-making systems. This incorporates the generic advantages of artificial neural networks like massive parallelism, robustness, and learning in data-rich environments into the system. The modeling of imprecise and qualitative knowledge as well as the transmission of uncertainty are possible through the use of fuzzy logic. Besides these generic advantages, the neuro–fuzzy approach also provides the corresponding application specific merits.

B. Different Neuro–Fuzzy Hybridizations

Neuro-fuzzy hybridization [4], [27], [31] is done broadly in two ways: a neural network equipped with the capability of handling fuzzy information [termed *fuzzy-neural network* (FNN)] and a fuzzy system augmented by neural networks to enhance some of its characteristics like flexibility, speed, and adaptability [termed *neural-fuzzy system* (NFS)].

In an FNN, either the input signals and/or connection weights and/or the outputs are fuzzy subsets or a set of membership values to fuzzy sets, e.g., [7], [32]–[34]. Usually, linguistic values such as *low*, *medium*, and *high*, or fuzzy numbers or intervals are used to model these. Neural networks with fuzzy neurons are also termed FNN as they are capable of processing fuzzy information.

A neural-fuzzy system (NFS), on the other hand, is designed to realize the process of fuzzy reasoning, where the connection weights of the network correspond to the parameters of fuzzy reasoning, e.g., [14], [35]–[40]. Using the backpropagation-type learning algorithms, the NFS can identify fuzzy rules

and learn membership functions of the fuzzy reasoning. Usually for an NFS, it is easy to establish a one-to-one correspondence between the network and the fuzzy system. In other words, the NFS architecture has distinct nodes for antecedent clauses, conjunction operators, and consequent clauses. A fuzzy control system can also be termed as an NFS. There can be, of course, another blackbox-type NFS where a multilayer network is used to determine the input–output relation represented by a fuzzy system. For such a system the network structure has no such relation to the architecture of the fuzzy reasoning system.

An NFS should be able to *learn* linguistic rules and/or membership functions, or optimize existing ones. There are three possibilities [14]: 1) the system starts without rules, and creates new rules until the learning problem is solved. Creation of a new rule is triggered by a training pattern which is not sufficiently covered by the current rulebase; 2) the system starts with all rules that can be created due to the partitioning of the variables and deletes insufficient rules from the rulebase based on an evaluation of their performance; 3) the system starts with a rulebase with a fixed number of rules. During learning, rules are replaced by an optimization process.

The state of the art for the different techniques of judiciously combining neuro–fuzzy concepts involves synthesis at various levels. In general, these methodologies can be broadly categorized as follows [41]. Note that categories 1 and 3–5 relate to FNN's, while category 2 refers to NFS.

1) Incorporating fuzziness into the neural net framework: fuzzifying the input data, assigning fuzzy labels to the training samples, possibly fuzzifying the learning procedure, and obtaining neural network outputs in terms of fuzzy sets [7], [8], [33], [34], [42].
2) Designing neural networks guided by fuzzy logic formalism: designing neural networks to implement fuzzy logic and fuzzy decision-making, and to realize membership functions representing fuzzy sets [14], [35]–[40], [43]–[46].
3) Changing the basic characteristics of the neurons: neurons are designed to perform various operations used in fuzzy set theory (like fuzzy union, intersection, aggregation) instead of the standard multiplication and addition operations [47]–[51].
4) Using measures of fuzziness as the error or instability of a network: the fuzziness or uncertainty measures of a fuzzy set are used to model the error or instability or energy function of the neural network-based system [52].
5) Making the individual neurons fuzzy: the input and output of the neurons are fuzzy sets and the activity of the networks involving the fuzzy neurons is also a fuzzy process [32].

There are other kinds of categorizations for neuro–fuzzy models reported in literature [14], [53]. Buckley and Hayashi [53] have classified fuzzified neural networks as follows. Networks can possess 1) real number inputs, fuzzy outputs, and fuzzy weights; 2) fuzzy inputs, fuzzy outputs, and real number weights; 3) fuzzy inputs, fuzzy outputs, and fuzzy weights. Hayashi *et al.* [49] fuzzified the delta rule for multilayer perceptron (MLP) using fuzzy numbers at the input, output, and weight levels. But there were problems with the stopping rule. Ishibuchi *et al.* [54] incorporated triangular or trapezoidal fuzzy number weights, thereby increasing the complexity of the algorithm. Some of these problems have been overcome by Feuring *et al.* in [55]. All these fuzzy neural networks can, however, be grouped under categories 1 and 3 of our neuro–fuzzy integration methodology.

Nauck *et al.* [14] deal mainly with neuro–fuzzy control and suggest the following: 1) a *cooperative* system where the ANN and fuzzy system work independently of each other; the combination lies in determining certain parameters of a fuzzy system by an ANN and 2) a *hybrid neuro–fuzzy* system which implements a fuzzy system with an ANN; here one generates a homogeneous entity which cannot be divided into a fuzzy system or an ANN. In our terminology, both these combinations can be termed as an NFS under category 2 of the neuro–fuzzy integration.

C. Soft Computing

In traditional hard computing, the prime desiderata are precision, certainty, and rigor. By contrast, in soft computing the principal notion is that precision and certainty carry a cost and that computation, reasoning, and decision-making should exploit (wherever possible) the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth for obtaining low-cost solutions. This leads to the remarkable human ability of understanding distorted speech, deciphering sloppy handwriting, comprehending the nuances of natural language, summarizing text, recognizing and classifying images, driving a vehicle in dense traffic and, more generally, making rational decisions in an environment of uncertainty and imprecision. The challenge, then, is to exploit the tolerance for imprecision by devising methods of computation that lead to *an acceptable solution at low cost*. This, in essence, is the guiding principle of soft computing [56].

Soft computing is a consortium of methodologies that works synergetically and provides in one form or another flexible information processing capability for handling real life ambiguous situations. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve tractability, robustness, and low-cost solutions. The guiding principle is to devise methods of computation that lead to an acceptable solution at low cost by seeking for an approximate solution to an imprecisely/precisely formulated problem. The neuro–fuzzy approach, which provides flexible information processing capability by devising methodologies and algorithms on a massively parallel system for representation and recognition of real-life ambiguous situations forms, at this juncture, a key component of soft computing.

We can have approaches that exploit the benefits of all three soft computation tools, *viz.* fuzzy logic, ANN's and genetic algorithms (GA's), for rule generation. GA's [57] have found various applications in fields like pattern recognition, image processing and neural networks. In the area of ANN's, they have been used in determining the optimal set of connection weights as well as the optimal topology of layered neural networks. A fuzzy reasoning system can be implemented using a multilayer network, where the free parameters of the system can be learned

using GA's. Similarly, the parameters of an FNN can also be learned using GA's. Such systems are termed *neuro–fuzzy-genetic* [58]–[63]. It may be mentioned in this connection that *computational intelligence* [30], [64] is also a field related to artificial intelligence that uses soft computing tools like ANN's, fuzzy systems, and GA's in order to build intelligent systems with the capability of rule generation.

The theory of *rough sets* [65] has recently emerged as another major mathematical tool for managing uncertainty that arises from granularity in the domain of discourse, i.e., from the indiscernibility between objects in a set. The intention is to approximate a *rough* (imprecise) concept in the domain of discourse by a pair of *exact* concepts, called the lower and upper approximations. These exact concepts are determined by an *indiscernibility* relation on the domain, which, in turn, may be induced by a given set of *attributes* ascribed to the objects of the domain. The lower approximation is the set of objects definitely belonging to the vague concept, whereas the upper approximation is the set of objects possibly belonging to the same. These approximations are used to define the notions of *discernibility matrices*, *discernibility functions*, *reducts*, and *dependency factors*, all of which play a fundamental role in the reduction of knowledge.

Hybridizations for rule generation, exploiting the characteristics of rough sets, include the *rough-neuro* [66], *rough-neuro–fuzzy* [12], [67], *rough-neuro-genetic* [68], and *rough-neuro–fuzzy genetic* [69] approaches. The primary role of rough sets here is in managing uncertainty and extracting domain knowledge.

III. Rule Generation

Here we review the different fuzzy, neural and neuro–fuzzy models for rule generation, inferencing, and querying, along with their salient features. Sections III-A and III-B cover the fuzzy and neural approaches, respectively. This is followed in Sections III-C–III-E by different neuro–fuzzy approaches, indicating three types of hybridization as mentioned in Section II-B. Incorporation of GA's is also referred to in Sections III-A-1, III-B-2, and III-D-4 under *fuzzy–genetic*, *neuro–genetic* and *neuro–fuzzy–genetic* hybridization.

Let us first explain the significance of querying and rule generation, by referring to medical decision making. The models are generally capable of dealing with nonavailability of data, and can enquire the user for additional data when necessary. In the medical domain, for instance, data may be missing for various reasons; for example, some examinations can be risky for the patient or contraindications can exist, an urgent diagnostic decision may need to be made and some very informative but prolonged test results may have to be excluded from the feature set, or appropriate technical equipment may not be available. In such cases, the network can query the user for additional information only when it is particularly necessary to infer a decision. Again, one realizes that the final responsibility for any diagnostic decision always has to be accepted by the medical practitioner. So the physician may want to verify the justification behind the decision reached, based on personal expertise. This requires the system to be able to explain its mode of reasoning for any inferred decision or recommendation, preferably in rule form, to convince the user that its reasoning is correct.

A. Fuzzy Models

First of all, let us touch upon some of the approaches in fuzzy inferencing and rule generation before embarking on connectionist models. In the fuzzy classification rule described by Ishibuchi *et al.* [70], the partitioning is uniform, i.e., the regions continue to be split until a sufficiently high certainty of the rule, generated by each region, is achieved. Ishibuchi *et al.* extended this work later [71] by using an idea of sequential partitioning of the feature space into fuzzy subspaces until a predetermined stopping criterion is satisfied and studied its application for solving various pattern classification problems.

Wang and Mendel [72] developed a slightly different method for creating a fuzzy rulebase, made up of a combination of rules generated from numerical examples and linguistic rules supplied by human experts. The input and output domain spaces are divided into a number of linguistic subspaces. Human intervention is sought to assign degrees to the rules and conflicts are resolved by selecting those rules yielding the maximum of a computed measure corresponding to each linguistic subspace.

Rovatti and Guerrieri [73] have attempted to identify the correct rule structure of a fuzzy system when the target input–output behavior is sampled at random points. The assumption that a rule can either be included or excluded from the rule set is relaxed, and degrees of membership are exploited to achieve good approximation results. Defuzzification methodologies are then used to extract well-behaving crisp rule sets. Symbolic minimization is carried out to obtain a compact structure that captures the high-level characteristics of the target behavior. For other details, one may refer to standard literature [74]–[76].

1) With Genetic Algorithms: A fuzzy model, containing a large number of IF–THEN rules, is liable to encounter the risk of *overfitting* and, hence, poor generalization. The strong searching capacity of GA's has been utilized in *fuzzy–genetic* hybridization to circumvent this problem by [77] 1) determining membership functions with a fixed number of fuzzy rules [78], [79]; 2) finding fuzzy rules with known membership functions [80]; and 3) finding both membership functions and fuzzy rules simultaneously [77], [81], [82].

Ishibuchi *et al.* [82] select a small number of significant fuzzy IF–THEN rules to construct a compact and efficient fuzzy classification system. GA's are used to solve this combinatorial optimization problem, with an objective function for simultaneously maximizing the number of correctly classified patterns and minimizing the number of fuzzy rules.

Wang and Yen [77] have designed a hybrid algorithm that uses GA's for extracting important fuzzy rules from a given rulebase to construct a *parsimonious* fuzzy model with a high generalization ability. The parameters of the model are estimated using the *Kalman* filter.

B. Neural Models

Here we first consider the layered connectionist models by Gallant [1] and Saito and Nakano [83] used for rule generation in the medical domain. The inputs and outputs consist of *crisp*

variables in all cases. Generally the symptoms are represented by the input nodes while the diseases and possible treatments correspond to the intermediate and/or output nodes. The multilayer network described by Saito and Nakano [83] has been applied to the detection of *headache*. A patient responds to a questionnaire regarding the perceived symptoms and these constitute the input to the network.

The model by Gallant [1], dealing with *sacrophagal* problems, uses a linear discriminant network (with no hidden nodes) that is trained by the simple *pocket algorithm*. The absence of the hidden nodes and nonlinearity limits the utility of the system in modeling complex decision surfaces. Dependency information regarding the variables in the form of an adjacency matrix is provided by the expert. Every input variable x is approximated by three Boolean variables x'_1, x'_2, x'_3. Cell activation is discrete, taking on values $+1$, -1, or 0, corresponding to logical values of *true*, *false*, or *unknown*. Each cell computes its new activation y'_i as a linear discriminant function.

