# CHAPTER 1

# A NEURO-SYMBOLIC HYBRID INTELLIGENT ARCHITECTURE WITH APPLICATIONS

**J. Ghosh**
Department of Electrical and Computer Engineering
University of Texas
Austin, TX 78712-1084
U.S.A.
ghosh@ece.utexas.edu

**I. Taha**
Military Technical College
Cairo, Egypt
ismail.taha@mailexcite.com

Hybrid Intelligent Architectures synergistically combine the strengths of diverse computational intelligence paradigms and avail of both domain knowledge and training data to solve difficult learning tasks. In particular, several researchers have studied some aspects of combining symbolic and neural/connectionist approaches, such as initializing a network based on existing rules, or extracting rules from trained neural networks. In this chapter, we present a *complete system* that embeds initial domain knowledge and/or statistical information into a custom neural network, refines this network using training data, and finally extracts back refined knowledge in the form of a refined rule base with an associated inference engine. Two successful applications of this hybrid architecture are described.

## 1 Introduction

The synergistic use of multiple models to difficult problems has been advocated in a variety of disciplines. Such approaches can yield systems that not only perform better, but are also more comprehensive and *robust*. A strong motivation for such systems was voiced by Kanal in his classic 1974 paper [17], prompting work on combining linguistic and statistical models, and heuristic search with statistical pattern recognition. In nonlinear control, multiple model methods, such as gain-schedule control, have a long tradition (see http://www.itk.ntnu.no/ansatte/Johansen_Tor.Arne/mmamc/address.html for a detailed list of researchers).

Sentiments on the importance of multiple approaches have also been voiced in the AI community, for example, by Minsky [27]:

> " To solve really hard problems, we'll have to use several different representations.... It is time to stop arguing over which type of pattern-classification technique is best.... Instead we should work at a higher level of organization and discover how to build managerial systems to exploit the different virtues and evade the different limitations of each of these ways of comparing things."

Indeed, there are several examples of successful multi-model approaches in the "learning" community – from the theory of neural network ensembles and modular networks [31] to multistrategy learning [26]. Hybridization in a broader sense is seen in efforts to combine two or more of neural network, Bayesian, GA, fuzzy logic and knowledge-based systems [1], [4], [25], [35]. The goal is again to incorporate diverse sources and forms of information and to exploit the somewhat complementary nature of different methodologies.

The main form of hybridization of interest in this chapter involves the integration of symbolic and connectionist approaches [6], [8], [13], [15], [18], [24], [35], [41]. Such combinations have attracted widespread interest for several reasons that are founded on the complementary strengths and weaknesses of these two approaches. For example, in many application domains, it is hard to acquire the complete domain knowledge and represent it in a rule based format. Moreover, the acquired knowledge may be uncertain or inconsistent [16], [30]. Expert systems can also suffer from the brittleness of rules and lead to problems when the domain theory is noisy [33]. Data driven connectionist models, on the other hand, can be trained in a supervised or unsupervised fashion to perform reasonably well even in domains with noisy training data. However, they cannot as readily incorporate existing domain knowledge or provide a symbolic explanation of results. Finally we note that in many real life situations, there is some amount of existing domain knowledge (which a purely model free neural network cannot exploit) as well as domain data that may be acquired over time. Being able to use both knowledge and data is paramount, specially in "non-stationary" scenarios that demand continuous model tuning or refinement, thus further motivating hybrid methods.

In this chapter, we present a comprehensive **Hybrid Intelligent Architecture (HIA)** that augments a knowledge base system with connectionist and statistical models to help the former refine its domain knowledge and improve its performance and robustness. Figure 1 shows the key modules of HIA.

