# Symbolic Interpretation of Artificial Neural Networks

ISMAIL TAHA and JOYDEEP GHOSH [1]

**Abstract**

*H*ybrid Intelligent Systems that combine knowledge based and artificial neural network systems typically have four phases involving domain knowledge representation, mapping of this knowledge into an initial connectionist architecture, network training and rule extraction respectively. The final phase is important because it can provide a trained connectionist architecture with explanation power and validate its output decisions. Moreover, it can be used to refine and maintain the initial knowledge acquired from domain experts. In this paper, we present three rule extraction techniques. The first technique extracts a set of binary rules from any type of neural network. The other two techniques are specific to feedforward networks with a single hidden layer of sigmoidal units. Technique 2 extracts partial rules that represent the most important embedded knowledge with an adjustable level of detail, while the third technique provides a more comprehensive and universal approach. A rule evaluation technique that orders extracted rules based on three performance measures is then proposed. The three techniques are applied to the iris and breast cancer data sets. The extracted rules are evaluated qualitatively and quantitatively, and compared with those obtained by other approaches.

**Index Terms**: rule extraction, hybrid systems, knowledge refinement, neural networks, rule evaluation.

## 1 Introduction

Several researchers have investigated the design of hybrid systems that combine expert and connectionist subsystems [40, 47, 7, 12, 11, 20]. The typical result is a Knowledge Based Neural Network (KBNN) system with theory refinement capabilities, usually involving four phases: (i) the rule base representation phase, where initial domain knowledge is extracted and represented in a symbolic format (e.g., a rule-based system) (ii) the mapping phase, where initial domain knowledge is mapped into an initial connectionist architecture (iii) the learning phase, where this connectionist architecture is trained by a set of domain examples, and (iv) the rule extraction phase, where the trained and thus modified connectionist architecture is mapped back into an updated rule-based system. KBNNs attempt to exploit the complementary properties of knowledge based and neural network paradigms to obtain more powerful and robust systems. HIA [40], KBANN [46, 26], RAPTURE [20] and KBCNN [7, 8] are examples of KBNN hybrid systems. Figure 1 sketches typical components of a KBNN system that combines rule-based and connectionist paradigms.

[1]Manuscript received ..., revised .... This research was supported in part by ARO contract DAAH04-94-G-0417, DAAH04-95-10494, and ATP grant #442. Ismail Taha was also supported by the Egyptian Government Ph.D. Fellowship. A preliminary version of this paper appears in *Proceedings of ANNIE'96*, St. Louis, November 1996.

Dr. Taha is with the Department of Computers and Operational Research, Military Technical Collage, Cairo, Egypt. Dr. Ghosh is with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712-1084. E-mails: ismail.taha@mailexcite.com, ghosh@ece.utexas.edu

Connectionist systems can alternatively be combined with fuzzy logic systems to obtain "NeuroFuzzy" hybrid systems. In this setting the neural network subsystem is typically used to adapt membership functions of fuzzy variables [5], or to refine and extract fuzzy rules [42, 41, 18]. Neural networks have also been used for refinement of initial theories expressed in other knowledge representation schemes such as Bayesian Belief networks [29].

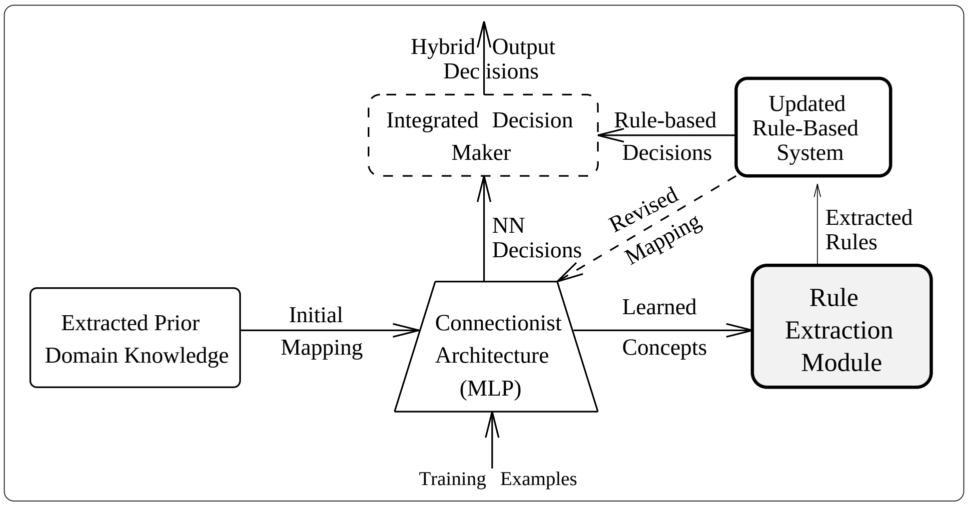


Figure 1: Typical components of a KBNN system that integrates knowledge based and connectionist paradigms.

Extraction of symbolic rules from trained Artificial Neural Networks (ANNs) is an important feature of comprehensive hybrid systems, as it helps to:

1. Alleviate the knowledge acquisition problem and refine initial domain knowledge.

2. Provide reasoning and explanation capabilities.

3. Support cross-referencing and verification capabilities.

4. Alleviate the "catastrophic interference" problem of certain ANNs [37]. For models such as MLPs[2], it has been observed that if a network originally trained on one task (data set) is subsequently trained on a different task (statistically different data set), then its performance on the first task degrades rapidly. In situations with multiple operating regimes, one can extract rules before the task or environment changes and thus obtain different rule sets for different environmental conditions. Together with a mechanism for detecting the current environment, this presents one solution to the "context discovery" and "context drift" problems.

[2]Multi-Layered Perceptron. This acronym is being used in this paper to signify a feedforward network with a single hidden layer of sigmoidal units, unless specified otherwise.

Other uses of rule extraction include improving acceptability of the product, transfer of knowledge to more suitable form, and induction of scientific theories.

This paper proposes three rule extraction techniques for KBNN hybrid systems. Also, it presents a simple rule evaluation procedure that orders rules obtained by any extraction approach, according to some performance criteria. A qualitative evaluation of the three new techniques and a comparison with some other approaches is also provided. The next section illustrates key issues in extracting rules from trained neural networks and summarizes some of the existing rule extraction techniques. Section 3 describes the proposed techniques and the rule evaluation procedure. In section 4, we present implementation results of these three techniques using an artificial problem, as well as the iris and breast cancer data sets. Section 5 compares the performance of rule sets extracted by our techniques with the rule sets extracted by some other approaches. In the concluding section we comment on the different rule extraction techniques, summarize the significance of the proposed techniques and point to future directions.

# 2 Rule Extraction

## 2.1 Issues

Several issues should be carefully considered while designing a rule extraction technique:

1. **Granularity of the explanation feature:** is the level of detailed hypotheses and evidence that the system can provide with each of its output decisions.

2. **Comprehensiveness of the extracted rules:** in terms of the amount of embedded knowledge captured by them. This directly determines the *fidelity* of the extracted rules in faithfully representing the embedded knowledge.

3. **Comprehensibility:** indicated by the number of rules and number of premises in each extracted rule from a trained network.

4. **Transparency of the extracted rules:** in terms of how well the decisions or conclusions can be explained.

5. **Generalization Capability:** on test samples.

6. **Portability:** is the capability of the rule extraction algorithm to extract rules from different network architectures.

7. **Modifiability:** is the ability of extracted rules to be updated when the corresponding trained network architecture is updated or retrained with different data sets.

8. **Theory Refinement Capability:** that can alleviate the knowledge acquisition bottleneck due to the incompleteness, inconsistency, and/or inaccuracy of initially acquired domain knowledge.

9. **Stability or Robustness:** is a measure of how insensitive the method is to corruptions in the training data or initial domain knowledge.

10. **Complexity and Scalability:** Computational issues that are relevant for large datasets and rule bases.

These issues, in addition to others, should be used to measure the quality and performance of rules extracted from trained neural networks as discussed in Section 5. Note that these issues also depend on the rule representation, insertion and network training methods used. Also, it is difficult to simultaneously optimize all of the above criteria. For example, a very comprehensive technique may extract too many rules, with some of them having many premises, thus degrading the robustness and comprehensibility of the resulting rule base.

## 2.2 Representative Rule Extraction Techniques

Research work in the area of extracting symbolic knowledge from trained ANNs has witnessed much activity recently. This subsection summarizes some of the existing approaches with emphasis on extracting rules from feedforward (specifically, MLP) architectures. An excellent literature review of different rule extraction approaches is a technical report written by Andrews et al. [1]. Also see the rule extraction home page at: *http://www.fit.qut.edu.au/ robert/rulex.html.*

### 2.2.1 Link Rule Extraction Techniques

The methodology behind most of the techniques for rule extraction from MLPs can be summarized in two main steps:

1. For each hidden or output node in the network, search for different combinations of input links whose weighted sum exceeds the bias of the current node.

2. For each of these combination generate a rule whose premises are the input nodes to this combination of links. All premises of a rule are conjuncted.

Either [27], KT [8] and Subset [45] are three notable rule extraction algorithms in this category. Some of the problems of KT and Subset algorithms are:

1. the size of the search algorithm is $O(2^l)$ for a hidden/output node with fan-in $= l$; assuming that network inputs are binary.