In [83], the system supplies the doctor with information regarding possible diagnoses on the basis of its output node values. Relation factors, estimating the strength of the relationship between symptom(s) and disease(s), are extracted from the network. Rules are generated from the changes in levels of input and output units; the connection weights are not involved in the process. Hence, this is a pedagogical approach. The search space is constrained by avoiding meaningless combination of inputs (symptoms) and restricting the maximum number of coincident symptoms to be considered. The rules are then used to allow patients to confirm the symptoms initially provided by them to the system, in order to eliminate noise from the answers. Nevertheless, the number of rules extracted for a relatively simple problem domain is exceedingly large [6].

Gallant's model [1] incorporates inferencing and forward chaining, confidence estimation, backward chaining, and explanation of conclusions by IF–THEN rules. In order to generate a rule, the attributes with greater inference strength (magnitude of connection weights) are selected and a conjunction of the more significant premises is formed to justify the output concept. Here the user can also be queried to supplement incomplete input information. During question generation, the system selects the unknown output variable whose *confidence* is maximum. Then it backtracks along the connection weights to find an unknown input variable, whose value is queried from the user. Rules are generated by traversing the trained connection weights as follows.

1) List all inputs that are known and have contributed to the ultimate positivity of a discriminant.
2) Arrange the list by decreasing absolute value of the weights.
3) Generate clauses for an IF–THEN rule from this ordered list.

Ishikawa [84] demonstrates the training of a network using *structural learning with forgetting*. An examination of the resultant simplified and nonredundant network architecture leads to easy extraction of rules. The positive weights are reduced and negative weights increased using a decay factor. A total of 8124 samples of mushrooms, with 22 attributes each, have been studied for the two-class (edible or poisonous) problem. The method selects two or four most relevant attributes. For the two-attribute case, odor and spore print color were found to be important. A sample of the antecedent part of an extracted rule for edible mushroom is (almond OR anise OR none) AND (spore print color $\neq$ green). Duch *et al.* [85] modified this algorithm by constraining the weights to $+1$, -1, or 0. This is supposed to result in the extraction of rules with more logical interpretation. They have also used a generalization of RBF networks for interpreting node functions as rules.

Fu [86], [87] has developed CFNet, whose activation function is based on the certainty factor (CF) model of expert systems. The CF model is a scheme for evidential reasoning in which a CF is assigned to a concept according to evidence observed. By mapping a CF model into an ANN, one can use the neural learning mechanism to help revise the former [87]. An analysis of the computational complexity of accurately discovering domain rules from a limited number of instances is provided. Rules can be *confirming* (positive) or *disconfirming* (negative). A rule's premise is limited to a conjunction of attributes. The presence of multiple rules with the same conclusion represents disjunction. Rules can be interpreted by an exact or inexact inference engine. In the latter case, a rule has to be attached with a number indicating the degree of belief in the conclusion given the premise and an attribute can also be assigned a weight. The activation function lies in the interval $[-1, 1]$, and the positive and negative inputs are combined separately. The output is implicitly quantizable in classification domains. The rule space is shrunk using pruning, resulting in a feasible complete search. Successive rule extraction is performed to circumvent the problem of generating rules from insufficient training data. In each learning cycle, some rules are learned and those positive instances, which can be explained by these rules, are removed. This cycle repeats until no more new rules can be further learned. Validation is performed on the test instances to determine the correct generalization capability. Performance of the model is compared with the decision tree-based rule generator C4.5 [88], KBANN by Towell and Shavlik [3], [11] (described in Section IV-A), and cascade ARTMAP by Tan [89] (Section IV-B). The decision tree approach (as in C4.5) is termed *monothetic* as it considers the utility of individual attributes one at a time, and may miss the case when multiple attributes are weakly predictive separately but become strongly predictive in combination. This problem can be overcome in neural approaches, also termed *polythetic*, like CFNet where multiple attributes are considered simultaneously [87]. Another advantage of CFNet is that it requires no initial domain knowledge and yet can perform reasonably well as compared to some knowledge-based networks [3], [11], [89]. Note that the knowledge-based model by Fu [10] (Section IV-A) is unable to extract most rules from a very large ANN and often generates only approximate rules. CFNet [86] overcomes some of the limitations of [10].

Setiono [90] has used a pruned network for extracting compact, meaningful rules, in terms of hidden unit activation. The activation are clustered into discrete values, and a process of splitting of hidden units and creation of new subnetwork is repeated until each hidden unit has only a small number of inputs connected to it. A penalty term augments a cross-entropy error

function, that is minimized to encourage weight decay and remove redundant weights. NeuroRule [90] can extract reasonably compact rule sets with high predictive accuracy. Unlike other algorithms [3], [10], this method does not require the activation values to be zero or one. The exponential complexity associated with the extraction of rules in search-based methods [10], [83] is avoided here. The accuracy and number of rules generated are better than those obtained by C4.5 [88]. Setiono and Leow [91] have recently developed a fast method for extracting rules from trained feedforward networks, that avoids the substantial overhead associated with pruning and retraining [90] while preserving the size and predictive accuracy of the rules. The algorithm uses information gain to identify relevant hidden units, and employs C4.5 to build a decision tree in terms of their activation values. Setiono has also reported [92] the extraction of M of N rules from a trained feedforward network whose weights and inputs are restricted to values in $\{-1, 1\}$. The rules are claimed to possess desirable qualities like accuracy, simplicity and fidelity.

Setiono and Liu [93] describe the extraction of oblique decision rules, corresponding to partition of the attribute space by hyperplanes that are not necessarily axis-parallel. This is claimed to result in the extraction of compact rules, with high predictive accuracy, from the trained network. The network is pruned and node activation discretized, followed by rule generation. The work is extended in Ref. [94] to generate oblique decision trees that can readily be translated into a set of rules. Since an oblique decision tree classifies patterns based on linear combinations of input attributes, the rules are more compact than that generated by an univariate tree over the same domain. Comparison is provided with other decision tree-based approaches, like C4.5 [88] and CART [95]. The compactness of these oblique rules is said to result in better rule comprehensibility and consistency.

Taha and Ghosh [13] have extracted rules along with certainty factors from trained feedforward networks. Input features are discretized and a linear programming problem is formulated and solved. A *greedy* rule evaluation mechanism is used to order the extracted rules on the basis of three performance measures, *viz.*, soundness, completeness, and false-alarm. A method of integrating the output decisions of both the extracted rulebase and the corresponding trained network is described, with a goal of improving the overall performance of the system. Comparison is provided with "NeuroRule" [90] and C4.5 [88].

Krishnan *et al.* [96] sort and order the input weights of a neuron, and prune the search space to determine those combinations of inputs that make the neuron active. This is used for rule generation from feedforward networks. Maire [97] backpropagates unions of polyhedra to design a new rule extraction technique. The fidelity of these rules is claimed to be very high.

1) With Recurrent Networks: Omlin and Lee Giles [98] use trained discrete-time recurrent neural networks to correctly classify strings of a regular language. Feedforward networks generally do not have the computational capabilities to represent recursive rules when the depth of the recursion is not known *a priori*. Such rules can, however, be conveniently represented by recurrent networks. Rules defining the learned grammar can be extracted from networks in the form of deterministic finite-state automata (DFA's) by applying clustering algorithms in the output space of recurrent state neurons. Starting from a defined initial network state that represents the root of the search space, the algorithm searches the equally partitioned output space of N state neurons in a breadth-first manner. A heuristic is used to choose among the consistent DFA's that model, that best approximates the learned regular grammar. Here the granularity of the underlying ANN within the DFA-extraction technique is at the level of ensembles of neurons, rather than individual neurons. Hence, the approach is not strictly decompositional. This is termed a *compositional* approach [5]. The extracted rules demonstrate high accuracy and fidelity and the algorithm is portable.

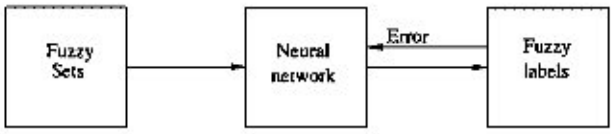

Fig. 1. Neural network implementing fuzzy classifier.

Vahed and Omlin [99] use a polynomial-time, symbolic learning algorithm to infer DFA's solely based on observation of a trained network's input–output behavior. This is a pedagogical approach and produces a minimal representation of the DFA. The clustering phase required in other recurrent net-based approaches [98] is eliminated, thereby increasing the fidelity of the extracted knowledge.

Chen *et al.* [100] have designed a recurrent network, that adapts from an analog phase to a discrete phase, for rule extraction. A modified objective function is used to accomplish the discretization process and logic learning. It is claimed that the network has significant advantage over other recurrent net-based approaches.

2) With Genetic Algorithms: Here we present rule generation methodologies in *neuro-genetic* hybridization. Fukumi and Akamatsu [101] have used an evolutionary algorithm for generating a compact neural network. Concepts of random optimization search and deterministic mutation are utilized for this purpose. This is followed by extraction of rules from the network.

Maeda and De Figuliredo [102] have designed a novel system for rule extraction of regulator control problems. The system employs a hybrid genetic search and reinforcement learning that requires neither supervision nor a reference model. The rules constitute a rule-based/table lookup structure capturing control actions. The extracted rules are claimed to be better than that generated by a neural controller trained with backpropagation.

C. Incorporating Fuzziness in Neural Net Framework

This is category 1 of the neuro–fuzzy hybridization described in Section II-B. A basic block diagram illustrating the process is provided in Fig. 1 [41].

As an illustration of the characteristics of layered fuzzy neural networks for inferencing and rule generation, the models by Hayashi [103], [104], and Hudson *et al.* [105] are described first. A *distributed single-layer perceptron-based* model trained with the *pocket algorithm* has been used [103], [104] for diagnosing *hepatobiliary disorders*. All contradictory training data are excluded, as these cannot be tackled by the

model. The input layer consists of fuzzy and crisp cell groups while the output is modeled only by fuzzy cell groups. The crisp cell groups are represented by m cells taking on two values in $\{(+1, +1, \cdots, +1), (-1, -1, \cdots, -1)\}$. Fuzzy cell groups, on the other hand, use binary m-dimensional vectors, each taking on values in $\{+1, -1\}$. Linguistic relative importance terms such as *very important* and *moderately important* are allowed in each proposition; linguistic truth values like *completely true*, *true*, *possibly true*, *unknown*, *possibly false*, *false*, and *completely false* are also assigned by the domain experts, depending on the output values. Provision is kept, using different linguistic truth values, for modeling the belonging of a pattern to more than one class. Extraction of fuzzy IF–THEN production rules is possible using a top–down traversal involving analysis of the node activation, bias and the associated link weights.

Hudson *et al.* [105] used a feedforward network for detecting *carcinoma of the lung*. The input nodes represent the data values for signs, symptoms, and test results (may be continuous or discrete) while the interactive nodes account for the interactions that may occur between these parameters. Information is extracted directly from the accumulated data and then combined with a rule-based system incorporating approximate reasoning techniques. The learning method is an adaptation of the *potential function* approach to pattern recognition and is used to determine the weighting factors as well as the relative strengths of rules for the two-class problem.

The fuzzy MLP [7] and fuzzy Kohonen network [8] are also used for linguistic rule generation and inferencing. Note that these models extend the concept of Gallant's method (which is derived for a perceptron) [1] to an MLP and a Kohonen network, by incorporating fuzzy set theoretic concepts at various levels. Here the input, besides being in quantitative, linguistic, or set forms, or a combination of these, can also be missing. The components of the input vector consist of membership values to the overlapping partitions of linguistic properties *low*, *medium*, and *high* corresponding to each input feature. This provides scope for incorporating linguistic information in both the training and testing phases of the said models and increases their robustness in tackling imprecise or uncertain input specifications. An n-dimensional feature space is decomposed into 3^n overlapping subregions corresponding to the three primary properties *low*, *medium*, and *high*. Although there is an associated increase in dimension and cost, one has to offset this with the specific gains achieved. The scheme enables the models to utilize more local information of the feature space and is found to be suitable in handling overlapping regions and highly nonlinear decision boundaries. Output decision is provided in terms of class membership values. The contribution of ambiguous or uncertain vectors to the weight correction is automatically reduced.

The connection weights of the trained network constitute the knowledge base for the problem under consideration. When partial information about a test pattern is presented at the input, the model either infers its category or queries the user for *relevant* information in the order of their relative importance (decided from the *learned* connection weights). If asked by the user, the network is capable of justifying its decision in rule form (relevant to a presented pattern) with the antecedent and consequent parts produced in linguistic and *natural* terms. The antecedent clauses are derived from the trained network by backtracking along *maximum-weighted* paths (through active nodes), whereas the consequent part is generated using a certainty measure. The effectiveness of the algorithms is demonstrated on vowel, synthetic, and medical data. An application of the fuzzy MLP to medical diagnosis [42] is described in detail in Section V.