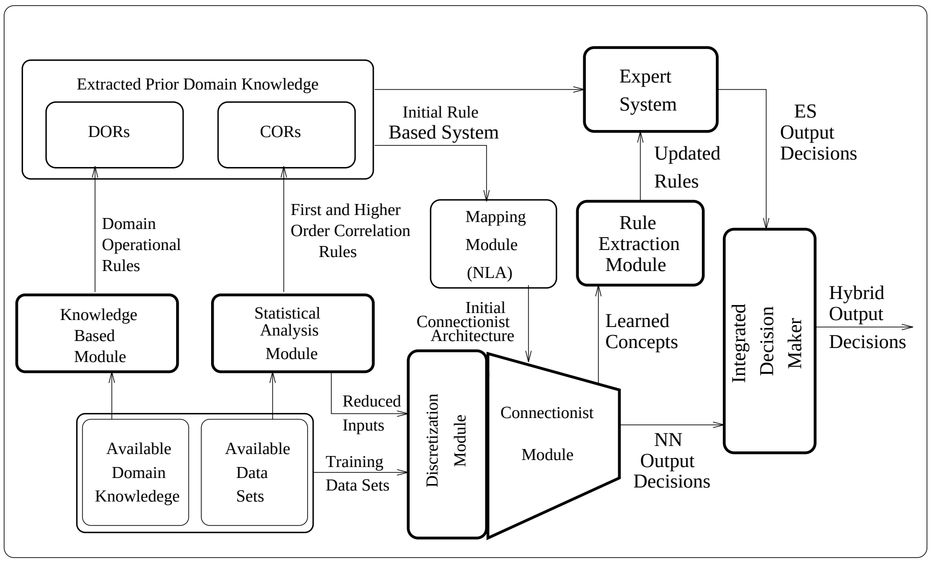

Figure 1. Major components of the Hybrid Intelligent Architecture.

The (optional) rule based system represents the initial domain theory extracted from domain experts in a rule-based format. The acquired rules are mapped into an initial connectionist architecture with uniform structure. The (optional) statistical module analyzes the available data sets and extracts certain correlations between different input parameters and also between input parameters and output decisions. The extracted statistical information is used to provide the mapped initial connectionist architecture with first and higher order input-input and input-output correlation rules. It is also used to provide supplementary rules to an initial rule-based system.

Before training the initial connectionist architecture, a fuzzy subsystem incorporating a coarse coding scheme is used to discretize the input parameters into multi-interval inputs with initial mean and variance for each interval. During the training phase of the connectionist architecture, an Augmented Backpropagation Algorithm (ABA) with momentum term is used to refine the discretization parameters and thus enhance the domain parameters. Therefore, the connectionist architecture can improve the efficiency of the domain theory and incorporate it in its topology.

At the end of the training phase, the final connectionist architecture, with the updated weights and links, can be viewed as a revised domain theory. It can be used to update the initial expert system with new learned concepts. Moreover, it can be converted back, if needed, to a rule based format to achieve the power of explanation [2], [12], [40]. Furthermore, one can use an integrated decision maker to combine the decisions taken by the updated expert system and the trained connectionist architecture and provide the combined decisions to the user.

The rest of this chapter describes in detail the different modules of HIA. In Sections 2-6, we elaborate upon the concepts of discretizing continuous inputs into multi-interval inputs, mapping available domain knowledge into a connectionist architecture, and enhancing the discretization parameters during the training phase. Sections 7-9 summarize options for rule extraction and output integration. Sections 10 and 11 describe two applications of HIA: (1) controlling water reservoirs of the Colorado river around Austin, and (2) characterizing and classifying the Wisconsin Breast Cancer Data Set. In the concluding section we comment on the relation between HIA and some other hybrid and fuzzy approaches, summarize the significance of the current work and point to future directions. Further details on HIA can be found in [34].

## 2 Knowledge Based Module for Representation of Initial Domain Knowledge

As depicted in the bottom left of Figure 1, available domain information can be divided into two parts: the knowledge that represents the operational rules of the domain and the data sets that represent the historical records of the application domain.

The first module in HIA is a knowledge based module that is used to represent the initial domain knowledge through a well-defined format named *Domain Operational Rule (DOR)* format. The DORs are built using only the basic domain primitives that can be acquired easily from the domain without consuming much time or effort. The basic components needed to build the DORs are: (i) the domain objects and their attributes; (ii) the relationship between the domain objects; and (iii) the valid range of the attributes. These basic components represent the initial domain theory and may not be sufficient for representing the complete problem in a rule-based format. However, they can be used to build an initial rule-based expert system. The **DOR** format is a general rule-based format that can be used to represent rule-based systems with and without certainty factors. In case of rule-based systems without certainty factors, the value of $cf$ is replaced by "1" in each rule. The following rules describe the syntax of the DOR format using the Backus-Naur Form (BNF):

**If** $Compound\text{-}Condition\ [OR\ Compound\text{-}Condition]^{*} \xrightarrow{cf} Consequent^{+}$

$Compound\text{-}Condition$ ::= $Simple\text{-}Condition \mid Simple\text{-}Condition$ "AND" $Compound\text{-}Condition$

$Simple\text{-}Condition$ ::= $[NOT]\ Boolean\text{-}Expression$

$Consequent$ ::= $Output\text{-}Variable$

where the symbol ::= means *"to be written as"*, $\mid$ a vertical bar to represent choices, $[\cdot]$ is an optional term that can be repeated one time, $[\cdot]^{*}$ is an optional term that can be repeated zero or more times, and $[\cdot]^{+}$ is a term that can be repeated one or more times.