2. the algorithms extract a large set of rules, up to $\beta_p * (1 + \beta_n)$ where $\beta_p$ and $\beta_n$ are the number of subsets of positively-weighted and negatively-weighted links respectively.

3. some of the generated rules may be repetitive, as permutations of rule antecedents are not automatically taken care of.

4. there is no guarantee that all useful knowledge embedded in the trained network will be extracted.

However, the rules extracted from both algorithms are simple to understand. The size of the extracted rules can be limited by specifying the number of premises of the rules. Generally, the rules extracted by both KT and Subset algorithms are tractable especially in small application domains.

Based on the shortcomings of the Subset algorithm, Towell and Shavlik [45] developed another rule extraction algorithm called MofN, whose name the rule format that is used to represent the extracted rules: *If ("at least" M of the following N premises are true) then (the concept designated by the unit is true).* The rationale behind MofN is to find a group of links that form an equivalence class whose members have similar effect (due to similar weight values) and can be used interchangeably with one another. MofN extracts rules from the KBANN trained network through six main procedures. For problems like *"promoter recognition in DNA nucleotides"* for which it is a natural fit, rules extracted by MofN are significantly superior to rules extracted by other symbolic approaches such as C4.5, Either and LINUS [45]. NeuroRule is another rule extraction approach that uses different combinations of weighted links to extract rules [36]. It extracts rules from networks after pruning their architectures and then discretizing their hidden units activation values. Another algorithm, by Howes and Crook [14], first puts limits on hidden nodes activation values to satisfy an activation on output nodes of at least 0.9. After this step the algorithm searches for input combinations that satisfy the predetermined (constrained) hidden nodes activation values. If found, the algorithm extracts a rule for each combination. While all the previously mentioned approaches are design for networks with binary inputs, this algorithm has a proposed, though not efficient, extension to continuous valued inputs.

We categorize all the approaches mentioned in this subsection as *Link Rule Extraction (LRE) techniques* because they all first search for weighted links that cause a node (hidden or output) to be *"active"*. Then these combinations of weighted links are used to generate symbolic rules. Heuristic methods are commonly used in the *LRE* category to bound the search space for rules and to increase the comprehensibility of the extracted rules. Some researchers use the term *"decompositional methods"* to refer to *LRE* type techniques [1, 8].

RuleNet [22] and RULEX [2] are two examples of "localized LRE" techniques, tailored to networks with localized hidden units. RULEX extracts rules from a Constrained Error Back-Propagation (CEBP) MLP network, similar to Radial Basis Function (RBF) networks. Each hidden node in this CEBP network is localized in a disjoint region of the training examples. A distinctive feature of RULEX is that it controls the search space through its network while other approaches use heuristic measures to do the same. RuleNet, on the other hand, uses the idea of adaptive mixture of local experts [15] to train a localized ANN then extracts binary rules in a LRE approach.

#### 2.2.2 Black-box Rule Extraction Techniques

Another class of rule extraction approaches extracts rules from feedforward networks only by examining their input-output mapping behavior. An example of such a rule extraction approach is the algorithm developed by Saito and Nakano to extract medical diagnostic rules from a trained network [31]. BRAINNE [33], Rule-extraction-as-learning [6], and DEDEC [44] are other examples of extracting rules by investigating the input-output mapping of a trained network. In this paper we refer to this class as the *Black-box Rule Extraction (BRE)* category because rules are extracted regardless the type or the structure of the neural network. Another given name to this class of rule extraction techniques is *"pedagogical"* approaches [2]. For example, DEDEC extracts rules by ranking the inputs of an ANN according to their importance (contribution) to the ANN outputs [44]. This ranking process is done by examining the weight vectors of the ANN, which puts DEDEC on the border between LRE and BRE techniques. The next step in DEDEC is to cluster these ranked inputs and use each cluster to generate a set of optimal binary rules that describes the functional dependencies between the attributes of this cluster and the outputs of the ANN. DEDEC has been implemented using a standard feedforward MLP and a Cascaded Correlation (CasCor) ANN. In spite of the LRE nature of its ranking procedure, DEDEC is classified as a BRE since its main theme is to extract rules based on the input-output mapping.

#### 2.2.3 Extracting fuzzy rules from ANNs:

Research in the area of Fuzzy Logic Neural Networks (FLNN) or "NeuroFuzzy systems" is concerned with combining neural networks and fuzzy logic. Some FLNN systems include a fuzzy rule extraction module for refining fuzzy sets membership functions and explaining the trained neural network [13, 42, 41, 18].

#### 2.2.4 Extracting rules from recurrent networks

Recurrent networks have shown great success in representing finite state languages and deterministic finite state automata [10]. Omlin and Giles, [25] have developed a heuristic algorithm to extract grammar rules in the form of Deterministic Finite-state Automata (DFA) from discrete-time neural networks and specifically from second-order networks. Starting from a defined initial network state that represents the root of the search space, the DFA rule extraction algorithm searches the equally partitioned output space of N state neurons in a breadth-first fashion. The authors claim that the DFA rules extraction algorithm improves network generalization performance based on the stability of the internal DFA representation.

## 3 Proposed Rule Extraction Approaches

In this section, we introduce three different approaches to extracting rule bases from *trained neural networks*. The suitability of each approach depends on the network type, inputs, complexity, nature of application, the required quality of the extracted rules and some other factors as explained later. The first approach is a Black-box Rule Extraction technique. The second and the third approaches belong to the Link Rule Extraction category. Subsequently, an evaluation procedure and rule ordering algorithm is proposed.

## 3.1 First Approach (BIO-RE)

The first approach is a simple black box rule extraction technique, that is surprisingly effective within its (relatively) narrow domain of applicability. It is named **Binarized Input-Output Rule Extraction (BIO-RE)** because it extracts binary rules from any neural network trained with "binary" inputs, based on its input-output mapping. If original inputs are not binary, they have to be binarized using Equation 1.

$$y_i = \begin{cases} 1 & \text{if } x_i \geq \mu_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where: $x_i$ is the value of original input $X_i$, $\mu_i$ is the mean value of $X_i$, and $y_i$ is the corresponding binarized input.

The outline of the BIO-RE algorithm is as follows:

---

*For a well trained neural network do:*

1. *Obtain the network output* $O(Y) = \{o_j(Y) \mid o_j \in \{0,1\}\}$ *corresponding to each binary input pattern* $Y$*. If the number of input nodes is* **n** *then this conceptually requires* $2^n$ *input patterns. However, the problem specification may remove some combinations that will not occur.*

2. *Generate a truth table by concatenating each input* $Y$ *of step 1 and its corresponding output decision* $O(Y)$ *from the trained network. An output* $o_j(Y)$ *is set to 1 if its corresponding output node is active (above a threshold), otherwise it is 0.*

3. *Generate the corresponding boolean function (represented in the previously described binary rule format) from the truth table of step 2.*

---

Any available boolean simplification method can be used to perform step 3 of the BIO-RE algorithm (e.g. Karnaugh map, algebraic manipulation, or a tabulation method). We used Espresso[3] to generate the extracted rules [3]. Rules extracted by BIO-RE are represented in the format:

$If$ $[Not]$ *Input-Variable* $[And$ $[Not]$ *Input-Variable*$]^* \longrightarrow Consequent_j$

where: $[\cdot]$ is an optional term and $[\cdot]^*$ means that the term $[\cdot]$ can be repeated 0 or $n$ times. In terms of final rules, an extracted rule *"If* $Y_1$ *And Not* $Y_2$ *Then* $O_1$*"* is rewritten as *"If* $X_1 > \mu_1$ *And* $X_2 \leq \mu_2$ *Then* $O_1$*"* where: a "true" binary input (e.g., $Y_1$) is represented as *"*$X_1 > \mu_1$*"* and a "negated" binary input variable (e.g., $Y_2$) is represented as: *"*$X_2 \leq \mu_2$*"*, see Section 4.3 for examples.

[3]Espresso is a software package for logic design [30].

The BIO-RE approach is suitable when the input/output variables are naturally binary or when binarization does not significantly degrade the performance. Also the input size ($n$) should be small. Given that the above conditions are satisfied, BIO-RE has some advantages:

1. It allows the use of available logic minimization tools.
2. Extracted rules are optimal and cannot be simplified any further. Hence, no rewriting procedure is required.
3. The set of rules extracted by BIO-RE is comprehensive. They are quite understandable since all premises of the extracted rules are conjuncted.
4. It does not require any information about the internal structure of the network, the type of network or the training algorithm used.

Section 4 presents experimental results obtained when the BIO-RE algorithm was tested on three problems.

## 3.2 Second Approach (Partial-RE)

While BIO-RE is efficient, it may not be suitable if the number of inputs is large and/or the forced binarizing of these inputs severely degrades network performance. These deficiencies motivate our second rule extraction approach - a link rule extraction method called *Partial-RE*. The idea underlying Partial-RE algorithm is to identify dominant link combinations. For each hidden or output node, $j$, the positive and negative incoming links are first sorted in descending order of weight values into two sets. Then, starting from the highest positive weight (say $i$), the algorithm searches for individual incoming links that can cause the node $j$ to be active regardless of other input links to this node. If such a link exists, it generates a rule: "$If\ Node_i \xrightarrow{cf} Node_j$", where $cf$ represents the measure of belief in the extracted rule and is equal to the activation value of $node_j$ with this current combination of inputs. Values of certainty factors are computed by Equation 3. If an afferent node $i$ was found strong enough to activate node $j$, then this node is marked and cannot be used in any further combinations when inspecting node $j$. Partial-RE continues checking subsequent weights in the positive set until it finds one that cannot activate the current node $j$ by itself.