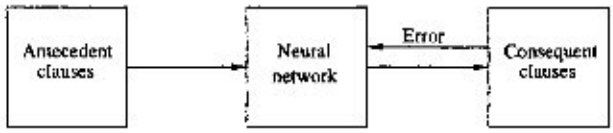

Fig. 2. Neural network implementing fuzzy logic.

Wang *et al.* [106] have used a fuzzy logic rule-based system to first determine a good feature set for the recognition of *Escherichia coli O157:H7*, a cause of serious health problems. Fuzzy membership functions are defined for each term set of each linguistic variable in the rules. The human inspired features of this reduced rule set are then incorporated in a multiple neural network fusion approach. The fuzzy integral is utilized in the fusion of the networks trained with different feature sets.

D. Designing Neural Net by Fuzzy Logic Formalism

Fig. 2 provides a block diagram [41], explaining the principle behind this form of hybridization (category 2, Section II-B). It encompasses both fuzzy reasoning and fuzzy control, where some IF–THEN rules are initially learned using training data and/or expert knowledge. Rules can later be generated (interpolated) for different input conditions. Integration with GA's is also considered briefly.

1) For Fuzzy Reasoning: The MLP-based approach to fuzzy reasoning reported by Keller and Tahani [35] falls under this category. It receives the possibility distributions of the antecedent clauses at the input, uses a hidden layer to generate an internal representation of the relationship, and finally produces the possibility distribution of the consequent at the output. The model is expected to function as an inference engine with each small subnetwork learning the functional input–output relationship of a rule. Trapezoidal possibility distributions, sampled at discrete points, are used to represent fuzzy linguistic terms and modifiers. The network is supposed to be able to extrapolate to other inputs (for a rule) following *modus ponens*. Conjunctive antecedent clauses are also modeled using separate groups of hidden nodes for each clause. Keller *et al.* [36] explicitly encode each rule in the structure of the network. A measure of disagreement between the input possibility distribution and the antecedent clause distribution is used at the *clause-checking* and *combination* layers to determine the uncertainty in the consequent part of the *fired* rule. Theoretical properties of various combination schemes are also investigated. Pal *et al.* [107] have reported an extension to this algorithm for computing an optimal value of α, the importance of the various antecedent clauses, which are supplied subjectively in Ref. [36]. The same membership values with more quantization levels are used at the antecedent and consequent levels. An improved network architecture is also proposed. This is extended in Ref. [108] by

using neural learning to find an optimal relation representing a set of fuzzy compositional rules of inference.

Ishibuchi *et al.* [45], on the other hand, use interval vectors to represent fuzzy input and output in an MLP. A backpropagation algorithm is applied on a cost function defined by α-level sets of actual and target fuzzy outputs, using the principles of interval arithmetic. Different fuzzy IF–THEN rules are interpolated from a few sample rules (used during training). Ishibuchi *et al.* [54], [109] have also reported learning methods of neural networks for utilizing expert knowledge represented by fuzzy IF–THEN rules. Both numeric and linguistic inputs are represented in terms of fuzzy numbers and intervals, which can be learned by the fuzzy neural network model. Here the connection weights are also modeled as fuzzy numbers represented by α-level sets. A generalization of this scheme for representing a fuzzy weight of any shape is reported in [110]. However, the use of interval arithmetic operations causes the computations to be complex and time-consuming. Since fuzzy numbers are propagated through the whole network, the computation time and required memory capacities are $2h'$ times of those in the traditional neural networks of comparable size, where h' represents the number of quantized membership grades.

The neural network-based fuzzy reasoning scheme by Takagi and Hayashi [44] is capable of learning the membership function of the IF part and determining the amount of control in the THEN part of the inference rules. The input data are clustered to find the best number of partitions corresponding to the number of inference rules applicable to the reasoning problem, with a single neural net block modeling one rule. The optimum number of cycles required is determined to avoid *overlearning* and the minimal number of input variables selected for inferring the control values. Takagi *et al.* [37] analyzed the identification error to improve the performance of the structured network based on fuzzy inference rules. The number of clusters determine the corresponding THEN parts to be added. The approach by Takagi *et al.* has been adapted in Japanese neuro–fuzzy consumer products [111]. Note that Mitra and Kuncheva [112] have developed a scheme to augment the IF parts of the relevant rules for the required pattern classification problem.

Nie [46] has developed a general and systematic approach for constructing a multivariable fuzzy model from numerical data using a self-organizing counterpropagation network. Both supervised and unsupervised algorithms are used. Knowledge can be extracted from the data in the form of a set of rules. This rulebase is then utilized by a fuzzy reasoning model. Moreover, an online adaptive fuzzy model updates the rulebase (in terms of connection weights) in response to the incoming data. The model claims a simple structure, fast learning speed, and good modeling accuracy. Chen and Xi [113] have developed an adaptive fuzzy inference system based on competitive learning. The input space is partitioned into local regions (clusters) and their decision boundaries determined. Fuzzy rules corresponding to each local region are then learned. A self-organizing learning algorithm has been used by Cai and Kwan [114] for designing a fuzzy inference network. The number of inference rules and their membership functions are automatically determined from the data during training. Learning speed is claimed to be fast. No prior information is required from experts while designing the system.

2) For Fuzzy Control: Wang and Mendel [38] represent a fuzzy system by a series of fuzzy basis functions, which are algebraic superposition of membership functions. Each such basis function corresponds to one fuzzy logic rule. An orthogonal least squares learning algorithm is utilized to determine the significant fuzzy logic rules (structure learning) and associated parameters (parameter learning) from the input–output training pairs. However, orthogonalization may lead to the production of incomprehensible and complex rules. Since a linguistic fuzzy IF–THEN rule from human experts can be directly interpreted, the fuzzy basis function network provides a framework for combining both numerical and linguistic information in a uniform manner.

Cho and Wang [115] describe an RBF-based adaptive fuzzy system to extract IF–THEN rules from sample data through learning. Different consequence types such as constant, first-order linear function, and fuzzy variable are modeled, thereby enabling the network to handle arbitrary fuzzy inference schemes. Neither is there an initial rulebase, nor does one need to specify in advance the number of rules required to be identified by the system. Fuzzy rules are generated, as and when needed, by recruiting basis function units.

Shann and Fu [116] have designed a layered network for learning rules of fuzzy control systems. The network is pruned to delete redundant rules and generate a concise fuzzy rulebase. The network developed by Horikawa *et al.* [117] is based on the truth space approach for automatic acquisition of fuzzy rules. The fuzzy variables in the consequent are labeled according to their linguistic truth values represented as fuzzy sets. Bastian [118] has introduced defuzzification weights to the overlapping areas of the consequent to control the linearity/nonlinearity at the transition between fuzzy logic rules. These weights are learned by a feedforward ANN. This can also be categorized as a cooperative neuro–fuzzy system according to the methodology of Nauck *et al.*

ANFIS by Jang [39] implements a Sugeno-like fuzzy system [24] in a five-layer network structure. Backpropagation is used to learn the antecedent membership functions, while least mean squares algorithm determines the coefficients of the linear combinations in the consequent of the rule. Here the min and max functions in the fuzzy system are replaced by differentiable functions. The rulebase must be known in advance, as ANFIS adjusts only the membership functions of the antecedent and consequent parameters. ANFIS can be easily implemented by flexible neural network simulators, and hence is attractive for application purposes. However, the learning algorithm being computationally expensive it is important to have an efficient implementation. Moreover, it is difficult for the model to handle high-dimensional problems, as this leads to a large number of input partitions, rules, and, hence, consequent parameters. The structure of ANFIS ensures that each linguistic term is represented by only one fuzzy set.

The neuro–fuzzy model designed by Chak *et al.* [119] can locate its rules and optimize their membership functions by competitive learning and Kalman filter algorithm. The key feature is that a high-dimensional fuzzy system can be implemented with fewer rules than that required by a conventional Sugeno-type model. This is because the input space partitions are unevenly

distributed. The network can be implemented in real time. Juang and Lin [120] have developed a self-constructing neural fuzzy inference network with on-line learning ability. Initially there are no rules, but they are created and adapted as learning proceeds via simultaneous structure and parameter identification. The input space is partitioned in a flexible way, using clustering, to identify the antecedents. The consequent is generated initially by clustering, followed by incremental learning using a projection-based correlation measure. Linear transformations are learned for each input variable, enabling the network to model fewer rules with higher accuracy. Kuo and Cohen [121] use a self-organizing and self-adjusting fuzzy model for manufacturing process control. The inputs and outputs are partitioned by Kohonen's feature mapping and the premise and consequence parameters are updated using backpropagation. The training parameters are dynamically updated using fuzzy models, leading to an acceleration in speed of learning. The self-organizing stage determines the initial position and shape of each membership function at the antecedent and the control action at the consequent. Backpropagation is then used to tune these parameters.

GARIC by Berenji and Khedkar [40] uses a differentiable *soft minimum* function to implement a fuzzy controller. A complex supervised learning procedure is used. All the models based on Sugeno-type systems are sometimes not as easy to interpret, as are Mamdani-type fuzzy systems. They are therefore more suited to applications where interpretation is not as important as performance. Initialization using prior knowledge is also not as easy as compared to models implementing Mamdani-type fuzzy systems. Berenji and Khedkar later developed a new architecture to control dynamic systems [122]. This model is capable of starting with approximate prior knowledge, which is refined using reinforcement learning.

Nauck *et al.* [14], [123] have developed NEFCON, NEFCLASS, and NEFPROX using a generic fuzzy perceptron to model Mamdani-type [23] neuro–fuzzy systems. The authors observe that a neuro–fuzzy system should be easy to implement, handle and understand. Fuzzy systems are designed to exploit the tolerance for imprecision, and hence should not concentrate on generating the *exact* solution. Reinforcement learning is found to be more suitable than supervised learning for handling control problems. The learning procedure uses a fuzzy error, and can operate both on fuzzy sets and rules. The system is claimed to be simple and highly interpretable. This is suitable in providing support to users during decision-making. Unlike the ANFIS model, NEFPROX offers a method of structure learning. The knowledge base of the fuzzy system is implicitly given by the network structure. The input units assume the task of the fuzzification interface, the inference logic is represented by the propagation functions, and the output unit is the defuzzification interface. The incremental rule learning algorithm can create a rulebase from scratch by adding rule after rule or can also operate on prior knowledge.

Reinforcement learning has also been used by Jouffe [124] to tune online the consequent part of fuzzy inference systems. The only information available for learning is the system feedback, which describes in terms of reward and punishment the task the fuzzy agent has to realize. At each time step, the agent receives a reinforcement signal according to the last action it has performed in the previous state. The problem involves optimizing not only the direct reinforcement, but also the total amount of reinforcement the agent can receive in the future.

3) With Recurrent Networks: Most neuro–fuzzy models reported so far deal with static input–output relationships. They are unable to process temporal input sequences of arbitrary length. Recurrent neural networks have the ability to store information over indefinite periods of time, can develop *hidden* states through learning, and are thus potentially useful for representing recursive linguistic rules. They are particularly well-suited for problem domains where incomplete or contradictory prior knowledge is available. In such cases, knowledge revision or refinement is also possible using recurrent nets. In fuzzy regular grammars, there is no question whether a production rule is applied; all applicable production rules are executed to some degree. For a given fuzzy grammar, there exists a fuzzy automaton. Fuzzy finite-state automata (FFA's) can model dynamic processes whose current state depends on the current input and previous states. Unlike deterministic finite-state automata (DFA's), FFA's are not in one particular state. Here each state is occupied to some degree defined by a membership function. Presently, FFA's are gaining significance as synthesis tools for a variety of problems. Based on their earlier design on encoding DFA's in discrete-time second-order recurrent neural networks [98], Omlin *et al.* have constructed an augmented recurrent network that encodes an FFA and recognizes a given fuzzy regular language with arbitrary accuracy [125]. The encoding methodology is empirically verified using randomly generated FFA's. As in [98], this approach of rule extraction can also be categorized as compositional.

Zhang and Morris [126] use a recurrent neuro–fuzzy network to build long-term prediction models for nonlinear processes. Process knowledge is used to initially partition the process operation into several local fuzzy operating regions and set up the initial fuzzification layer weights. Membership functions of fuzzy operating regions are refined through training, enabling the local models to learn. The global model output is obtained by center of gravity defuzzification involving the local models.