In many real applications, rules are not always fully true. Therefore, each rule represented in the DOR format has an attached certainty factor value, $cf$, which indicates the measure of belief, or disbelief if it is negative, in the rule consequent provided the premises (left hand side) of the rule are true. It is important to mention that:

- Rule consequents in the DOR format are not permitted to be used as conditions in any other rule. Such a restriction was introduced to avoid increasing the number of hidden layers of the connectionist architecture and hence reduces its complexity. This restriction leads to a simpler uniform connectionist network, as seen later.

- In spite of this restriction, the DOR format can be used for rule-based systems that allow rule consequents to be used as conditions (premises) in other rules. Such systems can be represented in the DOR format by replacing the condition, say of rule $R_n$, that has appeared as a consequent in another rule, say rule $R_m$, by the left hand side of the latter rule ($R_m$).

- The DOR format does not have any restriction on the number of conditions per rule or the order of logical operators in any of its rules.

- The rules represented in the DOR format are mapped directly into a corresponding initial connectionist architecture without any limitation on the number or the order of the operators in each rule. Another approach of mapping domain operational rules into an initial connectionist architecture is to convert them to another set of rules with only conjuncted (anded) premises to simplify the mapping operation [8]. The latter approach simplifies the mapping phase but it increases the complexity of the mapped connectionist architecture.

## 3 Extraction of Supplementary Rules via the Statistical Analysis Module

In many application domains, extracting complete domain operational rules from domain representatives suffers from the knowledge acquisition bottleneck [16], [30]. Therefore, we need to seek another source of information to get more prior knowledge from the application domain, such as statistical information from available datasets. We have investigated two simple statistical approaches to extract prior domain knowledge from the available data sets. As shown in Figure 2, the first approach is to extract the first and the higher order correlation coefficients of the available data sets to generate supplementary and constraint rules to the DORs extracted by the knowledge based module. The second approach is to project the input features vector into a lower dimensional input space and only deal with the most intrinsic input features [7], [14]. These two approaches can be done independently and their results can be combined with the extracted initial DORs. The following subsection presents how supplementary correlation rules can be extracted from available data sets and then used to update the initial rule-based system. Then the next subsection presents how input features can be projected into a lower dimensional input space.

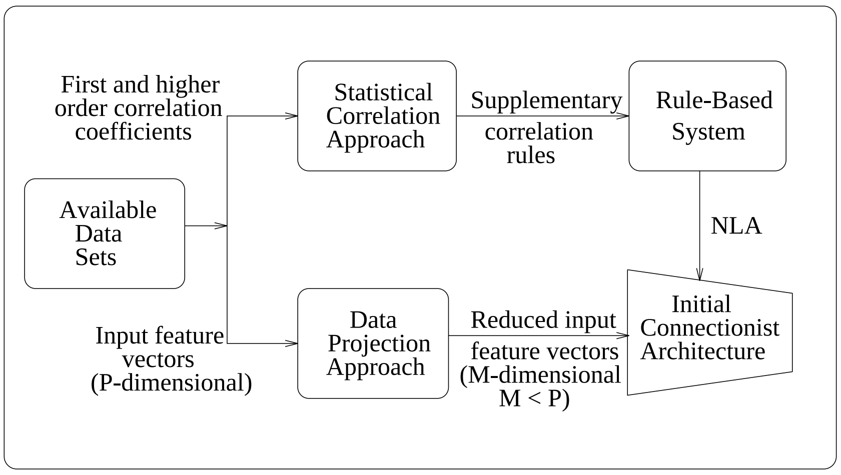
First and higher order correlation coefficients
Statistical Correlation Approach
Supplementary correlation rules
Rule-Based System
Available Data Sets
NLA
Input feature vectors (P-dimensional)
Data Projection Approach
Reduced input feature vectors (M-dimensional M < P)
Initial Connectionist Architecture

Figure 2. Two statistical approaches to extract additional domain knowledge.

## 3.1 Extraction of Correlation Rules

The correlation approach of the statistical module analyzes available data sets by extracting *"certain"* correlation rules between each pair of inputs and also between each input-output parameter. In addition, it extracts the main statistics of each input and output parameter (e.g., the mean and the variance). These statistics are used later to initialize the adaptive parameters of the discretization module.