It is important to mention that Partial-RE assumes that all inputs have the same range, so that their effect on the hidden layer is simply determined by the weights. Therefore, the original input features may need to be scaled using Equation 2.

$$z_i = \frac{1.0}{1.0 + e^{-(\frac{x_i - \mu_i}{2.0\sigma_i})}} \tag{2}$$

where, $z_i \in (0, 1)$ is the corresponding scaled input value of the original input value $x_i$ and $\sigma_i$ is the standard deviation of input feature $X_i$. In Equation 2, $\sigma_i$ is multiplied by "2" to provide a wider distribution of input

$X_i$ ( a range of $\mu_i \pm 2\sigma_i$ will contain approximately 95 percent of the $X_i$ measurements [23] if $X_i$ is normally distributed).

If more detailed rules are required (i.e. the comprehensibility measure $p > 1$), then Partial-RE starts looking for combinations of two unmarked links starting from the first (maximum) element of the positive set. This process continues until Partial-RE reaches its terminating criteria (maximum number of premises in rule = $p$). Also, it looks for negative weights such their not being active allows a node in the next layer to be active, and extracts rules in the format: $If\ Not\ Node_g \xrightarrow{cf} Node_j$. Moreover, it looks for small combinations of positive and negative links that can cause any hidden/output node to be active, to extract rules such as: $If\ Node_i\ And\ Not\ Node_g \xrightarrow{cf} Node_j$, where the link between node $i$ and $j$ is positive and between $g$ and $j$ is negative. After extracting all rules, a rewriting procedure takes place. Within this rewriting procedure any premise that represents an intermediate concept (i.e. a hidden unit) is replaced by the corresponding set of conjuncted input features that causes it to be active. Final rules are written in the format: "$If\ X_i \geq \mu_i\ And\ X_g \leq \mu_g \xrightarrow{cf} Consequent_j$". See Table 2 and 6 for examples.

Partial-RE can be used efficiently in applications where the main objective of extracting rules from trained neural networks is to study the primary parameters that cause specific output decisions to be taken. In such cases, the cost of implementing the Partial-RE is low compared to the MofN algorithm if a small number of premises per rule is enough. By extracting only certain rules with small number of premises per rule we are reducing the combinatorial nature of the rule extraction process into one that is polynomial in $n$, where $n$ is the number of input features. Partial-RE examines small subsets $S_{js}$ of incoming links to a hidden or output node $j$, and extracts a rule if $\sum_{w_{ji} \in S_{js}} (w_{ji}x_i - \theta_j) \geq \Delta$, where $w_{ji}$ is the weight value of the link between input $x_i$ and hidden/output node $j$, $\theta_j$ is the threshold value of the node $j$, and $\Delta$ is a small positive value (between 0.1 and 0.3) called *certainty parameter*, whose value is chosen based on how "certain" we want the extracted rules to be. Partial-RE is conceptually close to subset-type algorithms[8, 45], not surprising since all are LRE algorithms. Note however that while subset algorithms reduce the size of the search space (all subsets of a given size) through branch-and-bound, Partial-RE achieves this through sorting by weight sizes, which is less computationally demanding. Partial-RE also avoids the rewriting procedure involved in subset algorithms, and is able to produce soft rules with associated measures of belief or certainty factors. The adjustable parameters $\Delta$ and $p$ (which determines the number of premises in a rule) provides flexibility to the Partial-RE algorithm. Several other differences in implementational details are described in [38].

Partial-RE is easily parallelizable, as nodes can be inspected concurrently. Experimental results show that Partial-RE algorithm is suitable for large size problems, since extracting all possible rules is NP-hard and extracting only the most effective rules is a practical alternative.

### 3.3 Third Approach (Full-RE)

Like the Partial-RE approach, Full-RE falls in the LRE category. It is notable because:

- It extracts rules with certainty factors from trained feedforward networks whose cells have monotonically increasing activation functions.
- It can extract rules from networks trained with continuous, normal, and binary inputs. Therefore, there is no restriction on the values that any input feature can take. This capability makes Full-RE a "universal" extractor.

For each hidden node $j$, Full-RE first generates *intermediate* rules in the format:
$If\ [\ (w_{j1} \cdot X_1 + w_{j2} \cdot X_2 + \cdots + w_{jn} \cdot X_n) >= \theta_j + \Delta] \xrightarrow{cf} Consequent_j$
where $\theta_j$ and $\Delta$ are the same as defined for Partial-RE.

Note that a range of $X_i$ values may satisfy an intermediate rule, and one would want to determine a suitable extremum value in such a range. To make this tractable, each input range is discretized into a small number of values that can be subsequently examined. Thus, each input feature $X_i \in (a_i, b_i)$ is discretized using $k$ intervals: $(D_i \in \{d_{i,0} = a_i, d_{i,1}, \cdots, d_{i,k-1}, d_{i,k} = b_i\})$, where: $d_{i,l-1}$ and $d_{i,l}$ are the lower and upper boundary values of interval $l$ of input $X_i$ respectively. Different discretization approaches can be exploited to compute discretization boundaries of input features $X_i$s [39, 16]. Full-RE uses the Chi2 [19] algorithm [4], a powerful discretization tool, to compute discretization boundaries of input features. When Full-RE finds more than one discretization value of an input $X_i$ that satisfies the intermediate rule (i.e., the rule has more than one feasible solution), it chooses the minimum or the maximum of these values based on the *sign* of the corresponding link. If $w_{ji}$ is negative then Full-RE chooses the maximum discretization value of $X_i$, otherwise it chooses the minimum value. However, all selected discretization values should satisfy the left hand side (the inequality) of the intermediate rule and the boundary constraints of all input features of this inequality.

After discretization, the following Linear Programming (LP) problem is formulated:
Minimize $\quad w_{j1}D_1 + w_{j2}D_2 + \cdots + w_{jn}D_n$
such that:
$w_{j1}D_1 + w_{j2}D_2 + \cdots + w_{jn}D_n \geq \theta_j + \Delta, \quad$ and
$D_i \in \{a_i, d_{i,1}, \cdots, d_{i,k-1}, b_i\}\ \forall_i, i = 1 \cdots n.$

Full-RE solves this LP problem by selecting the discretization boundaries of $X_i$s ($D_is$) that determine the feasible solution(s) of the intermediate rule and satisfy all given constraints. Values of single input features that can satisfy this LP problem regardless of other input features can be found easily by substituting $X_i$s of positive weights to node $j$ by their minimum values ($a_i$s) and $X_i$s of negative weights by their maximum values ($b_i$s). Higher combinations are found similarly by examining different combinations of discretized inputs and finding the edges of the feasible solution surface of the LP problem. Note that any linear programming tools can alternatively be used to solve this standard LP problem[5]. As an example, assume that a feasible solution

[4] We are thankful to Liu and Setiono for making their Chi2 source code available to us.

[5] We also used Mathematica to find out feasible solutions.

is at $x_1 = e_1$ and $x_2 = e_2$, and that the weights from input $X_1$ and $X_2$ on $node_j$ are positive and negative respectively. Then Full-RE extracts the following rule: "$If\ X_1 \geq\ e_1\ \ And\ \ X_2\ \leq\ e_2 \xrightarrow{cf} h_j$" where $a_i \leq e_i \leq b_i$, and $e_i$s are determined by the discretization process or the LP tool.

Full-RE computes certainty factors of extracted rules based on

$$cf = \begin{cases} \frac{1}{1+\exp^{(\sum_{i=1}^{n} w_{ji}\cdot x_i - \theta_j - \Delta)}} & \text{if act(j) is a sigmoid} \\ min(1, \sum_{i=1}^{n} w_{ji} \cdot x_i - \theta_j - \Delta) & \text{if act(j) is a linear threshold} \\ 1 & \text{if act(j) is a hard limiting and } \sum_{i=1}^{n} w_{ji} \cdot x_i \geq \theta_j + \Delta \end{cases} \tag{3}$$

Extracted rules between hidden and output nodes are represented in the same format of Partial-RE (e.g. $If\ h_1\ \ And\ h_2 \xrightarrow{cf} O_k$). Full-RE replaces each hidden node ($h_j$) in the previous rule by the left hand side of the rule(s) whose right hand side is $h_j$. The general format of final rules extracted by the Full-RE is:
$If$ *Simple-Boolean-Expression* [*AND Simple-Boolean-Expression*]$^* \xrightarrow{cf}$ $Consequent_j$
where:
*Simple-Boolean-Expression ::= Variable Operator Constant,*
*Operator ::=* $> \mid < \mid \geq \mid \leq$.
The $*$ means that the term [*AND Simple-Boolean-Expression*] may be repeated (up to n-1 times), the symbol | stands for an alternation (i.e. *operator* can take any of the four boolean operators) and $cf$ represent the certainty factor computed by Equation 3 for each extracted rule. The final rules extracted by Full-RE have the same format as Partial-RE except that each $\mu_i$ is replaced by one of the discretization boundaries (say $d_{i,l}$) selected by Full-RE as described earlier. See Table 3 and 7 for examples. Note that there is no restriction on the number of premises in the final rules extracted by the Full-RE. The only limitation applied is on the number of premises of rules between nodes of adjacent layers (e.g., number of premises in intermediate rules between input and hidden nodes or between hidden and output nodes). Note that when input features are binary, the discretization step is no longer required.