4) With Genetic Algorithms: A *neuro–fuzzy-genetic* hybridization has been reported by Yupu *et al.* [59]. GA's are used to search optimal fuzzy rules and membership functions for the neuro–fuzzy system. *A priori* knowledge from the designer is combined with the learning ability of the network to design an optimal fuzzy controller. This self-learning system uses the control performance index as the fitness function of the GA while searching for the network parameters.

Farag *et al.* [60] present a neuro–fuzzy system capable of handling both quantitative and qualitative knowledge. The learning involves first finding the initial parameters of the membership functions of the fuzzy model with Kohonen's self-organizing feature map algorithm. This is followed by the extraction of linguistic fuzzy rules. A multiresolutional dynamic GA is then used for optimized tuning of membership functions.

Ishibuchi *et al.* [62] use GA's for selecting a small number of significant linguistic rules from a large number of extracted rules. As in [82] (Section III-A.1), the objective is to maximize the number of correctly classified patterns while minimizing the number of selected rules.

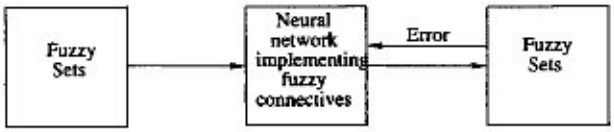

Fig. 3. Neural network implementing fuzzy connectives.

Wang and Archer [127] have introduced *ultrafuzzy* sets for modeling decision-making under conflict, using a modified version of backpropagation. In case of ultrafuzzy sets, the membership function takes on fuzzy values. Ultrafuzzy interval of certainty factor is modeled as the consequent of a rule. Two fuzzy membership functions termed as *participation* and *moderation* functions, falling in the ultrafuzzy interval, are developed based on the well-known *plausibility* and *belief* functions [128]. The concept of plausibility and belief functions is used to construct conflict measures, which help in explaining the compromise phenomena observed in decision-making. This fuzzy decision-making model is capable of cumulating human knowledge and is claimed to be useful for maintaining consistency while making decisions. ♣

Chow *et al.* [129] have introduced an interesting neuro–fuzzy method for enforcing heuristic constraints on membership functions, while extracting knowledge in the form of rules from limited information. In such cases, there is generally no ideal rulebase, which can be used to validate the extracted rules. Moreover, using output error measures to validate extracted rules is not sufficient as extracted knowledge may not make heuristic sense. This model ensures that the final membership functions conform to *a priori* heuristic knowledge, reduces the domain of search, and improves convergence speed.

E. Changing Basic Characteristics of Neurons

This pertains to category 3 of neuro–fuzzy hybridization described in Section II-B. Fig. 3 provides an overview of the whole process [41].

The work of Keller *et al.* [35], [36], which falls under the previous category, is extended [47] in the present framework. The model uses a fixed network architecture that employs parameterized families of operators, such as the generalized mean and multiplicative hybrid operators. The hybrid operator can behave as union, intersection, or mean operator for different sets of parameters, which can be learned during training. These networks possess extra predictable properties and admit a training algorithm that produces *sharper* inference results. Since the exact nature of each operator is learned by the network, the generated rules are capable of more accurately representing the input–output relationship.

Rhee and Krishnapuram [130] have reported a method for rule generation from minimal approximate fuzzy aggregation networks, using node activation and link weights. They estimate the linguistic labels and the corresponding triangular membership functions for the input features from the training data. Hybrid operators with compensatory behavior whose parameters can be learned during gradient descent to estimate the type of aggregation are employed at the neuronal level. Pruning of redundant features and/or hidden nodes helps in generating appropriate rules in terms of AND–OR operators that are represented by these hybrid functions. Zhang and Kandel [131] have also developed an adaptive fuzzy reasoning method using compensatory fuzzy operators. It is found to effectively learn fuzzy IF–THEN rules from both well- and ill-defined data. The efficiency of the compensatory learning algorithm can be enhanced by choosing an appropriate compensatory degree. Zurada and Lozowski [132] have applied T- and S-norms on input membership functions *negative*, *zero*, and *positive* to extract linguistic rules for pattern classes.

Mitra and Pal have used the fuzzy logical MLP for inferencing and rule generation [48]. The model consists of logical neurons employing conjugate pairs of t-*norms* T and t-*conorms* S, like *min–max* and *product–probabilistic sum*, in place of the *weighted sum* and *sigmoidal* functions of the conventional MLP. Various fuzzy implication operators are used to introduce different amounts of interaction during error backpropagation. The built-in AND–OR structure of the fuzzy logical MLP helps it to generate more appropriate rules in AND–OR form, expressed as disjunction of conjunctive clauses.

A neural network for formulating fuzzy production rules has been constructed by Yager [133]. Numerical information is used to find the preliminary partitioning of the input–output joint space. The linguistic variables associated with the antecedent and consequent parts of the rules are represented as weights in the neural structure. The membership values of these linguistic variables, modeled as fuzzy sets, can be learned. The determination of the firing level of a neuron is viewed as a measure of possibility between two fuzzy sets: the connection weights and the input. Unlike Keller *et al.* [36], here a self-organizing procedure is used to determine the structure and initial weights of the network, and obtain the nucleus of rules for a fuzzy knowledge base. This procedure is suitable in data-rich situations, where one is unable to find experts who can provide an organized description of the system. However, in the absence of expert opinions, the training data must be representative of the system's behavior and the unsupervised learning algorithm needs to be properly selected. Yager [134] has also employed neural modules for modeling the rules of fuzzy logic controllers with a combiner (using *min* or *product* functions). The various weights are learned and the importance of the antecedent clauses simulated.

Lin and Lu [135] have designed a five-layered network capable of processing both numerical and linguistic information. Fuzzy rules and membership functions are encoded for fuzzy inferencing. The inputs, outputs, and connection weights can be fuzzy numbers of any shape, represented by α-level fuzzy sets. Min and max operators are used to perform condition matching of fuzzy rules and integration of fired rules having the same consequent. Fuzzy supervised learning and fuzzy reinforcement learning are developed using interval arithmetic and fuzzy input–output pairs and/or linguistic information. The reinforcement signal from the environment involves linguistic information (fuzzy critic signal) such as *good*, *very good*, or *bad* instead of the normal numerical critic values like 0 (success) or 1 (failure). The system is used for reducing the number of rules in a fuzzy rulebase, and learning proper fuzzy control rules and membership functions.

The inferencing in the pseudo outer-product-based fuzzy neural network (POPFNN) [136] uses fuzzy rule-based systems

that employ the *truth value restriction* method. There are five layers, termed the input, condition, rulebase, consequence, and output layers. The fuzzification of the input and the defuzzification of the output are automatically accomplished. The learning process consists of three phases: self-organization, POP learning, and supervised learning. A self-organizing algorithm is employed in the first phase to initialize the membership functions of both the input and output variables by determining their centroids and widths. In the second phase, the POP algorithm is run in one pass to identify the fuzzy rules that are supported by the training set. The derived structure and parameters are then fine-tuned using the backpropagation algorithm.

A *cell recruitment* learning algorithm that is capable of forgetting previously learned facts by learning new information has been employed by Romaniuk and Hall [137] to build a neuro–fuzzy system for determining the *creditworthiness of credit applicants*. The network consists of *positive* and *negative collector cells* along with *unknown* and *intermediate* cells, and can handle *fuzzy* or *uncertain* data. Fuzzy functions such as *maximum, minimum*, and *negation* are applied at the neuronal levels depending on the corresponding bias values. This incremental learning algorithm can be used either in conjunction with an existing knowledge base or alone. Extraction of fuzzy IF–THEN rules is also possible.

IV. Using Knowledge-Based Networks

One of the major problems in connectionist/neuro–fuzzy design is the choice of the optimal network structure. This has an important bearing on any performance evaluation. Moreover, the models are generally very data-dependent, and the appropriate network size also depends on the available training data. Various methodologies developed for selecting the optimal network structure include growing and pruning of nodes and links, employing genetic search, and embedding initial knowledge in the network topology. The last approach—embedding initial knowledge—is usually followed in the case of knowledge-based networks. It is formally shown [138] that such knowledge-based networks require relatively smaller training set sizes for correct generalization. When the initial knowledge fails to explain many instances, additional hidden units and connections need to be added. The initial encoded knowledge may be refined with experience by performing learning in the data environment. The resulting networks generally involve less redundancy in their topology.

Incorporation of the concept of neuro–fuzzy integration at this level can also help in designing more efficient (intelligent) knowledge-based networks. The general role of fuzzy sets is to enhance ANN's by incorporating knowledge-oriented mechanisms. Preprocessing of training data leads to improvement in learning and/or enhanced robustness characteristics of the network. Prior knowledge, in the form of linguistic rules and membership functions, can be embedded into an ANN and thereby shorten the learning process. The blackbox aspect of an ANN is avoided in this manner and new knowledge can be extracted in rule form. Note that linguistic rules are more natural and easily interpretable. The heuristic, data-driven learning procedure operates on local information, causing only local modifications in the underlying fuzzy system. The fuzzy rules encoded within the system can be viewed as vague prototypes of the training data.

In this section, we embark on knowledge-based networks for performing inferencing and rule generation. We first describe the neural approaches. This is followed by different neuro–fuzzy knowledge-based approaches (hybridization categories 1 and 3, Section II-B). Next, we demonstrate how GA's are incorporated into this framework. Finally, some recent literature, using rough sets in this respect, is presented.

A. Connectionist Models

Let us consider here the models developed by Gallant [1], Fu [10], Shavlik *et al.* [3], [11], [139], Yin and Liang [140], and Lacher *et al.* [141]. The networks, other than that in [141], involve *crisp* inputs and outputs. The initial domain knowledge, in the form of rules, is mapped into the multilayer feedforward network topology, using binary link weights to maintain the semantics. Note that the rule generation aspect of Gallant's model [1] has already been discussed in Section III-B, as this is one of the seminal works in this direction. The other models are now described.

Yin and Liang [140] have employed a *gradually augmented-node* learning algorithm to incrementally build a dynamic knowledge base capable of both acquiring new knowledge and relearning existing information. The rules are explicitly represented among the *condition nodes*, *rule nodes*, and *action nodes*, and the algorithm gradually builds the multilayer feedforward network. The network structure is changed dynamically according to the new environment or through human intervention. This connectionist incremental model has been applied to the design of an *animal identification system*. In Fu's model [10] hidden units and additional connections are introduced appropriately when the network performance stagnates during training using backpropagation. Weight decay, pruning of weights, and clustering of hidden units are incorporated to improve the generalization of the network.

Towell and Shavlik [11] have designed a hybrid learning system KBANN, and applied it to problems of molecular biology. Disjunctive rules are rewritten as multiple conjunctive rules while mapping into the network structure. Nodes and links are incorporated, on instructions from the user, to augment the knowledge-based module. It is primarily a theory *refinement* system that is capable of pruning an inserted rule set, but not capable of adding new rules. It is largely topology preserving and assumes that the initial domain theory is basically correct and nearly complete. Learning or evolution of new knowledge, as a distributed representation, is not encouraged here [5].

An expansion of the network guided by both the domain theory and training data has been reported in TopGen by Opitz and Shavlik [139]. Dynamic additions of hidden nodes are made at the best place by heuristically searching through the space of possible network topologies, in a manner analogous to the adding of rules and conjuncts to the symbolic rulebase. This approach uses a specialized ANN architecture with a specialized training algorithm. It generates sparser rule sets as compared to KBANN and overcomes the latter's limitation of

not being able to extend a relatively weak initial domain theory. The computational expense is justified in terms of the human expert's willingness to wait for an extended period of time for better predictive accuracy. TopGen decreases false negatives by adding new rules, decreases false positives by adding new nodes to the network, and uses weight decay to preserve useful knowledge. The network which generalizes best on the corresponding validation set is selected as the best network.

A way of using the knowledge of the trained neural model to extract the revised rules for the problem domain is described by Fu [10] and Towell and Shavlik [3]. Knowledge, in the form of rules in disjunctive normal form, is encoded into the network. The other links represent low-weighted connections, allowing subsequent refinement. The network is trained through error backpropagation. This is followed by rule extraction. It is assumed that the neurons have binary inputs and hard-limiting activation functions, and the method of rule extraction searches for constraints on the inputs of a given neuron such that the weights are $>\theta$ (bias). An exhaustive search for firing conditions follows. Each firing corresponds to a rule under a certain combination of inputs. All combinations are checked, such that the rule search becomes a combinatorial task.