Assume that $X$ represents an input feature vector and $Y$ represents an output decision vector; let $CC_X = \frac{CO(X)}{\sqrt{\sigma_X}}$ and $CC_{XY} = \frac{CO(X,Y)}{\sqrt{\sigma_X \sigma_Y}}$, where CO(·) represents the covariance matrix. Thus $CC_X$ is the input correlation coefficients matrix and $CC_{XY}$ represents the cross-correlation coefficients matrix between the input and the output vectors $X$ and $Y$. After computing $CC_X$ and $CC_{XY}$, a threshold value, $\theta_0$, is chosen based on the application domain (usually $\theta_0 \geq 0.80$). Based on the chosen threshold $\theta_0$ and the elements of the correlation coefficients matrices $CC_X$ and $CC_{XY}$, the statistical module starts generating COrrelation Rules, named CORs. The statistical analysis module extracts input-input and input-output CORs based on the following algorithm:

---

1. *Let $c_{ij}$ be the correlation coefficient between any two pair of parameters $X_i$ and $X_j$.*

2. **IF** $c_{ij} \geq \theta_0$
   **THEN** *create the COR $R_1$:*

   $$IF \quad X_i \quad \xrightarrow{w_1} \quad X_j \tag{$R_1$}$$

   *where the value of $w_1$ represents the confidence level of the generated rule and $w_1 = cij$.*

3. **IF** $c_{ij} \leq -\theta_0$
   **THEN** *create the COR $R_2$:*

   $$IF \quad NOT \quad X_i \quad \xrightarrow{w_2} \quad X_j \tag{$R_2$}$$

   *where the value of $w_2$ equals the magnitude of $c_{ij}$.*

4. **IF** $-\theta_0 > c_{ij} < \theta_0$
   **THEN** *no rule is generated and the statistical module at this point can not conclude any certain correlation between these two parameters.*

---

The CORs generated by the statistical module are represented in the same DOR format to match the initial rule-based module extracted by the knowledge-based module. Therefore, the CORs and the DORs can be combined together with no additional overhead.

The CORs generated by the statistical module can be used as a constraint or as supplementary rules to the DORs and hence help initializing the connectionist architecture with more prior domain knowledge. The following three cases describe how CORs extracted by the statistical module can be used to simplify, maintain, and support the DORs.

1. **Case1:**
   Assume that the knowledge-based module extracts rule $R_3$, from the domain experts, and the statistical analysis module extracts rule $R_4$ based on the correlation between $X_1$ and $X_2$.

   $$IF \quad X_1 \quad AND \quad X_2 \quad \xrightarrow{w_3} \quad Y_1 \qquad (R_3)$$
   $$IF \quad X_1 \quad \xrightarrow{w_4} \quad X_2 \qquad (R_4)$$

   Therefore, rule $R_4$ can be used to simplify the previous DOR $R_3$ and a new rule $R_5$ is generated to replace both $R_3$ and $R_4$. Note that $w_4$ should be $\geq$ 0.80.

   $$IF \quad X_1 \quad \xrightarrow{w_5} \quad Y_1, \qquad (R_5)$$

   where $w_5 = w_3 \times w_4$. Note that the new rule $R_5$ does not depend on $X_2$ which is highly correlated with $X_1$ based on the COR $R_4$. The logical interpretation of rule $R_5$ results from combining the semantics of both $R_3$ and $R_4$ as follows:

   *"IF $X_1$ is true THEN $X_2$ is true (with $w_4$ confidence measure) AND IF $X_1$ AND $X_2$ are both true THEN $Y_1$ is true with a confidence level = $w_3$"*. This interpretation can be simplified to: *"IF $X_1$ is true; which implicitly implies that $X_2$ is true; THEN $Y_1$ is true with a confidence level $= w_5$"* which represents the semantics of rule $R_5$.

2. **Case2:**
   If the knowledge-based module extracts rule $R_3$ and the statistical module extracts rule $R_6$.

$$IF \qquad NOT \quad X_1 \quad \xrightarrow{.96} \quad X_2 \qquad (R_6)$$

In this case rule $R_3$ cannot be fired any more whatever the value of $X_1$ (i.e., in either cases if $X_1$ is true or false) because $R_6$ is considered as a constraint (in this example a strong contradiction) rule to the DOR $R_3$.

3. **Case3:**
The statistical module can extract CORs which do not exist in the DORs. As an example, if the statistical module extracts rule $R_7$

$$IF \qquad X_3 \quad \xrightarrow{.89} \quad Y_2 \qquad (R_7)$$

and there were no other rules in the DORs to represent the logical relationship between $X_3$ and $Y_2$. In this case the generated COR $R_7$ is added to the extracted DORs.