## 3.4 Rule Evaluation

The suitability of rule evaluation criteria dependents on the application domain. For example, in image understanding, one may be primarily interested in finding relational visual structures [4]. Also, one can define metrics for the complete rule set [17] or for individual rules. We will do the former in Section 5. In this section, we evaluate individual rules based on the following primary goals:

1. find the order of the extracted rules that maximizes their performance on the available data set, assuming an inference engine that inspects rules sequentially.

2. test the fidelity of the extracted rule-based system (i.e., its capability to mimic the embedded knowledge in the trained network).

3. measure how much knowledge is left unextracted from the internal structure of the trained network.

4. Identify cases where the extracted rule-based system surpasses the trained neural network and vice versa. This analysis helps in the process of integrating and combining the output decisions of the two subsystems.

The simplest and most popular inference engines examine rules in a predetermined sequential order, and make a decision based on the first fireable rule encountered. Alternatively, an inference engine can identify all fireable rules. The latter inference engine is considered more powerful then the former because it provides the system user with all possible output decisions and hence more choices can be examined.

For both types of inference engines, a predetermined order of the extracted rules plays an important role in determining which rule is going to be fired or the order in which all fireable rules are considered. Since the network does not directly provide information on rule ordering, a rule evaluation procedure that can rank order the extracted rules is crucial. We achieve this based on three performance measures:

1. **The soundness measure:** This measures how many times each rule is correctly fired, as one goes through the available data set. A rule is correctly fired if all its *premises* are satisfied and its *consequent* matches the target decision. The *soundness measure* of an extracted rule represents the ability of this rule to correctly interpret the output decisions of the trained network. Note that the soundness measure does not depend on the rule order.

2. **The completeness measure:** A completeness measure attached to a rule represents how many *distinct* times this rule is correctly fired (i.e., how many unique patterns are correctly identified/classified by this rule and not by any other extracted rule that is inspected by the inference engine before this rule). Thus this measure depends on the order in which the extracted rules are applied. For example, a rule with soundness measure is $> 0$ can have zero completeness measure if all the input patterns covered by this rule are also covered by at least one of the preceding rules.

3. **The false-alarm measure:** which measures how many times a rule is misfired over the available data set. When considered for application, a rule is *misfired* if all its premises are satisfied but its consequent does not match the target output. The value of this measure also depends on the order of rule application and the mechanism of the inference engine.

### 3.4.1 Rule ordering algorithm

Finding the optimal ordering of extracted rules is a combinatorial problem. So we have developed the following *"greedy"* algorithm to order any set of extracted rules, based on the three performance measures. The rule ordering algorithm first creates a list $L$ that contains all extracted rules. This list is divided into two lists, a head list ($L_h$) and a tail list ($L_t$), where $L_h$ is the list of all ordered rules and $L_t$ is the list of all remaining (unordered) rules[6]. Initially, $L_h$ is empty and $L_t$ includes all the extracted rules. A performance

[6] i.e., the ordering of rules in $L_t$ has no effect.

criteria is used to select one rule from $L_t$ to be moved to the end of $L_h$, and the process continues till $L_t$ is null.

The steps of the rule ordering algorithm are as follows:

---

1. *Initialize* $L_h = \{\ \}$, $L_t = \{$*all extracted rules*$\}$.
2. **WHILE** $L_t \neq \{\ \}$, **DO**
   (a) *Fire all rules in* $L_h$ *in order.*
   (b) *Compute the completeness and false-alarm measures for each rule in* $L_t$ *using the available data set.*
   (c) **IF** $\exists$ *a rule with zero false-alarm*
   **THEN** *this rule is moved from* $L_t$ *to the end of* $L_h$[7].
   **ELSE** *Among all rules in* $L_t$ *select the one with the highest (Completeness - False-alarm) measure; add this rule to the end of* $L_h$*, delete it from* $L_t$*.*
   (d) **IF** $\exists$ *any rule in* $L_t$ *with a zero completeness measure then remove this rule from* $L_t$*.*
3. **END DO**.

---

In this paper, all rules extracted by our approaches are ordered using the above algorithm. Also, the measures attached to all extracted rules assume that an inference engine that fires only one rule per input (namely, the first fireable rule) is used. Note that the three performance measures along with a rule's certainty factor (if applicable) can be used to form a composite measure (in an application dependent manner) which specifies the importance of the extracted rules.

An important issue that needs to be addressed here is: *"Should one discard rules with low soundness, completeness and/or high false-alarm measure(s)?"*. An example of such a rule is $R_{10}$ in Table 5. For small data sets we might still retain such rules at the bottom of the application ladder in the hope for better generalization, as they are part of the overall characteristics of the corresponding trained network. In cases where available data sets are representative, the answer depends on the application nature. For example, in medical applications where one is interested in high detection rates, rules with low soundness, completeness and/or high false-alarm measures may still be kept. In other applications, like Automatic Target Recognition (ATR), one may only retain rules with low false-alarm rate to reduce the chances of *"friendly fire"*.

[7] *If* $\exists$ *more than one rule with zero false-alarm* **THEN** *select the one with the highest completeness measure out of these rules to be moved from* $L_t$ *to the end of* $L_h$.

# 4 Implementation and Performance Evaluation

## 4.1 Data Sets

We applied all three rule extraction techniques to three problems:

1. An artificial rule-based system which has six rules relating four binary inputs and four binary outputs.

2. Iris database, a simple classification problem which contains 50 examples each of classes Iris Setosa, Iris Versicolor, and Iris Virginica [24]. These 150 instances were partitioned into a training set of size 89 and a testing set of size 61. Each input pattern has four continuous input features: $I_1$ = Sepal-length , $I_2$ = Sepal-width, $I_3$ = Petal-length, and $I_4$ =Petal-width.

3. Breast-Cancer data set which has nine inputs and two output classes [21, 24]. The input features are: $X_1$ = Clump Thickness, $X_2$ = Uniformity of Cell Size, $X_3$ = Uniformity of Cell Shape, $X_4$ = Marginal Adhesion, $X_5$ = Single Epithelial Cell Size, $X_6$ = Bare Nuclei, $X_7$ = Bland Chromatin, $X_8$ = Normal Nucleoli, and $X_9$ = Mitoses. All 9 inputs are continuous and in the range [1,10]. Each of the 683 available instances is labeled as Benign (444 instances) or Malignant. These instances are divided into a training set of size 341 and a test set of size 342.

Other popular data sets that have been used as benchmarks for rule extraction approaches are the Monk [43], Mushroom [32] and the DNA promoter [47] data sets. All three of these data sets inputs are symbolic/discrete by nature. Since we want to test more general problems that may include continuous valued variables, Iris and Breast-Cancer were preferred for our initial experiments.

## 4.2 Methodology

We now describe some of the important procedures followed to perform the experimental work presented in this paper.

1. **Training procedure:** in all experiments, an MLP network is trained using the backpropagation algorithm with momentum as well as a regularization term $P$ which adds $\frac{-2\lambda w_{jk} {w_0}^2}{{w_0}^2 + w_{jk}^2}$ to the weight update term in the backpropagation equation [9]. Cross-validation is used for the stopping criteria.

2. **Network architectures and data reduction:** for the iris problem, an MLP with 4 input, 6 hidden, and 3 output nodes is used for the three experiments but trained with different data sets each time, as described later. For the breast-cancer classification problem, we reduced the dimensionality of the input space from 9 to 6 inputs. This has been done by removing $X_4, X_5$, and $X_6$, as these inputs correspond to the lowest three eigenvalues of the covariance matrix of the original input space. The remaining 6 input features are then used for training and testing an MLP with 9 hidden and 2 output nodes.

3. **Network initialization:** for the artificial problem the six initial rules are used by the Node Link Algorithm [40], to initialize a network of 4 input, 6 hidden, and 4 output nodes. For both the iris and breast-cancer data sets, there is no prior knowledge so the corresponding networks are initialized randomly.

4. **Input representation:** inputs of the artificial problem are naturally binary, so there was no required mapping. Since the input features of both iris and breast-cancer problems are continuous, a binarized and a normalized version of these two data sets were computed and then used for training/testing the network architectures subsequently used for BIO-RE and Partial-RE respectively.

5. **Extraction techniques and networks labeling:** BIO-RE is used to extract rules from networks trained with binary/binarized input patterns. For iris and breast-cancer problems, these networks are labeled *Iris-Bin* and *Cancer-Bin* respectively. Partial-RE is used to extract rules from the networks trained with normalized input patterns (labeled *Iris-Norm* and *Cancer-Norm*). Full-RE uses the original data sets of both problems to train the corresponding networks. These two networks are labeled *Iris-Cont* and *Cancer-Cont*.