The *subset* algorithm [10] can be used by the network to improve the search complexity for the combination of firing conditions. Here one searches for any single weight exceeding the bias and rewrites all conditions so found as rules with single input variable. The search continues for increased size of sets until all sets have been explored and possibly rewritten as rules. The extracted rules are simple to understand and their size can be restricted by specifying the number of premises/antecedents to be considered. However, some of the problems associated with this algorithm are as follows [13]. It requires lengthy, exhaustive searches of size $O(2^k)$ for a hidden/output node with a fan-in of k. It extracts a large set of rules up to $\beta_p * (1 + \beta_n)$, where β_p and β_n are the number of subsets of positively and negatively weighted links, respectively. Some of the generated rules may be repetitive, as permutations of rule antecedents are not taken care of automatically. Moreover, there is no guarantee that all useful knowledge embedded in the trained network will be extracted.

The subset algorithm has been further modified in Towell and Shavlik [3] by the M *of* N algorithm for extracting meaningful rules. A general rule in this case is of the form: IF (*at least* M of the following N antecedents are true), THEN $\cdots$. The rationale is to find a group of links that form an equivalence class, whose members have similar effect (weight values) and can be used interchangeably with one another.

The steps of this algorithm involve *clustering* the weights of each neuron into groups, *averaging* their values to create equivalence classes, *eliminating* low-value weights if they have no effect on the sign of the total activation and *optimizing* by freezing the remaining weights and retraining the biases using the backpropagation algorithm. This is followed by *rule extraction*. Arithmetic is performed such that one searches for all weighted antecedents, which, when summed up, exceed the threshold value of a given neuron.

This algorithm has good generalization (accuracy), but can have degraded comprehensibility [6]. Note that the algorithm considers groups of links as equivalence classes, thereby generating a bound on the number of rules rather than establishing a ceiling on the number of antecedents. This approach differs from that of Saito and Nakano [83] (described in Section III-B), where a breadth-first search is employed to exhaustively find those input settings that cause the weighted sum to exceed the bias at a node. Even though the algorithms in [3], [10] are exponential, their inherent simplicity makes them extremely useful.

Lacher *et al.* [141] have designed event-driven, acyclic networks of neural objects called *expert networks*. There are regular nodes and operation nodes (for conjunction and negation). Input weights are hard wired, while the output weights of a node are adaptive. Antecedents of a disjunction in a rule are simplified to generate a set of individual rules before formulating the initial network architecture. Virtual rules are used to create potential connections for learning in order to overcome situations involving small initial set of rules. The backpropagation algorithm is modified to work in the event-driven environment, where both forward and backward signals propagate in *data-flow* fashion. The form of the rules (coarse knowledge) is tuned with the associated certainty factors (fine knowledge), and the resultant network trained for better performance.

B. Incorporating Fuzzy Sets

A brief survey on the knowledge-based networks involving fuzziness at different stages is provided here. The approaches in [9], [142]–[144] fall under category 1 of the fusion methodologies described in Section II-B, while those in [89], [145], and [146] can be grouped in category 3.

Knowledge extracted from experts in the form of membership functions and fuzzy rules (in AND–OR form) is used to build and preweight the neural net structure, which is then tuned using training data. Kasabov [142] uses three neural subnets—production memory, working memory, and variable binding space—to encode the production rules, which can later be updated. FuNN [143] is a five-layered feedforward architecture with the second layer calculating fuzzy input membership functions, the third layer representing fuzzy rules, the fourth layer calculating output membership functions, and the fifth layer computing output defuzzification. The network has features of both a neural network and a fuzzy inference machine.

Fuzzy signed digraph with feedback, termed *fuzzy cognitive map*, has been used by Kosko [144] to represent knowledge. Additive combination of augmented connection matrices are employed to include the views of a number of experts for generating the knowledge network. Kosko [15] interprets a fuzzy rule as an association between antecedent and consequent. Neural associative memory or bidirectional associative memory is used to store fuzzy rules. The weight of a rule is indicative of its importance.

Machado and Rocha [145] have used a connectionist knowledge base involving fuzzy numbers at the input layer, fuzzy AND at the hidden layers, and fuzzy OR at the output layer. The hidden layers chunk input evidences into clusters of information for representing regular patterns of the environment. The output layer computes the degree of possibility of each hypothesis. The initial network architecture is generated using *knowledge graphs* elicited from experts. The experts express their knowledge about each hypothesis of the problem domain by selecting

an appropriate set of evidences and building an acyclic weighted AND–OR graph (knowledge graph) to describe how these must be combined to support decision making.

Tan [89] has used a generalization of fuzzy ARTMAP [50], called *cascade ARTMAP*. It represents intermediate attributes and rule cascades of rule-based knowledge explicitly, and performs multistep inferencing. A major problem of using MLP to refine rule-based knowledge [3], [10] is the preservation of symbolic knowledge under the weight tuning mechanism of the backpropagation algorithm. Another limitation is that unless the initial rulebase is roughly complete, the initial network architecture may not be sufficiently rich for handling the problem domain. A rule insertion algorithm translates IF–THEN symbolic rules into cascade ARTMAP architecture. This knowledge can be refined and enhanced by the learning algorithm. During learning, new recognition categories (rules) can be created dynamically to cover the deficiency of the domain theory. This is in contrast to the static architecture of the standard slow learning backpropagation networks. Learning in cascade ARTMAP is match-based (not error-based); it does not wash away existing knowledge and the meanings of units do not shift. It relies on a specific architecture, *viz.* adaptive resonance theory mapping, which enables it to handle the *stability–plasticity* dilemma. The extracted rules involve discrete inputs and are of good quality. The algorithmic complexity is linear in the number of recognition categories. Results indicate that the performance is superior as compared to the KBANN [11], ID-3 (decision tree) and MLP.

Most of these models are mainly concerned with the encoding of initial knowledge by a fuzzy neural network followed by refinement during training. Extraction of fuzzy rules in this framework has been attempted [9], [89], [142], [145], [146]. Connection weights of FuNN, above a preset threshold, determine the *condition* or *action* elements in the extracted rules along with their corresponding *degrees of importance* and *confidence factors* [143]. Inference, inquiry, and explanation are possible during consultation with the expert in [145]. As the cascade ARTMAP [89] preserves symbolic rule form, the extracted rules can be directly compared with the originally inserted rules. These rules are claimed [89] to be simpler and more accurate than the M of N rules [3]. Besides, each extracted rule is associated with a confidence factor that indicates its importance or usefulness. This allows ranking and evaluation of the extracted knowledge.

Machado and Rocha [146] have also used an interval-based representation for membership grades (MGI) to allow reasoning with different types of uncertainty: vagueness, ignorance, and relevance. The model incorporates the facilities of incremental learning, inference, inquiry, censorship of input information, and explanation as in expert systems. The utility-based inquiry process permits significant reduction of consultation cost or risk and gives the system the common sense property possessed by experts when selecting tests to be performed. The ability to criticize input data when they disrupt a trend of acceptance or rejection observed for a hypothesis mimics the behavior of experts, who are often able to detect suspicious input data and either reject them or ask for their confirmation. The explanation algorithm provides responses to queries such as *how* a particular conclusion was reached or *why* a particular question was formulated. The network forms a set of pathways that compete to send the largest evidential flow to the output neuron representing the hypothesis. The structure of the winning pathway represents a chain of fuzzy pseudoproduction rules that can be presented to the user either in a graphical format or as English text.

Application of this algorithm has been made to the deforestation monitoring of the Amazon region, using Landsat-V satellite images. The classes considered are forest, savanna, water, deforested area, cloud, and shadow. Eighty-two numerical features of spectral, textural, and geometric nature were measured on each image segment (of spectrally homogeneous regions, generated by region growing). Fuzzy classification allows the modeling of complex situations such as transition phenomena (as in the regeneration of forest in a previously burned area) or multiple classification (as in the case of forest overcast by clouds).

A model by Mitra *et al.* [9], falling under category 1 of the neuro–fuzzy integration scheme, has been developed for classification, inferencing, querying, and rule generation. It is capable of generating both *positive* (indicating the belongingness of a pattern to a class) and *negative* (indicating its degree of *not* belonging to a class) rules in linguistic form to justify any decision reached. This is found to be useful for inferencing in ambiguous cases. The knowledge encoding procedure, unlike many other methods [10], [11], involves a nonbinary weighting mechanism. The *a priori* class information and the distribution of pattern points in the feature space are taken into account while encoding the crude domain knowledge from the data set among the connection weights. Fuzzy intervals and linguistic sets are used in the process. Each pattern class is modeled in terms of positive and negative hidden nodes. An estimation of the links connecting the output and hidden layers (in terms of the preceding layer link weights and node activation) is made. The network topology is then refined, using growing and/or pruning, thereby generating a near optimal network architecture. The knowledge-based network is shown to converge much earlier, resulting in more meaningful rules at this stage as compared to other models.

The trained knowledge-based network is used for rule generation in IF–THEN form. These rules describe the extent to which a test pattern belongs or does not belong to one of the classes in terms of antecedent and consequent clauses provided in natural form. Two rule generation strategies, as developed by Mitra *et al.* [9] are 1) pedagogical—treating the network as a blackbox and using the training set input (in numeric and/or linguistic forms) and network output (with confidence factor) to generate the antecedent and consequent parts and 2) decompositional—backtracking along maximal weighted paths using the trained net and utilizing its input and output activation (with confidence factor) for obtaining the antecedent and consequent clauses. The concept of generating *negative* rules and its implication to medical diagnosis is described in Section V-C. The model has been tested on *vowel*, synthetic, and medical data.

C. With Recurrent Networks

Omlin and Lee Giles [147] insert prior knowledge in the form of rules into recurrent networks for performing rule revision.

The inserted rules are compared with those in the DFA extracted from the trained network. It is claimed that the network is able to preserve the correct rules, while simultaneously adapting (through training) the incorrect inserted rules.

D. Incorporating Genetic Algorithms

Opitz and Shavlik [148] have used the domain theory of Towell and Shavlik [3], [11], as described in Section IV-A, to generate the knowledge-based network structure. Random perturbation is applied to create an initial set of candidate networks or *population*. A node is perturbed by either deleting it or by adding new nodes to it. Next, these networks are trained using backpropagation and placed back into the population. New networks are created by using crossover and mutation operators specifically designed to function on these networks. The algorithm tries to minimize the destruction of the rule structure of the crossed-over networks, by keeping intact nodes belonging to the same syntactic rule (i.e., the nodes highly connected to each other). The mutation operator adds diversity to a population, while still maintaining a directed heuristic search technique for choosing where to add nodes. In this manner, the algorithm searches the topology space in order to find suitable networks, which are then trained using backpropagation.

Evolutionary strategy is used by Jin *et al.* [61] to optimize a fuzzy rule system. The neuro–fuzzy hybridization employed here falls under category 2 (Section II-B). However this initial knowledge is tuned using evolutionary algorithms before being mapped to a radial basis function network for refinement. The number of fuzzy rules equals the number of hidden nodes in the network. A neural network regularization technique, termed adaptive weight sharing, is developed to extract understandable fuzzy rules from the trained network.

Kasabov and Woodford [149] have *evolved* the FuNN [143] (Section IV-B) as an associative memory for the purpose of dynamically storing and modifying a rulebase. Rules can be extracted and inserted from/into the system (EFuNN) in both on-line and off-line modes in a changing environment.

E. Incorporating Rough Sets

Let us first describe a model by Yasdi [66], which uses rough sets for the design of knowledge-based networks in the rough-neuro framework. The intention is to use rough sets as a tool for structuring the neural networks. The methodology consists of generating rules from training examples by using rough set-theoretic concepts and mapping them into a single layer of connection weights of a four-layered neural network. Attributes appearing as rule antecedent (consequent) become the input (output) nodes, while the dependency factors become the weight of the adjoining links in the hidden layer. The input and output layers involve nonadjustable binary weights. *Max*, *min*, and OR operators are modeled at the hidden nodes, based on the syntax of the rules. The backpropagation algorithm is slightly modified. However, the network has not been tested on any real life problem and no comparative study is provided to bring out the effectiveness of this hybrid approach.

Now we demonstrate a way of integrating rough sets and fuzzy-neural network for designing a knowledge-based system, where the theory of rough sets is utilized for extracting domain knowledge. In the rough–fuzzy MLP [12], [67], the extracted crude domain knowledge is encoded among the connection weights. This helps one to automatically generate an appropriate network architecture in terms of hidden nodes and links. Neuro-fuzzy hybridization of category 1 (Section II-B) is employed here. Methods are derived to model: 1) convex decision regions with single-object representatives and 2) arbitrary decision regions with multiple-object representatives. From the perspective of pattern recognition, this implies using a single prototype to model a (convex) decision region in case of method 1. For method 2, this means using multiple prototypes to serve as representatives of any arbitrary decision region. A three-layered fuzzy MLP is considered where the feature space gives the condition attributes and the output classes the decision attributes so as to result in a decision table. This table may be transformed, keeping the complexity of the network to be constructed in mind. Rules are then generated from the (transformed) table by computing relative reducts. The dependency factors of these rules are encoded as the initial connection weights of the fuzzy MLP. The knowledge encoding procedure involves a nonbinary weighting mechanism based on a detailed and systematic estimation of the available domain information. Moreover, the appropriate number of hidden nodes is automatically determined here.