Based on the previous cases, the statistical module can provide the DORs with either a constraint or supplementary CORs. Moreover, if there were no DORs extracted from the application domain, the statistical module is used to generate correlation rules and represent it in the same DOR format. See the experimental results presented in Section 11.

After combining the rules extracted by the knowledge-based and the statistical modules and representing them in the DOR format, the Node-Links Algorithm (NLA) is used to map these combined rules into an initial connectionist architecture.

## 3.2 Reducing the Input Dimensionality

In many application domains the input data are noisy and may have some redundancy. To obtain better network performance it is important to retain the most intrinsic information in the input data and reduce network complexity at the same time, if it is possible. We use Principal Component Analysis (PCA), a well known technique in multivariate analysis, for this purpose [7], [14].

As a preprocessing step, the correlation matrices represented by $CC_X$ of Equation 1 are used first to determine highly correlated input pairs.

Then, PCA is applied after one variable is removed from each highly correlated pair. This typically enhances the PCA algorithm performance while reducing the connectionist architecture input dimensionality. The resulting feature vector from the PCA algorithm is used as an input to the constructed neural network.

The experimental results presented in Section 11 illustrate how we used the statistics of a public domain data set to extract additional prior domain knowledge.

## 4 The Mapping Module

The Node-Links Algorithm (NLA) utilizes a set of mapping principles to map the initial domain theory, represented in the DOR/COR format, into an initial connectionist architecture that can learn more new domain concepts during the training phase. It results in a three layer AND-OR tree, as exemplified by Figure 3. Note that a *Negated Simple-Condition* is translated into a negative initial weight ($-0.6$ or $-0.7$) for the corresponding link. Also, the NLA creates a hidden node even when there is only one *Simple-Condition* in the premise. This type of hidden node is named *self-anded* hidden node, because it ANDs one input node with itself. Therefore, output nodes are viewed as OR nodes and hidden nodes are viewed either as AND or as *self-anded* nodes. The NLA creates a light link between each *self-anded* node, as well as each AND node, and all other input and output nodes that are not linked with it. Introducing such *self-anded* hidden nodes and light links provides the initial connectionist architecture with the power to learn more domain knowledge and extract new features during the training phase. The overhead due to the introduction of the self-anded nodes and their related links is much less than that incurred by interrupting the training phase and adding, heuristically, a random number of hidden nodes [9]. The initial connectionist architecture generated by the NLA has only three layers, independent of the hierarchy of the initial rule-based system and regardless of the nature of the application domain. Moreover, all hidden nodes functionally implement soft conjunctions and use the sigmoid activation function, which clearly improves the training phase of the initial connectionist architecture. This is in contrast to models that have variable network structure, based on the application domain, and hidden units with different functionalities [6], [10], [44].

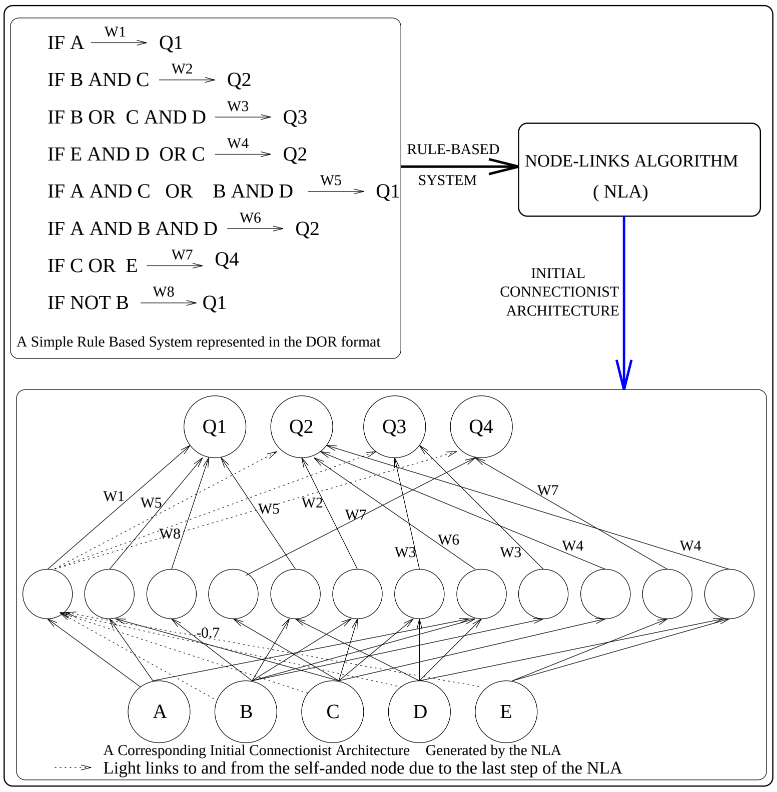
IF A $\xrightarrow{W1}$ Q1
IF B AND C $\xrightarrow{W2}$ Q2
IF B OR C AND D $\xrightarrow{W3}$ Q3
IF E AND D OR C $\xrightarrow{W4}$ Q2
IF A AND C OR B AND D $\xrightarrow{W5}$ Q1
IF A AND B AND D $\xrightarrow{W6}$ Q2
IF C OR E $\xrightarrow{W7}$ Q4
IF NOT B $\xrightarrow{W8}$ Q1
A Simple Rule Based System represented in the DOR format
RULE-BASED SYSTEM
NODE-LINKS ALGORITHM ( NLA)
INITIAL CONNECTIONIST ARCHITECTURE
Q1
Q2
Q3
Q4
W1
W5
W8
W5
W2
W7
W3
W6
W3
W7
W4
W4
-0.7
A
B
C
D
E
A Corresponding Initial Connectionist Architecture Generated by the NLA
Light links to and from the self-anded node due to the last step of the NLA