6. **Default class rule:** A comprehensive rule extraction approach is one that extracts rules to cover all input-output mapping cases. In some cases achieving such a goal is hard and it may be convenient to use default rules to cover the input-output mapping cases that cannot be covered by the extracted rule-base. Such default rules make the set of extracted rules complete but they do not provide any interpretation of why the default action was done other than *"none of the extracted rules could be fired"*. If a default rule is used, its output (consequent) can be chosen to minimize the false alarm rate and to maximize the correct classification rate. Note that the default rule may only fire when none of the extracted rules can be fired.

## 4.3 Experimental Results

### 4.3.1 An Artificial Binary Problem

This experiment is designed to test the soundness and completeness of the three rule extraction techniques. The initial rules are as follows:

*Rule #1:* $If\ A\ And\ B \xrightarrow{0.8} O_1$

*Rule #2:* $If\ B\ And\ C\ And\ D \xrightarrow{0.7} O_2$

*Rule #3:* $If\ Not\ C \xrightarrow{0.6} O_3$

*Rule #4:* $If\ Not\ A\ And\ D \xrightarrow{0.7} O_4$

*Rule #5:* $If\ B\ And\ D \xrightarrow{0.7} O_1$

*Rule #6:* $If\ D \xrightarrow{0.8} O_1$

where, $A$, $B$, $C$, and $D$ are binary inputs, and $O_i$s $(i = 1, ..., 4)$ are binary consequents. After using the Node

Link Algorithm [40] to map these six rules, into an initial network with 4 input, 6 hidden, and 4 output nodes, the following two experiments were performed.

**Experiment 1:** The objective of this experiment is to check whether the three approaches are able to extract the original rules from the mapped network. Therefore, the network was not trained before the extraction procedures were applied.

The results of applying the three rule extraction techniques to the generated (but not trained) network are as follows:

- BIO-RE extracts the same set of binary rules but without certainty factors.
- Partial-RE with $p = 2$ (i.e., maximum 2 conditions per rule) extracts all 5 original rules with two conditions or less. On increasing $p$ to 3, rule#2 was also extracted. The certainty factors attached to each output decision were approximately the same as the original rules.
- Full-RE extracts the same six original rules from the untrained network.

**Experiment 2:** Based on the original rules, $2^4$ binary patterns were generated. After training the previous network, we applied the three approaches to the final network architecture (i.e., the adapted one):

- Both BIO-RE and Full-RE extract the same six original rules.
- Partial-RE extracts all the six rules plus an extra one: Rule #7: $If\ B$ And $D \xrightarrow{0.74} O_2$.
  This rule was extracted when $p = 3$.

### 4.3.2 Iris Classification

Tables 1, 2 and 3 present the ordered rules extracted by BIO-RE, Partial-RE, and Full-RE techniques respectively from their corresponding networks trained on the Iris data set. They also present the corresponding measures for each extracted rule as generated by the rule evaluation procedure. Table 4 provides a summary of the performance of each rule extraction technique and compares it with the performance of the corresponding trained network. It shows that binarizing or scaling input patterns of the iris problem degrades the performance of the trained networks *("Iris-Bin" and "Iris-Norm")* as well as the corresponding rules extracted from these two networks. Also, it shows the remarkable performance of the rules extracted from network *"Iris-Cont"* by Full-RE.

Note that:

(i) The numeric values compared with input features $I_i$s in the rules extracted by both BIO-RE and Partial-RE represent the mean ($\mu_i$s) of these input feature (see the rule bodies in Table 1 and Table 2). This coarse thresholding is largely responsible for the (relatively) poor performance of the two networks and subsequently of the extracted rules.

(ii) In Table 3, a numeric value that is compared to an input feature $I_i$ in the rule body represents one of

Table 1: Rules extracted from network *"Iris-Bin"* by BIO-RE technique.

| Rule No. | Rule Body | Iris Class | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|
| 1 | If $I_2 \leq 3.0$ $I_3 \geq 3.7$ and $I_4 \leq 1.2$ | Versicolor | 10/50 | 10/50 | 0/150 |
| 2 | If $I_1 \leq 5.8$ and $I_3 \leq 3.7$ and $I_4 \leq 1.2$ | Setosa | 50/50 | 50/50 | 5/150 |
| 3 | If $I_1 \geq 5.8$ and $I_3 \geq 3.7$ and $I_4 \geq 1.2$ | Virginica | 47/50 | 47/50 | 24/150 |
| 4 | If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_4 \geq 1.2$ | Versicolor | 15/50 | 11/50 | 3/150 |
| Overall Performance | | | | 118/150 | 32/150 |
| % | | | | 78.67% | 21.33% |

Table 2: Rules extracted from network *"Iris-Norm"* by Partial-RE technique.

| Rule No. | Rule Body | Iris Class | Certainty Factor | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|---|
| 1 | If $I_2 \geq 3.0$ and $I_3 \leq 3.7$ | Setosa | 0.98 | 48/50 | 48/50 | 0/150 |
| 2 | If $I_1 \geq 5.8$ and $I_3 \geq 3.7$ and $I_4 \geq 1.2$ | Virginica | 0.99 | 47/50 | 47/50 | 27/150 |
| 3 | If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_3 \geq 3.7$ | Versicolor | 0.74 | 18/50 | 16/50 | 3/150 |
| 4 | If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_4 \geq 1.2$ | Versicolor | 0.72 | 15/50 | 1/50 | 0/150 |
| 5 | Default Class | Versicolor | 1.0 | 6/50 | 6/50 | 2/150 |
| Overall Performance | | | | | 118/150 | 32/150 |
| % | | | | | 78.67% | 21.33% |

Table 3: Rules Extracted from network *"Iris-Cont"* by Full-RE technique.

| Rule No. | Rule Body | Iris Class | Certainty Factor(cf) | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|---|
| 1 | If $I_3 \leq 2.1$ | Setosa | 0.99 | 50/50 | 50/50 | 0/150 |
| 2 | If $I_3 \leq 5.1$ and $I_4 \leq 1.7$ | Versicolor | 0.97 | 49/50 | 49/50 | 3/150 |
| 3 | If $I_3 \geq 4.8$ | Virginica | 0.98 | 47/50 | 47/50 | 1/150 |
| Overall Performance | | | | | 146/150 | 4/150 |
| % | | | | | 97.33% | 2.67% |

Table 4: Performance comparison between the sets of extracted rules and their corresponding trained networks for the iris problem.

| | | Neural Network | | Extracted Rules | |
|---|---|---|---|---|---|
| | | ratio | % match | ratio | % match |
| Binarized Network (Iris-Bin) | Training | 66/89 | 74.16 | 67/89 | 75.28 |
| | Testing | 43/61 | 70.49 | 51/61 | 83.61 |
| | Overall | 109/150 | 72.67 | 118/150 | 78.67 |
| Normalized Network (Iris-Norm) | Training | 84/89 | 94.38 | 69/89 | 77.53 |
| | Testing | 56/61 | 91.80 | 49/61 | 80.33 |
| | Overall | 140/150 | 93.33 | 118/150 | 78.67 |
| Continuous Network (Iris-Cont) | Training | 87/89 | 97.75 | 83/86 | 96.63 |
| | Testing | 59/61 | 96.72 | 60/61 | 98.36 |
| | Overall | 146/150 | 97.33 | 146/150 | 97.33 |

the critical discretization boundaries of that feature which was selected by Full-RE.
(iii) For rules examined later (e.g., rule 4 in Table 2), completeness may be much less than soundness, because some instances where these rules would fire correctly have already been covered by other preceding rules.
(iv) Full-RE leads to three simple rules that classify the iris data set very well.

#### 4.3.3 Breast-Cancer Classification

For the breast-cancer classification problem, Tables 5, 6 and 7 present three sets of ordered rules extracted by the three rule extraction techniques, along with the corresponding performance measures. Table 8 provides an overall comparison between the extracted rules and their corresponding trained networks. It shows that the three techniques were successfully used with approximately the same performance regardless of the nature of the training and testing data sets used for each network. Also, it shows that binarizing and scaling breast cancer data set did substantially degrade performance. Since the original input features of the breast cancer problem have the same range (1-10) and are not very skewed, by binarizing and/or scaling them we did not change their nature much.

### 4.4 Discussion

The implementation results of Section 4.3 indicate that:

1. All rules extracted by the three techniques are sound.

2. Partial-RE is sound but not complete. Its completeness depends on the chosen degree of comprehensibility ($p$).

3. Rules extracted by Full-RE are much more comprehensible than those extracted by the BIO-RE and Partial-RE. This is likely due to the finer gradation of the input space allowed by this algorithm.