Such a network is found to be more efficient than the conventional version [12]. The architecture of the network becomes simpler, due to the inherent reduction of the redundancy among the connection weights. The dependency rule for each class is obtained by considering the corresponding reduced attribute-value table. A smaller table leads to a simpler rule in terms of conjunctions and disjunctions, which is then translated into a network having fewer hidden nodes. The objective is to strike a balance by reducing the network complexity and reaching a *good* solution, perhaps at the expense of not achieving the *best* performance. While designing the initial structure of the fuzzy MLP, the union of the rules of all the pattern classes is considered. Here the hidden nodes model the conjuncts in the antecedent part of a rule, while the output nodes model the disjuncts. The appropriate number of hidden nodes is automatically generated by the rough set theoretic knowledge encoding procedure. On the other hand, both the fuzzy and conventional versions of the MLP are required to empirically generate a suitable size of the hidden layer(s). Banerjee *et al.* [12] further compared the rough–fuzzy MLP with other related techniques like decision trees.

A modular approach has been pursued by Mitra *et al.* [69] to combine the knowledge-based rough-fuzzy MLP subnetworks/modules generated for each class, using GA's. Dependency rules are extracted directly from real-valued attribute table consisting of fuzzy membership values. This helps in preserving all the class representative points in the dependency rules by adaptively applying a threshold that automatically takes care of the shape of the membership functions. An l-class classification problem is split into l two-class problems. The generated subnetworks are combined, and the final network evolved using a GA with restricted mutation operator that utilizes the inherent knowledge of the modular

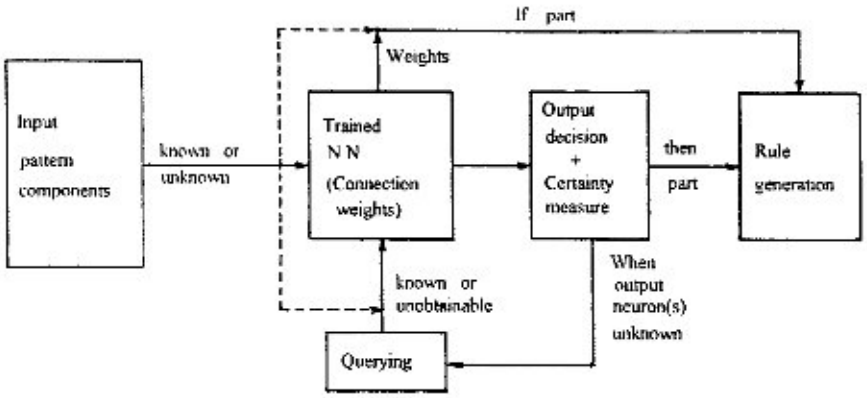

Fig. 4. Block diagram of inferencing and rule generation phases of fuzzy MLP.

structure. This *divide and conquer* strategy, followed by evolutionary optimization, is found to enhance the performance of the network. A compact set of more meaningful and less redundant (refined) rules are generated. This work is a novel *rough–neuro–fuzzy–genetic* hybridization in the soft computing framework. Application of the model has been made for medical diagnosis [68], [69].

V. Application to Medical Diagnosis

Here we describe a fuzzy MLP for rule generation [7] and demonstrate its effectiveness in medical diagnosis problems. This is followed by a short description of *negative* rule generation by a knowledge-based fuzzy MLP [9]. At the end of the training phase the network is supposed to have encoded the input–output information distributed among its connection weights. This constitutes the *knowledge base* of the desired decision-making system. Handling of imprecise inputs is possible and natural decision is obtained associated with a certainty measure denoting the confidence in the decision.

A. Model

The model is capable of

- inferencing based on complete and/or partial information;
- querying the user for unknown input variables that are key to reaching a decision;
- producing justification for inferences in the form of IF–THEN rules.

Fig. 4 gives an overall view of the various stages involved in the process of inferencing and rule generation.

The input can be in quantitative, linguistic, or set forms or a combination of these. It is represented as a combination of membership values to the three primary linguistic properties *low*, *medium*, and *high*, modeled as π functions [33]. The model can handle the linguistic hedges *very*, *more or less*, and *not*, as well as the set form modifiers *about*, *less than*, *greater than*, and *between*. *Missing* or *unknown* input features can also be taken care of.

The user can ask the system why it inferred a particular conclusion. The system answers with an IF–THEN rule applicable to the case at hand. Note that these IF–THEN rules are not represented explicitly in the knowledge base; they are generated by the *inferencing system*, by backtracking, from the connection weights as needed for explanation. As the model has already inferred a conclusion (at this stage), a subset of the currently known information is selected to justify this decision. The antecedent and consequent parts of the generated rules are in *natural* form using the linguistic modifiers and a certainty factor.

An input pattern $\boldsymbol{F}_p$ from the training set is presented to the input of the trained network and its output computed. To find the antecedent clauses of the rule, one may backtrack from the output layer to the input through the maximal weighted links. The path from node k in the output layer to node i_A in the input layer through node j in the hidden layer is maximal if

$$w^1_{kj} y^1_j + w^0_{ji_A} y^0_{i_A} = \max_m \left\{ w^1_{km} y^1_m + w^0_{mi_A} y^0_{i_A} \right\} \tag{3}$$

provided node activation $y^1_j \geq 0.5$, $y^0_{i_A} > 0.5$, and the *maximum* is computed over the index m. Here the *path length* from node k in the output layer to node j in the hidden layer is $w^1_{kj} y^1_j$, the superscript referring to the layer [9]. Only one node i_A corresponding to the three linguistic values of each feature F_i is considered so that

$$w^0_{ji_A} y^0_{i_A} = \max_{B \in \{L, M, H\}} w^0_{ji_B} y^0_{i_B} \tag{4}$$

where A and B correspond to *low* (L), *medium* (M), or *high* (H). The three-dimensional linguistic pattern vector, with or without modifiers [corresponding to the linguistic feature F_{i_A} computed by (4)], which is closest to the relevant three-dimensional part of pattern $\boldsymbol{F}_p$, is selected as the antecedent clause. This is done for all input features to which a path may be found by (3). The complete IF part of the rule is obtained by ANDing clauses corresponding to each of the features, e.g.,

$$\text{If } F_1 \text{ is } \textit{more or less } A \text{ and } F_2 \text{ is } \textit{not } A \text{ and } \cdots \text{ and } F_n \text{ is } \textit{very } A.$$

The consequent part of the corresponding IF–THEN rule is generated using a certainty factor bel^H_j. For the linguistic output form, one uses one of the following.

1) *Very likely* for $0.8 \leq bel^H_j \leq 1$.
2) *Likely* for $0.6 \leq bel^H_j < 0.8$.

TABLE I
RULE GENERATION AND QUERYING PHASES ON *HEPATO* DATA

Input features			y_{j1}^H	Rule generated		Initial rule
Initially supplied	Initially unknown	Query for		IF clause	THEN part	
$GOT > 100$ $GPT < 40$ $LDH > 700$ $GGT < 60$ $15 < BUN < 20$ $30 < MCH < 36$ *male*	MCV, $TBil$, $CRTNN$	–	.77	*GOT medium*, *GPT low*, *GGT very low*, *BUN very med.*	*Likely* PH	ALD *false*
$MCV > 100$ *male*	GOT, GPT, LDH, GGT, BUN, MCH, $TBil$, $CRTNN$	 GOT GPT MCH BUN LDH	.0 .1 .07 .26 .74 1.0	*GOT very low*, *GPT very low*, *LDH very low*, *MCV very high*, *MCH Mol med.*, *MCH Mol high*	*Very likely* LC	PH *false*
$GOT < 40$ *female*	GPT, LDH, GGT, BUN, MCV, MCH, $TBil$, $CRTNN$	 GPT GGT BUN LDH MCV	.04 .07 .62 .53 .67 .72	*GOT very low*, *GPT very low*, *LDH low*, *GGT very low*, *BUN Mol med.*, *MCV Mol med.*	*Likely* LC	PH *false*
$GOT < 40$ $40 < GPT < 100$ $MCV < 90$ *female*	LDH, GGT, BUN, MCH, $TBil$ $CRTNN$	 MCH	.75 .8	*GOT very low*, *MCV very low*, *MCH Mol med.*, *MCH Mol low*	*Likely* C	PH *false*
$LDH > 700$ $MCV < 90$ *male*	GOT, GPT, GGT, BUN, MCH, $TBil$, $CRTNN$	 GOT GPT MCH GGT BUN	.87 .91 .94 .94 .87 .84	*GOT low*, *GPT Mol low*, *BUN very med.*, *MCV very low*, *MCH Mol med.*	*Very likely* PH	LC *false*
$BUN > 20$ $MCV > 100$ *male*	GOT, GPT, LDH, GGT, MCH, $TBil$, $CRTNN$	 GOT GPT MCH	.85 .85 .9 .84	*GOT high*, *BUN Mol high*, *MCV high*, *MCH Mol med.*, *MCH Mol high*	*Very likely* PH	C *false*
$GOT < 40$ $40 < GPT < 100$ *male*	LDH, GGT, BUN, MCV, MCH, $TBil$, $CRTNN$	 MCH $CRTNN$ GGT BUN MCV	.01 .50 .77 .88 .78 .89	*GOT very low*, *BUN very low*, *MCV very high*, *MCH high*, *CRTNN low*, *CRTNN very med.*	*Very likely* ALD	PH *false*

3) *More or less likely* for $0.4 \leq bel_j^H < 0.6$.
4) *Not unlikely* for $0.1 \leq bel_j^H < 0.4$.
5) *Unable to recognize* for $bel_j^H < 0.1$.

A sample rule, in terms of input features F_1 and F_2, is as follows: If F_1 is *very medium* AND F_2 is *high* then *likely* class 1.

B. Medical Data

Medical diagnosis, or more specifically, the results of tests involve imprecision, noise, and individual difference. Often one cannot clearly distinguish the difference between normal and pathological values. Such test results cannot be precisely evaluated by crisp sets. Sometimes the patient can be simultaneously diagnosed as suffering in different degrees from multiple diseases. It is also more dangerous to classify a sick person as healthy than vice versa. Incorporation of fuzziness at the input and output of the neural network under consideration appears to be a good solution to such problems. Here one can simultaneously assign one or more finite nonzero membership values. An effective handling of a certain medical diagnosis problem involving hepatobiliary disorders [42] is demonstrated in this section. The data is available in http://www.isical.ac.in/~sushmita/patterns.

The data *hepato* consists of 536 patient cases of various hepatobiliary disorders. The nine input features are the results of different biochemical tests: glutamic oxalacetic transaminate (GOT; Karmen unit), glutamic pyruvic transaminase (GPT; Karmen unit), lactate dehydrase (LDH; iu/liter), gamma glutamyl transpeptidase (GGT; mu/ml), blood urea nitrogen (BUN; mg/dl), mean corpuscular volume of red blood cell (MCV; fl), mean corpuscular haemoglobin (MCH; pg), total bilirubin (TBil; mg/dl), and creatinine (CRTNN; mg/dl). The 10th feature corresponds to the sex of the patient and is represented in binary mode as (1,0) or (0,1). The hepatobiliary disorders alcoholic liver damage (ALD), primary hepatoma (PH), liver cirrhosis (LC), and cholelithiasis (C) constitute the four output classes. Table I depicts the rule generation and querying phases of the fuzzy MLP for a sample set of partially

known input features of the *hepato* data. Columns 3 and 4 refer, respectively, to the input feature supplied by the user after querying and the resulting output membership of the neuron corresponding to the disorder supported by the THEN part of the generated rule in column 6.

The last column of the table indicates the rules obtained from the initially supplied feature set in column 1. There were only two types of such rules in Hayashi's model [103]: the ones excluding a disease and the ones confirming a disease. The fuzzy MLP [42] resorts to querying and further updating to obtain rules that are more specifically indicative of a disease. Note that querying should be resorted to at a particular stage, and therefore querying is not required in all cases with a partial set of input features (e.g., see row 1 of Table I).