Figure 3. From rule-based system to initial connectionist architecture using the Node-Links Algorithm.

## 5 The Discretization Module

Measured inputs in control domains are often continuous. Since the operational rules that represent the system use multi-interval ranges to describe the application domain, a discretization function is needed to map continuous inputs into multiple interval inputs. Assume that a continuous measured input $z$ always lies in the range $[a, b]$. A discretization function is used to map it into a corresponding vector $X$: $(x_1, x_2, ..., x_n)$, where $x_i \in [0, 1]$, $\forall i$ and $n$ is the number of discretized intervals. In a basic symbolic system, exactly one of the $x_i$s is set to 1 for a given $z$ value, and all others are zero. However, we prefer a continuous discretization approach to a binary discretization since it allows “coarse coding,” i.e., more than one interval can be active at the same time with different certainty values, based on the value of the measured input $z$. Coarse coding is a more robust representation of noisy data, which is a prime objective here. This discretization process is a typical fuzzification approach for determining the degree of membership of the measured input $z$ in each interval $i$. The value of each element $x_i$ is interpreted as the measure of belief that $z$ falls in the $i^{th}$ interval [43].

A Gaussian function with mean $\mu_i$ and standard deviation $\sigma_i$ is selected to represent the distribution of the measured input $z$ over each interval $i$; so $n$ Gaussian curves are used to fuzzify $z$ into $n$ intervals. [1] The technique is illustrated in Figure 4, where a continuous measured input $z$ gets fuzzified into an input vector $X$, resulting in $x_1 = 0.25$ and d$x_2 = 0.75$. This fuzzification is done as a preprocessing phase to the initial connectionist architecture. The output of the fuzzification process, $X$, represents the activation values of the input nodes of the initial connectionist architecture, where each interval is represented by an input node. If the application domain has $k$ continuous measured inputs the fuzzification approach results in a total of $\sum_{i=0}^{k} n_k$ input nodes, where $n_k$ is the number of discretized intervals of the $k^{th}$ measured input.

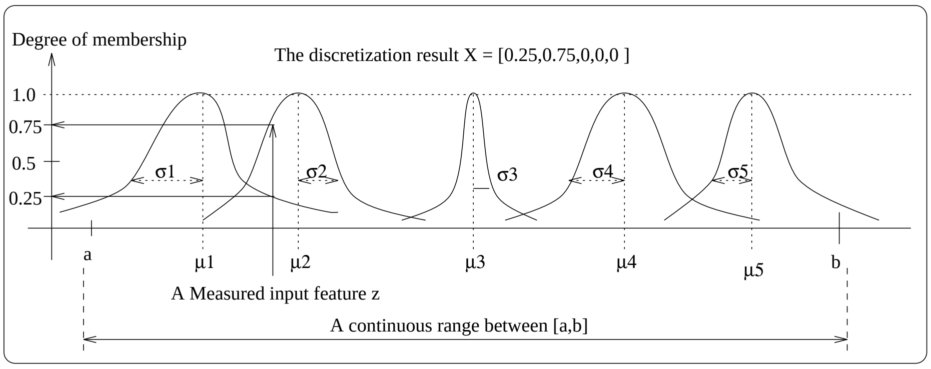
Degree of membership
The discretization result X = [0.25,0.75,0,0,0 ]
1.0
0.75
0.5
0.25
σ1
σ2
σ3
σ4
σ5
a
μ1
μ2
μ3
μ4
μ5
b
A Measured input feature z
A continuous range between [a,b]

Figure 4. Discretizing a continuous measured input into $n$ intervals using $n$ Gaussian functions.