Table 5: Rules Extracted from network *"Cancer-Bin"* by BIO-RE technique.

| Rule No. | Rule Body | B-Cancer Class | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|
| 1 | If $X_3 \leq 3.0$ and $X_7 \leq 3.3$ and $X_8 \leq 2.7$ and $X_9 \leq 1.5$ | Benign | 391/444 | 391/444 | 2/683 |
| 2 | If $X_1 \leq 4.1$ and $X_3 \leq 3.0$ and $X_7 \leq 3.3$ and $X_9 \leq 1.5$ | Benign | 317/444 | 8/444 | 0/683 |
| 3 | If $X_1 \leq 4.1$ and $X_3 \leq 3.0$ and $X_8 \leq 2.7$ and $X_9 \leq 1.5$ | Benign | 316/444 | 7/444 | 0/683 |
| 4 | If $X_1 \leq 4.1$ and $X_3 \leq 3.0$ and $X_7 \leq 3.3$ and $X_8 \leq 2.7$ | Benign | 316/444 | 7/444 | 0/683 |
| 5 | If $X_1 \leq 4.1$ and $X_7 \leq 3.3$ and $X_8 \leq 2.7$ and $X_9 \leq 1.5$ | Benign | 314/444 | 5/444 | 0/683 |
| 6 | If $X_1 \geq 4.1$ and $X_3 \geq 3.0$ | Malignant | 200/239 | 199/239 | 15/683 |
| 7 | If $X_3 \geq 3.0$ and $X_7 \geq 3.3$ | Malignant | 187/239 | 27/239 | 2/683 |
| 8 | If $X_3 \geq 3.0$ and $X_8 \geq 2.7$ | Malignant | 187/239 | 3/239 | 0/683 |
| 9 | If $X_1 \geq 4.1$ and $X_7 \geq 3.3$ | Malignant | 167/239 | 7/239 | 1/683 |
| 10 | If $X_1 \geq 4.1$ and $X_9 \geq 1.5$ | Malignant | 100 | 1 | 3 |
| 11 | Default Class | Benign | 5/444 | 5/444 | 0/239 |
| Total For Benign Rules | | | | 423/444 | 2/683 |
| % | | | | 95.27% | 0.29% |
| Total For Malignant Rules | | | | 237/239 | 21/683 |
| % | | | | 99.16% | 3.07% |
| Overall Performance | | | | 660/683 | 23/683 |
| % | | | | 96.63% | 3.37% |

Table 6: Rules extracted from network *"Cancer-Norm"* by Partial-RE technique.

| Rule No. | Rule Body | B-Cancer Class | Certainty Factor(cf) | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|---|
| 1 | If $X_2 \leq 3.0$ and $X_3 \leq 3.0$ and $X_7 \leq 3.3$ | Benign | 0.99 | 412/444 | 412/444 | 6/683 |
| 2 | If $X_1 \leq 4.1$ and $X_3 \leq 3.0$ and $X_7 \leq 3.3$ | Benign | 0.99 | 324/444 | 1/444 | 0/683 |
| 3 | If $X_2 \geq 3.0$ and $X_3 \geq 3.0$ | Malignant | 0.99 | 222/239 | 219/239 | 15/683 |
| 4 | If $X_1 \geq 4.1$ and $X_7 \leq 3.3$ and $X_8 \geq 2.7$ | Malignant | 0.99 | 137/239 | 8/239 | 0/683 |
| 5 | If $X_1 \leq 4.1$ and $X_2 \leq 3.0$ and $X_7 \leq 3.3$ | Benign | 0.84 | 327/444 | 4/444 | 0/683 |
| 6 | If $X_1 \geq 4.1$ and $X_2 \geq 3.0$ | Malignant | 0.99 | 198/239 | 2/239 | 0/683 |
| 7 | If $X_1 \leq 4.1$ and $X_2 \leq 3.0$ and $X_3 \leq 3.0$ | Benign | 0.84 | 333/444 | 9/444 | 1/683 |
| 8 | If $X_1 \geq 4.1$ and $X_3 \geq 3.0$ | Malignant | 0.99 | 200/239 | 3/239 | 2/683 |
| 9 | If $X_2 \leq 3.0$ and $X_3 \leq 3.0$ and $X_8 \leq 2.7$ | Benign | 0.99 | 409/444 | 1/444 | 0/683 |
| Total For Benign Rules | | | | | 427/444 | 7/683 |
| % | | | | | 96.17% | 1.02% |
| Total For Malignant Rules | | | | | 232/239 | 17/683 |
| % | | | | | 97.07% | 2.49% |
| Overall Performance | | | | | 659/683 | 24/683 |
| % | | | | | 96.49% | 3.51% |

Table 7: Rules extracted from network *"Cancer-Cont"* by Full-RE technique.

| Rule No. | Rule Body | B-Cancer Class | Certainty Factor | Soundness Measure | Completeness Measure | False-Alarm Measure |
|---|---|---|---|---|---|---|
| 1 | If $X_1 < 8$ and $X_3 < 3$ | Benign | 0.96 | 394/444 | 394/444 | 5/683 |
| 2 | If $X_2 \geq 2$ and $X_7 \geq 3$ | Malignant | 0.83 | 227/239 | 223/239 | 18/683 |
| 3 | If $X_1 < 8$ and $X_7 < 3$ | Benign | 0.75 | 300/444 | 27/444 | 1/683 |
| 4 | If $X_1 \geq 8$ | Malignant | 0.89 | 123/239 | 9/239 | 1/683 |
| 5 | If $X_1 < 8$ and $X_2 < 2$ | Benign | 0.79 | 369/444 | 4/444 | 1/683 |
| Total For Benign Rules | | | | | 425/444 | 7/683 |
| % | | | | | 95.72% | 1.02% |
| Total For Malignant Rules | | | | | 232/239 | 19/683 |
| % | | | | | 97.07% | 2.78% |
| Overall Performance | | | | | 657/683 | 26/683 |
| % | | | | | 96.19% | 3.81% |

Table 8: Performance comparison between the sets of extracted rules and their corresponding trained networks for the breast-cancer problem.

| | | Neural Network | | Extracted Rules | |
|---|---|---|---|---|---|
| | | ratio | % match | ratio | % match |
| Binarized Network (Cancer-Bin) | Training | 333/341 | 97.65 | 331/341 | 97.07 |
| | Testing | 317/342 | 92.69 | 329/342 | 96.20 |
| | Overall | 650/683 | 95.17 | 660/683 | 96.63 |
| Normalized Network (Cancer-Norm) | Training | 329/341 | 96.48 | 331/341 | 97.07 |
| | Testing | 325/342 | 95.03 | 328/342 | 95.91 |
| | Overall | 654/683 | 95.75 | 659/683 | 96.49 |
| Continuous Network (Cancer-Cont) | Training | 334/341 | 97.95 | 330/341 | 96.77 |
| | Testing | 331/342 | 96.78 | 327/342 | 95.61 |
| | Overall | 665/683 | 97.36 | 657/683 | 96.19 |

4. Binarizing or normalizing continuous features may degrade the accuracy of the extracted rules as well as the generalization capability of the corresponding trained neural network. See the first six rows of Table 4.

5. Full-RE was tested several times on different networks initialized randomly each time and trained with different sets of training patterns. Each time the set of extracted rules are similar except for the values of the certainty factors. This indicates that Full-RE is more accurate and can extract rules based on more combinations of input features, not just the most effective features, see Table 3 and Table 7.

6. Although BIO-RE and Partial-RE were used to extract rules from networks trained with binarized and normalized input features, they were still able to extract “certain” rules that may be adequate in some application examples. See Table 5 and Table 6.

7. Although some of the extracted rules have a low firing rate on the available data set, they were extracted to represent the generalization capability of the trained network on unseen data. Also, they were extracted to cover all training and testing data sets and hence increase the completeness of the extracted set of rules. Examples of such rules are: $R_1$ and $R_4$ of Table 1, $R_4$ of Table 2, and $R_2$-$R_5$ of Table 5.

# 5 Comparative Performance Evaluation

Since both iris and breast cancer problems have continuous input features, Full-RE is naturally suited for them. There is no need to prune the trained network since Full-RE is capable of extracting rules from MLPs of any size. In this section, we compare the performance of the extracted rules from the iris and the breast-cancer databases with the rules extracted by both NeuroRule[35] and C4.5rules algorithms [36], since these methods have previously been applied on the same databases. Moreover, they both extract comprehensive rules with relatively high correct classification rate, as reported in [35]. For iris problem, we also compare the set of rules extracted by Full-RE with the corresponding set of rules extracted by KT algorithm [8].

Before analyzing the extracted rules, we summarize the computational complexity of NeuroRule and C4.5rules:

- **NeuroRule:** as a starting point of the NeuroRule algorithm, a large number (100) of fully connected MLPs are generated. Before training any of these 100 networks, each input feature is discretized into one of $n$ values, which are then converted into binary ones using a thermometer coding. Although this discretization step helps simplify the last step of the rule extraction process, it has the major drawback of greatly increasing the number of input nodes and hence the complexity of the required network architecture. For example, the network architecture generated by NeuroRule has 39 input nodes for the iris problem and 91 input nodes for the breast cancer problem, while the corresponding networks used by Full-RE have 4 and 6 input nodes respectively. This increase in input size makes the network more difficult to train, both in terms of training time and performance.

NeuroRule employs a pruning procedure after the training phase to reduce the complexity of the generated network architectures. The pruning process continues until network performance drops to 95% of original performance. This process is applied to the 100 MLPs. The rule extraction procedure starts by choosing the *best one* out of the 100 pruned networks (the one with the highest performance). NeuroRule extracts rules by clustering the remaining hidden nodes activation values and then checking which input combination can make each hidden (and later output) node active.

The power of NeuroRule lies in its pruning and clustering techniques. In its pruning phase, NeuroRule removes input nodes. For example, the best pruned architecture for the iris problem is a network of 4 input, 2 hidden, and 3 output nodes. For breast-cancer problem the best pruned network has 6 input, 1 hidden and 2 output nodes. The pruning and clustering processes lead to substantial overhead, but since the resulting network architectures from the pruning step are very small, the rule extraction process becomes easy.