C. Negative Rules

It may sometimes happen that we are unable to classify a test pattern directly with the help of the *positive* rules (concerning its belonging to a class). In such cases, one proceeds by discarding some classes that are unlikely to contain the pattern, and thereby arrive at the class(es) to which the pattern possibly belongs. In other words, in the absence of positive information regarding the belonging of pattern $\boldsymbol{F}_p$ to class C_k, the complementary information about the pattern $\boldsymbol{F}_p$ not belonging to class C_k, is used. To handle such situations, *negative* rules are generated with the consequent part of the form *not in class* C_k, by backtracking from the output layer through the trained connection weights along *negative* hidden nodes corresponding to this class [9]. A sample *negative* rule generated for the medical data *hepato* is: If GOT is *low* AND GPT is *low* AND LDH is *very medium* AND GGT is *low* AND BUN is *low* AND MCV is *medium* AND MCH is *Mol medium* AND TBil is *low* AND CRTNN is *very medium* then the pattern is *not in* class ALD.

VI. Conclusions

We have provided an exhaustive survey of fuzzy, neural, and neuro–fuzzy rule generation algorithms. The neuro–fuzzy approach, symbiotically combining the merits of connectionist and fuzzy approaches, constitutes a key component of soft computing at this stage. To date, there has been no detailed and integrated categorization of the various neuro–fuzzy models used for rule generation. We have attempted to collect these under a unified soft computing framework.

Moreover, we have included both rule extraction and rule refinement in the broader perspective of rule generation. Rules learned and interpolated for fuzzy reasoning and fuzzy control have also been considered from this wider viewpoint.

Although the focus remained on neuro–fuzzy models, we also dealt with other fuzzy, neural, GA's, and rough set-based approaches to rule generation. Both feedforward and recurrent neural networks were considered. We concentrated on categorizing the different neuro–fuzzy approaches based on their level of synthesis. In the course of our study we noticed that other than the fuzzy perceptron [34], not much work has been reported in literature on the convergence analysis of neuro–fuzzy learning. This remains an open problem for future research.

Rule generation from fuzzy/nonfuzzy knowledge-based networks were found to result in more refined rules, as compared to both the initial crude domain knowledge used to encode them as well as those generated by networks involving no initial knowledge encoding. Finally, real-life application to medical diagnosis was provided.

References

[1] S. I. Gallant, "Connectionist expert systems," *Commun. ACM*, vol. 31, pp. 152–169, 1988.

[2] ——, *Neural Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1994.

[3] G. G. Towell and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Mach. Learn.*, vol. 13, pp. 71–101, 1993.

[4] S. K. Pal and S. Mitra, *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing*. New York: Wiley, 1999.

[5] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks," *IEEE Trans. Neural Networks*, vol. 9, pp. 1057–1068, 1998.

[6] R. Andrews, J. Diederich, and A. B. Tickle, "A survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowl.-Based Syst.*, vol. 8, pp. 373–389, 1995.

[7] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, Jan. 1995.

[8] ——, "Fuzzy self organization, inferencing and rule generation," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 608–620, 1996.

[9] S. Mitra, R. K. De, and S. K. Pal, "Knowledge-based fuzzy MLP for classification and rule generation," *IEEE Trans. Neural Networks*, vol. 8, pp. 1338–1350, 1997.

[10] L. M. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 173–182, 1993.

[11] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, pp. 119–165, 1994.

[12] M. Banerjee, S. Mitra, and S. K. Pal, "Rough fuzzy MLP: Knowledge encoding and classification," *IEEE Trans. Neural Networks*, vol. 9, pp. 1203–1216, 1998.

[13] I. A. Taha and J. Ghosh, "Symbolic interpretation of artificial neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 11, pp. 448–463, 1999.

[14] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro–Fuzzy Systems*. Chichester, U.K.: Wiley, 1997.

[15] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[16] H. Takagi, "Fusion technology of fuzzy theory and neural network—Survey and future directions," in *Proc. Int. Conf. Fuzzy Logic Neural Networks*, Iizuka, Japan, 1990, pp. 13–26.

[17] Y. Hayashi and J. J. Buckley, "Approximations between fuzzy expert systems and neural networks," *Int. J. Approx. Reas.*, vol. 10, pp. 63–73, 1994.

[18] J. J. Buckley and Y. Hayashi, "Numerical relationship between neural networks, continuous functions and fuzzy systems," *Fuzzy Sets Syst.*, vol. 60, no. 1, pp. 1–8, 1993.

[19] J. J. Buckley, Y. Hayashi, and E. Czogala, "On the equivalence of neural nets and fuzzy expert systems," *Fuzzy Sets Syst.*, vol. 53, no. 2, pp. 129–134, 1993.

[20] J. M. Benitez, J. L. Castro, and I. Requena, "Are artificial neural networks black boxes?," *IEEE Trans. Neural Networks*, vol. 8, pp. 1156–1164, 1997.

[21] J. J. Buckley and Y. Hayashi, "Hybrid neural nets can be fuzzy controllers and fuzzy expert systems," *Fuzzy Sets Syst.*, vol. 60, pp. 135–142, 1993.

[22] J. S. R. Jang and C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 156–159, 1993.

[23] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, pp. 1–13, 1975.

[24] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, 1985.

[25] J. J. Buckley and T. Feuring, *Fuzzy and Neural: Interactions and Applications*, ser. Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Physica-Verlag, 1999.

[26] H. Ishibuchi, M. Nii, and I. B. Turksen, "Bidirectional bridge between neural networks and linguistic knowledge: Linguistic rule extraction and learning from linguistic rules," in *Proc. IEEE Int. Conf. Fuzzy Syst. FUZZ-IEEE'98*, Anchorage, AK, May 1998, pp. 1112–1117.
[27] C. T. Lin and C. S. George Lee, *Neural Fuzzy Systems—A Neuro–Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[28] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press, 1996.
[29] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro–Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[30] W. Pedrycz, *Computational Intelligence: An Introduction*. Boca Raton, FL: CRC, 1998.
[31] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[32] S. C. Lee and E. T. Lee, "Fuzzy neural networks," *Math. Biosci.*, vol. 23, pp. 151–177, 1975.
[33] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, 1992.
[34] J. K. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, pp. 693–699, 1985.
[35] J. M. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks," *Int. J. Approx. Reas.*, vol. 6, pp. 221–240, 1992.
[36] J. M. Keller, R. R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets Syst.*, vol. 45, pp. 1–12, 1992.
[37] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their applications," *IEEE Trans. Neural Networks*, vol. 3, pp. 752–760, 1992.
[38] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, 1992.
[39] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.
[40] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, 1992.
[41] S. K. Pal and A. Ghosh, "Neuro-fuzzy computing for image processing and pattern recognition," *Int. J. Syst. Sci.*, vol. 27, pp. 1179–1193, 1996.
[42] S. Mitra, "Fuzzy MLP based expert system for medical diagnosis," *Fuzzy Sets Syst.*, vol. 65, pp. 285–296, 1994.
[43] E. Tsao, J. C. Bezdek, and N. R. Pal, "Fuzzy Kohonen clustering networks," *Pattern Recognit.*, vol. 27, pp. 757–764, 1992.
[44] H. Takagi and I. Hayashi, "Artificial neural network driven fuzzy reasoning," *Int. J. Approx. Reas.*, vol. 5, pp. 191–212, 1991.
[45] H. Ishibuchi, H. Tanaka, and H. Okada, "Interpolation of fuzzy if–then rules by neural networks," *Int. J. Approx. Reas.*, vol. 10, pp. 3–27, 1994.
[46] J. Nie, "Constructing fuzzy model by self-organizing counterpropagation network," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 963–970, 1995.
[47] J. M. Keller, R. Krishnapuram, and F. C.-H. Rhee, "Evidence aggregation networks for fuzzy logic inference," *IEEE Trans. Neural Networks*, vol. 3, pp. 761–769, 1992.
[48] S. Mitra and S. K. Pal, "Logical operation based fuzzy MLP for classification and rule generation," *Neural Networks*, vol. 7, pp. 353–373, 1994.
[49] Y. Hayashi, J. J. Buckley, and E. Czogala, "Fuzzy neural network with fuzzy signals and weights," *Int. J. Intell. Syst.*, vol. 8, no. 4, pp. 527–537, 1993.
[50] 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 Trans. Neural Networks*, vol. 3, pp. 698–713, 1992.
[51] W. Pedrycz, "A referential scheme of fuzzy decision making and its neural network structure," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 1593–1604, 1991.
[52] A. Ghosh, N. R. Pal, and S. K. Pal, "Self-organization for object extraction using multilayer neural network and fuzziness measures," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 54–68, 1993.
[53] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets Syst.*, vol. 66, pp. 1–13, 1994.
[54] H. Ishibuchi, K. Kwon, and H. Tanaka, "A learning algorithm of fuzzy neural networks with triangular fuzzy weights," *Fuzzy Sets Syst.*, vol. 71, pp. 277–293, 1995.
[55] T. Feuring, J. J. Buckley, and Y. Hayashi, "A gradient descent learning algorithm for fuzzy neural networks," in *Proc. IEEE Int. Conf. Fuzzy Syst. FUZZ-IEEE'98*, Anchorage, AK, May 1998, pp. 1136–1141.
[56] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Commun. ACM*, vol. 37, pp. 77–84, 1994.
[57] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[58] M. Su and H. Chan, "Extracting rules from composite neural networks for medical diagnostic problems," *Neural Process. Lett.*, vol. 8, pp. 253–263, 1998.
[59] Y. Yupu, X. Xiaoming, and Z. Wengyuan, "Real-time stable self-learning FNN controller using genetic algorithm," *Fuzzy Sets Syst.*, vol. 100, pp. 173–178, 1998.
[60] W. A. Farag, V. H. Quintana, and G. Lambert-Torres, "A genetic-based neuro–fuzzy approach for modeling and control of dynamical systems," *IEEE Trans. Neural Networks*, vol. 9, pp. 756–767, 1998.
[61] Y. Jin, W. von Seelen, and B. Sendhoff, "An approach to rule-based knowledge extraction," in *Proc. IEEE Int. Conf. Fuzzy Syst. FUZZ-IEEE'98*, Anchorage, AK, May 1998, pp. 1188–1193.
[62] H. Ishibuchi, M. Nii, and T. Murata, "Linguistic rule extraction from neural networks and genetic-algorithm-based rule selection," in *Proc. IEEE Int. Conf. Neural Networks*, Houston, TX, 1997, pp. 2390–2395.
[63] R. Sawa, Y. Makita, and M. Hagiwara, "Knowledge extraction and integration by artificial life approach," *J. Adv. Comput. Intell.*, vol. 3, no. 3, 1999.
[64] J. C. Bezdek, "What is computational intelligence?," in *Computational Intelligence Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds. New York: IEEE Press, 1994, pp. 1–12.
[65] Z. Pawlak, *Rough Sets, Theoretical Aspects of Reasoning about Data*. Dordrecht, The Netherlands: Kluwer, 1991.
[66] R. Yasdi, "Combining rough sets learning and neural learning method to deal with uncertain and imprecise information," *Neurocomputation*, vol. 7, pp. 61–84, 1995.
[67] S. Mitra, M. Banerjee, and S. K. Pal, "Rough knowledge-based network, fuzziness and classification," *Neural Comput. Applicat.*, vol. 7, pp. 17–25, 1998.
[68] P. Mitra, S. Mitra, and S. K. Pal, "Staging of cervical cancer with soft computing," IEEE Trans. Biomed. Eng., 2000, to be published.
[69] S. Mitra, P. Mitra, and S. K. Pal, "Evolutionary modular design of rough knowledge-based network using fuzzy attributes," Neurocomputation, 2000, to be published.
[70] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets Syst.*, vol. 52, pp. 21–32, 1992.
[71] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Efficient fuzzy partition of pattern space for classification problems," *Fuzzy Sets Syst.*, vol. 59, pp. 295–304, 1993.
[72] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, 1992.
[73] R. Rovatti and R. Guerrieri, "Fuzzy sets of rules for system identification," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 89–102, 1996.
[74] S. Abe and M. S. Lan, "Fuzzy rules extraction directly from numerical data for function approximation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 119–129, 1995.
[75] T. Hong and C. Lee, "Induction of fuzzy rules and membership functions from training examples," *Fuzzy Sets Syst.*, vol. 84, pp. 33–37, 1996.
[76] "Special issue on fuzzy diagnosis," *Artif. Intell. Med.*, vol. 16, no. 2, 1999.
[77] L. Wang and J. Yen, "Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and Kalman filter," *Fuzzy Sets Syst.*, vol. 101, pp. 353–362, 1999.
[78] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 46–53, 1993.
[79] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.
[80] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, San Diego, CA, July 1991, pp. 509–513.
[81] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 129–139, 1995.
[82] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy If–Then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, 1995.
[83] K. Saito and R. Nakano, "Medical diagnostic expert system based on PDP model," in *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, 1988, pp. I.255–I.262.
[84] M. Ishikawa, "Structural learning with forgetting," *Neural Networks*, vol. 9, pp. 509–521, 1996.