[1]The choice of the differentiable Gaussian function instead of the typical triangular membership functions used in fuzzy logic is important as it facilitates membership adaptation, as described in the next section.

## 6 Refining Input Characterization

The initial connectionist architecture is trained by the output vectors ($X$s) of the fuzzification function. Assuming that the measured input values are normally distributed within each interval $i$ with mean $\mu_i$ and standard deviation $\sigma_i$, the Gaussian functions:

$$x_i = f_i(z) = e^{-\frac{1}{2}(\frac{z-\mu_i}{\sigma_i})^2} \tag{1}$$

are used to discretize the measured input value $z$. An Augmented version of the Backpropagation Algorithm, **ABA**, with momentum term is used to train the initial architecture and stochastically search for the optimal weights to and from all hidden nodes (anded and self-anded nodes). Moreover, the ABA is used to refine the initial discretization parameters $\mu_i$ and $\sigma_i$ for each interval $i$. The ABA calculates the stochastic gradient descents of the output error with respect to $\mu_i$ and $\sigma_i$ and propagates them one more step back to the fuzzification function, i.e., to the external inputs of the connectionist architecture. Refining the discretization parameters ($\mu_i$, $\sigma_i$) helps the connectionist architecture to extract features from the measured inputs that are more discriminating and thus enhances the decision process. The chain rule was used to derive the derivative of the output error, $E$, with respect to $\mu_i$ and $\sigma_i$:

$$\frac{\partial E}{\partial \mu_i} = \frac{\partial E}{\partial f_i(z)} \cdot \frac{\partial f_i(z)}{\partial \mu_i} = \frac{1}{\sigma_i^2} \cdot (z - \mu_i) \cdot \sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}} \tag{2}$$

$$\frac{\partial E}{\partial \sigma_i} = \frac{\partial E}{\partial f_i(z)} \cdot \frac{\partial f_i(z)}{\partial \sigma_i} = \frac{(z - \mu_i)^2}{{\sigma_i}^3} \cdot \sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}} \tag{3}$$

where the term $\sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}}$ represents the gradient descent of the output error propagated back to all the $h$ hidden nodes linked to $i^{th}$ input node. Note that the $\frac{\partial E}{\partial w_{ij}}$s do not need to be recomputed as they are already obtained from updating the weights into the hidden units. The center and width of the $i^{th}$ interval are adjusted as follows:

$$\mu_{inew} = \mu_{iold} - \eta \cdot \frac{\partial E}{\partial \mu_i} + MomentumTerm \tag{4}$$

$$\sigma_{inew} = \sigma_{iold} - \eta \cdot \frac{\partial E}{\partial \sigma_i} + MomentumTerm \tag{5}$$

# 7 Rule Extraction

Extraction of symbolic rules from trained neural networks 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 [32]. For models such as MLPs 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.

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

The rule extraction module of HIA maps the trained connectionist architecture back into a rule based format. This mapping is much harder than the mapping from an initial rule based system to an initial connectionist architecture because: (i) one should guarantee that the extracted domain concepts should not contradict with certain concepts that are known to be true about the domain, (ii) one should refine uncertain domain concepts, and (iii) new concepts may get extracted. An efficient rule extraction module should be able to deal with these three 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. 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.

A variety of rule-extraction techniques have been proposed in the recent literature [2], [36], [39]. Also see the rule extraction home page at: `http://www.fit.qut.edu.au/~robert/rulex.html`. 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 [28], KT [9] and Subset [40] are three notable rule extraction algorithms in this category, which we describe as Link Rule Extraction Techniques.

In this section we summarize three recent techniques for extracting rules from trained feedforward ANNs. The first approach is a binary Black-box Rule Extraction technique. The second and the third approaches belong to the Link Rule Extraction category. Details can be found in [36].

## 7.1 First Technique (BIO-RE)

The first approach 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. It is surprisingly effective within its domain of applicability. The idea underlying BIO-RE is to construct a truth table that represents all valid input-output mappings of the trained network. BIO-RE then applies a logic minimization tool, Espresso [29], to this truth table to generate a set of optimal binary rules that represent the behavior of the trained networks. For example, an extracted rule: "**IF** $Y_1$ **AND NOT** $Y_2 \longrightarrow O_1$", is rewritten as "**IF** $X_1 > \mu_1$ **AND** $X_2 \leq \mu_2 \longrightarrow O_1$", where $\mu_i$ is set to be the threshold of $X_i$ (see Table 2 for examples). 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.