- **C4.5rules:** C4.5rules was used by the authors of NeuroRule to extract rules from the iris and breast-cancer databases for comparison reasons. Like ID3, C4.5rules [28] generates decision tree rules based on the available input samples. Therefore, the complexity is moderate, but the performance of the rules generated by C4.5rules is highly affected by the noise level in the available data samples [8].

## 5.1 Comparison using iris data set

The rules extracted by the Full-RE techniques for the iris problem were given in Table 3. The rules extracted by NeuroRule for the same problem are:

*Rule 1: If* $I_3 \leq 1.9$*, then Iris Setosa*

*Rule 2: If* $I_3 \leq 4.9$ *and* $I_4 \leq 1.6$*, then Iris Versicolor*

*Rule 3: Default Rule (Iris Virginica)*

The corresponding rules extracted by C4.5rules are:

*Rule 1: If* $I_3 \leq 1.9$*, then Iris Setosa*

*Rule 2: If* $I_3 \geq 1.9$ *and* $I_4 \leq 1.6$*, then Iris Versicolor*

*Rule 3: If* $I_4 \geq 1.6$*, then Iris Virginica*

*Rule 4: Default Rule (Iris Setosa)*

The corresponding rules extracted by KT approach are:

*Rule 1: If* $I_3 \leq 2.7$*, then Iris Setosa*

*Rule 2: If* $I_3 \leq 5.0$ *and* $I_3 > 2.7$ *and* $I_4 \leq 1.6$ *and* $I_4 > 0.7$ *, then Iris Versicolor*

*Rule 3: If* $I_3 > 5.0$*, then Iris Virginica*

*Rule 4: If* $I_4 > 1.6$*, then Iris Virginica*

Table 9: Correct classification rate (%) of the rule sets extracted by different techniques.

| | | Full-RE | NeuroRule | C4.5rules | KT |
|---|---|---|---|---|---|
| Iris | with default rule | 97.33 | 98.00 | 96.00 | 97.33 |
| | without default rule | 97.33 | 64.67 | 96.00 | 97.00 |
| Breast Cancer | with default rule | 96.19 | 97.21 | 97.21 | N/A |
| | without default rule | 96.19 | 63.10 | 94.72 | N/A |

*Rule 5: If* $I_2 > 3.1$ *and* $I_3 > 2.7$ *and* $I_3 \leq 5.0$*, then Iris Versicolor*

From the methodologies and results, we note that:

1. **Completeness:** Both Full-RE and KT extracted complete sets of rules that cover all cases, and so no default rule was required. However, a default rule is essential for both NeuroRule (due to its pruning step) and for C4.5.

2. **Comprehensibility:**

   (a) **Number of rules:** Both Full-RE and NeuroRule extract 3 rules while KT extracts 5 rules and C4.5 extracts 4 rules.

   (b) **Number of premises per rule:** Except KT, the maximum number of conditions per rule for all other techniques is 2. KT extracted rules with a maximum of 4 conditions per rule.

3. **Performance:** Since iris is a simple classification problem, all techniques performed well. In fact, all of them were able to show that the Setosa class is linearly separable from the other two classes. Moreover, rule extracted by all of them showed that *Petal-length* is the most dominant input feature (see row 1 and 2 in Table 9).

4. **Certainty factors:** Rules extracted by the Full-RE provide a certainty factor attached with each extracted rule, unlike the other approaches.

## 5.2 Comparison using breast cancer data set

For the breast cancer database, the rules extracted by the Full-RE from a simple MLP architecture (6 input, 6 hidden, and 2 output nodes) are presented in Table 7. The rules extracted by NeuroRule from the best among the pruned 100 MLP network architectures (6 inputs, 1 hidden, and 2 output nodes) are [36, 34]:

*Rule 1: If* $X_1 < 7.0$ *and* $X_2 < 8.0$ *and* $X_3 < 3.0$ *and* $X_8 < 9.0$*, then Benign*

*Rule 2: If* $X_1 < 7.0$ *and* $X_2 < 8.0$ *and* $X_3 < 3.0$ *and* $X_6 < 9.0$*, then Benign*

*Rule 3: If* $X_2 < 8.0$ *and* $X_3 < 3.0$ *and* $X_6 < 3.0$ *and* $X_8 < 9.0$*, then Benign*

*Rule 4: Default Rule (Malignant)*

The corresponding rules extracted by C4.5rules are [36]:

*Rule 1: If* $X_1 < 7.0$ *and* $X_2 < 8.0$ *and* $X_3 < 3.0$*, then Benign*

*Rule 2: If* $X_1 < 7.0$ *and* $X_2 < 2.0$*, then Benign*

*Rule 3: If* $X_2 \geq 5.0$*, then Malignant*

*Rule 4: If* $X_6 \geq 9.0$*, then Malignant*

*Rule 5: If* $X_1 \geq 7.0$*, then Malignant*

*Rule 6: If* $X_4 \geq 4.0$*, then Malignant*

*Rule 7: Default Rule (Benign)*

Comparing these three sets of extracted rules, we observed:

1. **Completeness:** NeuroRule did not extract any rule for class Malignant. Both NeuroRule and DT (C4.5) have a default rule while rules extracted by Full-RE cover all cases and hence there was no need for a default rule. Default rules are undesirable because they cannot provide a symbolic interpretation of the decision other than *"because none of the above occurred"*.

2. **Comprehensibility:**

   (a) **Number of rules:** Number of rules extracted by Full-RE is 5 and by DT (C4.5) is 7. NeuroRule extracted only 4 rules, as it used a default rule to cover all cases of class Malignant, which were applied to a highly pruned network with only one hidden node.

   (b) **Number of premises per rule:** For Full-RE, the maximum number of conditions per extracted rule is 2. All rules extracted by NeuroRule have 4 conditions, while those extracted by DT have a maximum of 3 conditions per rule. Thus the rules extracted by Full-RE are more comprehensible than those extracted by the other two techniques.

3. **Performance:** The performance of the rules extracted by all the three techniques are very high and they all achieve very low misclassification rate (see row 3 of Table 9). When default rules are removed, the performance of NeuroRule drops dramatically (see row 4 of Table 9). Authors of NeuroRule reported that by choosing different trained network architectures they extracted different rules. In one case, only two rules were extracted, one of which is a default rule. In another experiment, NeuroRule extracts only 3 rules (one of which is also a default rule). In both experiments, the achieved completeness measure is approximately 95%. However, we did not observe any effect on the rules extracted by the Full-RE due to changing the initialization of the "Cancer-Cont" network or when we used different input samples for training and testing. This indicates that Full-RE rules are quite stable so long the network is trained reasonably well.

4. **Certainty factors:** Rules extracted by the Full-RE provide a certainty factor attached with each extracted rule while NeuroRule and C4.5rules do not. Note that KT was not used to extract rules from the breast cancer problem.

Table 10: A qualitative comparison of different rule extraction techniques.

| | BIO-RE | Partial-RE | Full-RE | NeuroRule | KT or Subset | MofN |
|---|---|---|---|---|---|---|
| Provides CF | No | Yes | **Yes** | No | No | No |
| May need a default rule | Yes | Yes | **No** | Yes | Yes | Yes |
| Works for 1.Binary inputs | Yes | Yes | **Yes** | Yes | Yes | Yes |
| 2.Normalized inputs | No | Yes | **Yes** | No | No | No |
| 3.Continuous inputs | No | No | **Yes** | No | No | No |
| Complexity | Very Low | Low | **Med** | Very High | Med | High |
| Additional overheads | No | No | **No** | Yes | No | Yes |

Table 9 compares the classification rates obtained using the rules extracted by the four technique (Full-RE, NeuroRule, C4.5rules, and KT), for the iris and the breast-cancer databases, while Table 10 presents a qualitative comparison between our three techniques and some other notable rule extraction techniques from trained neural networks (NeuroRule, KT, Subset, MofN). Note that C4.5rules was not included in the comparative study presented by Table 10 because it extracts rules from decision trees and not from trained networks like the other approaches.

# 6 Conclusions

In this paper, we introduced three new rule extraction techniques. The suitability of each approach depends on the network type and architecture, complexity, the application nature, inputs, and the required transparency level. All three methods are able to extract meaningful rules for the well known Iris database and Wisconsin breast cancer diagnosis database, where no pre-existing rules are available. The extracted rules compare favorably with other reported implementation results. The proposed techniques are less complex, and the rules extracted by them are efficient, comprehensible and powerful.

The ordering of extracted rules has to be determined while designing the inference engine. The neural network does not provide any (direct) information on this issue, and other KBNN researchers have not reported on this aspect. We developed a simple greedy rule evaluation procedure and an algorithm that can order rules extracted by any rule extraction algorithm, with a goal of maximizing performance and minimizing error rates of the extracted rules over available data. We also presented a qualitative comparison of some key issues involved in the process of extracting rules from trained networks by different approaches.

It is important to mention that obtaining all possible combinations of rules is NP-hard and a feasible alternative is often to extract key rules that cover most of the embedded knowledge. More progress is needed in determining when an adequate set of rules has been extracted. Another important issue that needs to be investigated is how the outputs of both the rule extraction and the trained ANN modules can be integrated to provide more robust decisions, and how the extracted rules can be used for knowledge refinement and truth maintenance of domain knowledge. The stability or robustness of rule extraction methods in presence of noise or outliers in the data should also be explored further.