[85] W. Duch, R. Adamczak, and K. Grabczewski, "Extraction of logical rules from neural networks," *Neural Process. Lett.*, vol. 7, pp. 211–219, 1998.
[86] L. M. Fu, "A neural-network model for learning domain rules based on its activation function characteristics," *IEEE Trans. Neural Networks*, vol. 9, pp. 787–795, 1998.
[87] ——, "Learning in certainty-factor-based multilayer neural networks for classification," *IEEE Trans. Neural Networks*, vol. 9, pp. 151–158, 1998.
[88] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[89] A. H. Tan, "Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing," *IEEE Trans. Neural Networks*, vol. 8, pp. 237–250, 1997.
[90] R. Setiono, "Extracting rules from neural networks by pruning and hidden-unit splitting," *Neural Computat.*, vol. 9, pp. 205–225, 1997.
[91] R. Setiono and W. K. Leow, "FERNN: An algorithm for fast extraction of rules from neural networks," Appl. Intell., 2000, to be published.
[92] R. Setiono, "Extracting *M* of *N* rules from trained neural networks," IEEE Trans. Neural Networks, 2000, to be published.
[93] R. Setiono and H. Liu, "NeuroLinear: From neural networks to oblique decision rules," *Neurocomputation*, vol. 17, pp. 1–24, 1997.
[94] ——, "A connectionist approach to generating oblique decision trees," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, pp. 440–444, 1999.
[95] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth Brooks/Cole, 1984.
[96] R. Krishnan, G. Sivakumar, and P. Bhattacharya, "A search technique for rule extraction from trained neural networks," *Pattern Recognit. Lett.*, vol. 20, pp. 273–280, 1999.
[97] F. Maire, "Rule extraction by backpropagation of polyhedra," *Neural Networks*, vol. 12, pp. 717–725, 1999.
[98] C. W. Omlin and C. Lee Giles, "Extraction of rules from discrete-time recurrent neural networks," *Neural Networks*, vol. 9, pp. 41–52, 1996.
[99] A. Vahed and C. W. Omlin, "Rule extraction from recurrent neural networks using a symbolic machine learning algorithm," in *Proc. Int. Conf. Neural Inform. Processing ICONIP'99*, Perth, Australia, Nov. 1999, pp. 712–717.
[100] L. H. Chen, H. C. Chua, and P. B. Tan, "Grammatical inference using an adaptive recurrent neural network," *Neural Process. Lett.*, vol. 8, pp. 211–219, 1998.
[101] M. Fukumi and N. Akamatsu, "A new rule extraction method from neural networks," in *Proc. IEEE Int. Joint Conf. Neural Networks IJCNN'99*, Washington, DC, July 1999.
[102] Y. Maeda and J. P. De Figulerido, "Learning rule for neuro-controller via simultaneous perturbation," *IEEE Trans. Neural Networks*, vol. 8, pp. 1119–1130, 1997.
[103] Y. Hayashi, "Neural expert system using fuzzy teaching input and its application to medical diagnosis," *Inform. Sci. Applicat.*, vol. 1, pp. 47–58, 1994.
[104] ——, "A neural expert system with automated extraction of fuzzy if–then rules and its application to medical diagnosis," in *Advances in Neural Information Processing Systems*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. Los Altos, CA: Morgan Kaufmann, 1991, pp. 578–584.
[105] D. L. Hudson, M. E. Cohen, and M. F. Anderson, "Use of neural network techniques in a medical expert system," *Int. J. Intell. Syst.*, vol. 6, pp. 213–223, 1991.
[106] D. Wang, J. M. Keller, C. A. Carson, K. K. McAdoo-Edwards, and C. W. Bailey, "Use of fuzzy-logic-inspired features to improve bacterial recognition through classifier fusion," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 583–591, 1998.
[107] K. Pal, N. R. Pal, and J. Keller, "Some neural net realizations of fuzzy reasoning," *Int. J. Intell. Syst.*, vol. 13, pp. 859–886, 1998.
[108] K. Pal and N. R. Pal, "A neuro–fuzzy system for inferencing," *Int. J. Intell. Syst.*, vol. 14, pp. 1155–1182, 1999.
[109] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy If–Then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 85–97, 1993.
[110] H. Ishibuchi, K. Morioka, and I. B. Turksen, "Learning by fuzzified neural networks," *Int. J. Approx. Reas.*, vol. 13, pp. 327–358, 1995.
[111] H. Takagi, "Applications of neural networks and fuzzy logic to consumer products," in *Industrial Applications of Fuzzy Logic and Intelligent Systems*, J. Yen, R. Langari, and L. A. Zadeh, Eds. Piscataway, NJ: IEEE Press, 1995, pp. 93–104.
[112] S. Mitra and L. I. Kuncheva, "Improving classification performance using fuzzy MLP and two-level selective partitioning of the feature space," *Fuzzy Sets Syst.*, vol. 70, pp. 1–13, 1995.
[113] J. Chen and Y. Xi, "Nonlinear system modeling by competitive learning and adaptive fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 231–238, May 1998.
[114] L. Y. Cai and H. K. Kwan, "Fuzzy classifications using fuzzy inference networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 334–347, 1998.
[115] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction," *Fuzzy Sets Syst.*, vol. 83, pp. 325–339, 1996.
[116] J. J. Shann and H. C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," *Fuzzy Sets Syst.*, vol. 71, pp. 345–357, 1995.
[117] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "A new type of fuzzy neural network based on a truth space approach for automatic acquisition of fuzzy rules with linguistic hedges," *Int. J. Approx. Reas.*, vol. 13, pp. 249–268, 1995.
[118] A. Bastian, "Handling the nonlinearity of a fuzzy logic controller at the transition between rules," *Fuzzy Sets Syst.*, vol. 71, pp. 369–387, 1995.
[119] C. K. Chak, G. Feng, and J. Ma, "An adaptive fuzzy neural network for MIMO system model approximation in high-dimensional spaces," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 436–446, 1998.
[120] C. Juang and C. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, 1998.
[121] R. J. Kuo and P. H. Cohen, "Manufacturing process control through integration of neural networks and fuzzy model," *Fuzzy Sets Syst.*, vol. 98, pp. 15–31, 1998.
[122] H. R. Berenji and P. S. Khedkar, "Using fuzzy logic for performance evaluation in reinforcement learning," *Int. J. Approx. Reas.*, vol. 18, pp. 131–144, 1998.
[123] D. Nauck and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets Syst.*, vol. 101, pp. 261–271, 1999.
[124] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 338–355, 1998.
[125] C. W. Omlin, K. K. Thornber, and C. Lee Giles, "Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 76–89, 1998.
[126] J. Zhang and A. J. Morris, "Recurrent neuro–fuzzy networks for nonlinear process modeling," *IEEE Trans. Neural Networks*, vol. 10, pp. 313–326, 1999.
[127] S. Wang and N. P. Archer, "A neural network based fuzzy set model for organizational decision making," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 194–203, May 1998.
[128] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
[129] M. Chow, S. Altug, and H. J. Trussell, "Heuristic constraints enforcement for training of and knowledge extraction from a fuzzy/neural architecture—Part I: Foundation," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 143–150, 1999.
[130] F. C. H. Rhee and R. Krishnapuram, "Fuzzy rule generation methods for high-level computer vision," *Fuzzy Sets Syst.*, vol. 60, pp. 245–258, 1993.
[131] Y. Q. Zhang and A. Knadel, "Compensatory neuro–fuzzy systems with fast learning algorithms," *IEEE Trans. Neural Networks*, vol. 9, pp. 83–105, Jan. 1998.
[132] J. M. Zurada and A. Lozowski, "Generating linguistic rules from data using neuro–fuzzy framework," in *Proc. 4th Int. Conf. Soft Comput.*, Iizuka, Japan, 1996, pp. 618–621.
[133] R. R. Yager, "Modeling and formulating fuzzy knowledge bases using neural networks," *Neural Networks*, vol. 7, pp. 1273–1283, 1994.
[134] ——, "Implementing fuzzy logic controllers using a neural network framework," *Fuzzy Sets Syst.*, vol. 48, pp. 53–64, 1992.
[135] C. Lin and Y. Lu, "A neural fuzzy system with linguistic teaching signals," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 169–189, 1995.
[136] R. W. Zhou and C. Quek, "POPFNN: A pseudo outer-product based fuzzy neural network," *Neural Networks*, vol. 9, pp. 1569–1581, 1996.
[137] S. G. Romaniuk and L. O. Hall, "Decision making on creditworthiness, using a fuzzy connectionist model," *Fuzzy Sets Syst.*, vol. 48, pp. 15–22, 1992.
[138] L. M. Fu, "Learning capacity and sample complexity on expert networks," *IEEE Trans. Neural Networks*, vol. 7, pp. 1517–1520, 1996.
[139] D. W. Opitz and J. W. Shavlik, "Dynamically adding symbolically meaningful nodes to knowledge-based neural networks," *Knowl.-Based Syst.*, vol. 8, pp. 310–311, 1995.
[140] H. F. Yin and P. Liang, "A connectionist incremental expert system combining production systems and associative memory," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 5, pp. 523–544, 1991.
[141] R. C. Lacher, S. I. Hruska, and D. C. Kuncicky, "Backpropagation learning in expert networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 62–72, 1992.
[142] N. K. Kasabov, "Adaptable neuro production systems," *Neurocomputation*, vol. 13, pp. 95–117, 1996.

[143] ——, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets Syst.*, vol. 82, pp. 135–149, 1996.
[144] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," *Int. J. Approx. Reas.*, vol. 2, pp. 377–393, 1988.
[145] R. J. Machado and A. F. Rocha, "A hybrid architecture for fuzzy connectionist expert systems," in *Intelligent Hybrid Systems*, A. Kandel and G. Langholz, Eds. Boca Raton, FL: CRC, 1992, pp. 136–152.
[146] ——, "Inference, inquiry, evidence censorship, and explanation in connectionist expert systems," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 443–459, 1997.
[147] C. W. Omlin and C. Lee Giles, "Rule revision with recurrent neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 8, pp. 183–188, 1996.
[148] D. W. Opitz and J. W. Shavlik, "Connectionist theory refinement: Genetically searching the space of network topologies," *J. Artif. Intell. Res.*, vol. 6, pp. 177–209, 1997.
[149] N. Kasabov and B. Woodford, "Rule insertion and rule extraction from evolving fuzzy neural networks: Algorithms and applications for building adaptive, intelligent expert systems," in *Proc. IEEE Int. Conf. Fuzzy Syst. FUZZ-IEEE'99*, Seoul, Korea, Aug. 1999, pp. III-1406–III-1411.

Sushmita Mitra (M'99) received the B.Sc. (honors) degree in physics and the B. Tech and M. Tech degrees in computer science from the University of Calcutta, India, in 1984, 1987, and 1989, respectively, and the Ph.D. degree in computer science from Indian Statistical Institute, Calcutta, India, in 1995.

From 1992 to 1994, she was with the European Laboratory for Intelligent Techniques Engineering, Aachen, Germany, as a German Academic Exchange Service (DAAD) Fellow. Since 1995 she has been an Associate Professor of the Indian Statistical Institute, Calcutta, which she joined in 1989. She was a Visiting Professor at Meiji University, Japan, in 1999. She coauthored *Neuro–Fuzzy Pattern Recognition: Methods in Soft Computing Paradigm* (New York: Wiley, 1999). Her interests include pattern recognition, fuzzy sets, artificial intelligence, neural networks, and soft computing.

Dr. Mitra was a recipient of the National Talent Search Scholarship (1978–83) from the National Council for Educational Research and Training, India, the IEEE TNN Outstanding Paper Award in 1994 and CIMPA-INRIA-UNESCO Fellowship in 1996.

Yoichi Hayashi (M'86–SM'00) received the B.E. degree in management science, and the M.E. and Dr. Eng. degrees in systems engineering, all from the Science University of Tokyo, Japan, in 1979, 1981, and 1984, respectively.

He joined Ibaraki University, Japan, as an Assistant Professor in 1986 and was a Visiting Professor at the University of Alabama at Birmingham for ten years. Currently, he is a Professor of computer science at Meiji University, Japan. He has published 130 papers in academic journals and international conference proceedings in the fields of computer and information sciences. His current research interest includes artificial neural networks, fuzzy logic, soft computing, expert systems, computational intelligence, data mining, database management, and medical informatics.

Dr. Hayashi is an Associate Editor of IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is a Member of the ACM, AAAI, IFSA, INNS, NAFIPS, IPSJ, and IEICE.