**Acknowledgements**: The second author benefited from the superb NIPS'96 workshop on Rule Extraction organized by Robert Andrews and Joachim Diederich, and the comments of Lee Giles, Paul Munro, Mark Craven and other workshop participants. Discussions on theory refinement with Ray Mooney and Xiaowei Wang were very helpful, and so were the inputs provided by the anonymous referees.

## References

[1] R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. Technical Report 95-4, Neurocomputing Research Center, Queensland University of Technology, Queensland, Australia, 1995.

[2] R. Andrews and S. Geva. Inserting and extracting knowledge from constrained error backpropagation networks. In *Proceedings of Sixth Australian Conference on Neural Networks*, Sydney NSW, 1995.

[3] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithm for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[4] T. Caelli and W. Bischof. The role of machine learning in building image interpretation systems. *Journal of Artificial Intelligence and Pattern Recognition*, 11:143.

[5] R. Challo, R.A. McLauchlan, D.A. Clark, and S.I. Omar. A fuzzy neural hybrid system. In *IEEE International Conference on Neural Networks*, volume III, pages 1654–1657, Orlando, FL., 1994.

[6] M.W. Craven and J.W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Machine Learning: Proceedings of the Eleventh international Conference*, pages 37–45, 1994.

[7] L.M. Fu. Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):173–182, 1993.

[8] L.M. Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, Inc., 1994.

[9] J. Ghosh and K. Tumer. Structural adaptation and generalization in supervised feed-forward networks. *Journal of Artificial Neural Networks*, 1(4):431–458, 1994.

[10] C.L. Giles et al. Extracting and learning an unknown grammar with recurrent neural networks. In S.J. Hanson J.E. Moody and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems-4*. Morgan Kaufmann, San Mateo, CA, 1992.

[11] C.W. Glover, M. Silliman, M. Walker, and P. Spelt. Hybrid neural network and rule-based pattern recognition system capable of self-modification. In *Proceedings of SPIE, Application of Artificial Intelligence VIII*, pages 290–300, 1990.

[12] J.A. Hendler. Marker-passing over microfeatures: Towards a hybrid symbolic/connectionist model. *Cognitive Science*, 13:79–106, 1989.

[13] S. Horikawa, T. Furuhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with back-propagation algorithm. *IEEE Transactions on Neural Networks*, 3(5):801–806, September 1992.

[14] P. Howes and N. Crook. Rule extraction from neural networks. In R. Andrews and J. Diederich, editors, *Rules and Networks: Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop*, pages 60–67. Queensland University of Technology, Neurocomputing Research Center QUT NRC, April 1996.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:78–88, 1991.

[16] R. Kerber. ChiMerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence AAAI*, pages 123–128, July 1992.

[17] R.H. Klinkenberg and D. C. St.Clair. Rule set quality measures for inductive learning algorithms. In *Intelligent Engineering Systems Through Artificial Neural Networks ANNIE*, volume 6, pages 161–168. ASME Press, November 1996.

[18] C.T. Lin and C.S.G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, 40(12):1320–1326, December 1991.

[19] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, pages 388–391, November 1995.

[20] J.J. Mahoney and R.J. Mooney. Combining connectionist and symbolic learning to refine certainty factor rule bases. *Connection Science*, 5(3-4):339–364, 1993.

[21] O.L. Mangasarian and H.W. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[22] C. McMillan, M.C. Mozer, and P. Smolensky. The connectionist scientist game: Rule extraction and refinement in a neural network. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 1991.

[23] W. Mendenhall. *Introduction to Probability and Statistics, Fifth Edition.* Wadsworth Publishing Company, Inc., 1979.

[24] P.M. Murphy and D.W. Aha. UCI repository of machine learning database. Technical report, University of California, Department of Computer Science, 1992.

[25] C.W. Omlin and C.L. Giles. Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, 9(1):41–52, 1996.

[26] D.W. Optiz and J.W. Shavlik. Heuristically expanding knowledge-based neural network. In *Proceedings of Thirteenth International Joint Conference on Artificial Intelligence*, pages 512–517, 1993.

[27] D. Ourston and R.J. Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 815–820. AAAI Press, 1990.

[28] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, San Mateo, California, 1992.

[29] S. Ramachandran. *Theory Refinement of Bayesian Netowrks with Hidden Variables.* PhD thesis, Dept. of Computer Sc., Univ. of Texas at Austin, Dec, 1997.

[30] R. Ruddel and A. Sangiovanni-Vincentelli. Espresso-MV: Algorithms for multiple-Valued logic minimization. In *Proceedings of Cust. Int. Circ. Conf.* Portland, May 1985.

[31] K. Saito and R. Nakano. Medical diagnostic expert system based on PDP model. In *Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 255–262, 1988.

[32] J.S. Schlimmer. *Concept Acquisition Through Representational Adjustment.* PhD thesis, University of California, Irvine Department of Information and Science, May 1996.

[33] S. Sestito and T. Dillon. Automated knowledge acquisition of rules with continuously valued attributes. In *Proceedings of Twelfth International Conference on Expert Systems and their Applications (AVIGNON)*, pages 645–656, May 1995.

[34] R. Setiono. Extracting rules from pruned neural networks for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 8(1):37–51, February 1996.

[35] R. Setiono and H. Liu. Understanding neural networks via rule extraction. In *Proceedings of Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 480–485, 1995.

[36] R. Setiono and H. Liu. Symbolic representation of neural networks. *IEEE Computer*, pages 71–77, March 1996.

[37] N.E. Sharkey and A.J.C. Sharkey. Understanding catastrophic interference in neural networks. Technical Report CS-94-4, Dept. of Computer Sc., Sheffield, UK, 1994.

[38] I. Taha. *A hybrid intelligent architecture for revising domain knowledge.* PhD thesis.

[39] I. Taha and J. Ghosh. A hybrid intelligent architecture for refining input characterization and domain knowledge. In *Proceedings of World Congress on Neural Networks, (WCNN)*, volume II, pages 284–287, July 1995.

[40] I. Taha and J. Ghosh. Hybrid intelligent architecture and its application to water reservoir control. *International Journal of Smart Engineering Systems*, 1:59–75, 1 1997.

[41] H. Takagi and I. Hayashi. NN-driven fuzzy reasoning. In J.C. Bezdek and S.K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 496–512. IEEE Press, 1992.

[42] E. Tazaki and N. Inoue. A generation methods for fuzzy rules using neural networks with planar lattice architecture. In *IEEE International Conference on Neural Networks*, volume III, pages 1743–1748, Orlando, FL., 1994.

[43] S.B. Thrun, J. Bala, E. Bloedorn, B. Cheng I. Bratko, S. Dzeroski k. De-Jong, S. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, K. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The monk's problem: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1990.

[44] A.B. Tickle, M. Orlowski, and J. Diederich. DEDEC: A methodology for extracting rules from trained artificial neural networks. In R. Andrews and J. Diederich, editors, *Rules and Networks: Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop*, pages 90–102. Queensland University of Technology, Neurocomputing Research Center QUT NRC, April 1996.

[45] G.G. Towell and J.W. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, 1993.

[46] G.G. Towell and J.W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1-2):119–165, 1994.

[47] G.G. Towell, J.W. Shavlik, and M.O. Noordwier. Refinement of approximate domain theories by knowledge-based artificial neural network. In *Proceedings of Eighth National Conference on Artificial Intelligence*, pages 861–866, 1990.

**Ismail A. Taha** was born on July 10, 1962. He obtained BS and MS degrees from The Military Technical College (MTC) Cairo, Egypt, in Electrical Engineering and Computer Science, both with top honors and MTC fellowship. He was a lecturer at MTC, Cairo till 1993 when he joined the graduate program at the University of Texas at Austin with an Egyptian Government Graduate Fellowship. After completing his Ph.D in May 1997 and a stint at Trilogy Development Group, Austin, Dr. Taha has rejoined the faculty at MTC as Assoc. Professor and Graduate Coordinator. Dr. Taha has several refereed publications in the areas of intelligent systems and databases, including one that received the best application paper award (1st runners up) at ANNIE'95.

**Joydeep Ghosh** was educated at IIT Kanpur (B. Tech '83) and The University of Southern California (MS, Ph.D, '88). He is currently a Full Professor with the Department of Electrical and Computer Engineering at UT Austin, where he holds the Endowed Engineering Foundation Fellowship. He directs the Laboratory for Artificial Neural Systems (LANS), where his research group is studying the theory and applications of adaptive pattern recognition and multi-learner systems. Dr. Ghosh has published more than 125 refereed papers and edited 6 books. He received the 1992 Darlington Award for the Best Paper in the areas of CAS/CAD, besides "best conference paper" citations for five neural network papers. Dr. Ghosh served as the general chairman for the SPIE/SPSE Conference on Image Processing Architectures, Santa Clara, Feb. 1990 as Conference Co-Chair of Artificial Neural Networks in Engineering (ANNIE)'93 through ANNIE'96, and in the program committee of several conferences on neural networks and parallel processing. He is an associate editor of Pattern Recognition, IEEE Trans. Neural Networks,Neural Computing Surveys and Int'l Jl. of Smart Engineering Design, and the Letters Editor for IEEE Trans. Neural Networks. He was a plenary speaker for ANNIE'97.