## 7.2 Second Technique (Partial-RE)

The idea underlying Partial-RE algorithm is that it first sorts both positive and negative incoming links for each hidden and output node in descending order into two different sets based on their weight values. Starting from the highest positive weight (say $i$), it searches for individual incoming links that can cause a node $j$ (hidden/output) 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. If a node $i$ was found strong enough to activate a node $j$, then this node is marked and cannot be used in any further combinations when checking the same 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. Partial-RE performs the same procedure for negative links and small combinations of both positive and negative links if the required number of premises in a rule is $> 1$. 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. See Table 3 for examples.

## 7.3 Third Technique (Full-RE)

Full-RE first generates intermediate rules in the format:

$$\textbf{IF } [\ (c_1 \cdot X_1 + c_2 \cdot X_2 + \cdots + c_n \cdot X_n) >= \lambda_j] \xrightarrow{cf} Consequent_j,$$

where: $c_i$ is a constant representing the effect of the $i^{th}$ input ($X_i$) on $Consequent_j$ and $\lambda_j$ is a constant determined based on the activation function of node $j$ to make it active. If node $j$ is in the layer above node $i$ then $c_i$ represents the weight value $w_{ji}$ of the link between these two nodes. In cases where the neural network inputs ($X_i$s) are continuous valued inputs, then 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 has to be 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 into $k$ intervals [21]. When Full-RE finds more than one discretization value of an input $X_i$ that can satisfy the intermediate rule (i.e., the rule has more than one feasible solution) then it chooses the minimum or the maximum of these values based on the $sign$ of the corresponding effect parameter $c_i$. If $c_i$ is negative then Full-RE chooses the minimum discretization value of $X_i$; otherwise it chooses the maximum 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. Final rules extracted by Full-RE are represented in the same format of 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 4 for examples.

## 8 Rule Evaluation and Ordering Procedure for the Refined Expert System

To evaluate the performance of rules extracted from trained networks by any of the three presented techniques (or by any other rule extraction approach), a simple rule evaluation procedure which attaches three performance measures to each extracted rule is developed. The three performance measures used to determine the order of the extracted rules are:

(i) **The soundness measure:** it measures how many times each rule is correctly fired.

(ii) **The completeness measure:** a completeness measure attached to a rule represents 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. For each extracted set of rules with the same consequent, if the sum of the completeness measures of all rules in this set equals the total number of input patterns having the corresponding output then this set of extracted rules is 100% complete with respect to that consequent. An extracted rule with zero completeness measure but having a soundness measure $> 0$ means that there is a preceding rule(s), in the order of rule application, that covers the same input patterns that this rule covers. Such a rule may be removed.

(iii) **The false-alarm measure:** it measures how many times a rule is misfired over the available data set. While the values of both the completeness and false-alarm measures depend on the order of rule application and the inference engine the soundness measure does not.

## 8.1 The Rule Ordering Procedure

An expert system requires a set of rules as well as an inference engine to examine the data, determine which rules are applicable, and prioritize or resolve conflicts among multiple applicable rules. A simple way of conflict resolution is to order the rules, and execute the first applicable rule in this ordering. Finding the optimal ordering of extracted rules is a combinatorial problem. So the following *"greedy"* algorithm to order any set of extracted rules, based on the three performance measures, is developed. The rule ordering algorithm first creates a list $L$ that contains all extracted rules. Assume that the list $L$ 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[2]. Initially, $L_h$ is empty and $L_t$ includes all the extracted rules. A performance 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 $L_h$[3].
```

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

[3]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 $L_h$.

```
        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 $L_t$. This means that the rules in $L_h$
        cover this rule.
3. END DO.
```

In this chapter, all rules extracted by our approaches are ordered using the above rule ordering 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.

## 9 The Integrated Decision Maker

The main objective of combining or integrating different learning modules is to increase the overall generalization capability. Since the set of extracted rules is an *"approximated symbolic representation"* of the embedded knowledge in the internal structure of the corresponding trained network, it is expected that when an input is applied to the extracted rules and the trained network, they will usually both provide the same output decision (see Table 6 for examples). The integration module should be able to choose the *"better"* output decision when the two decisions differ, and to compute the certainty factor of the final output decision. When the two output decisions are different, the integration module can use the following selection criteria to select a suitable decision.

1. Select the sub-system (i.e., the set of extracted rules or the trained ANN) with the highest overall performance if none of the following conditions are satisfied:

2. For any mismatched pair of output decisions, check the value of the neural network output decision (i.e., the activation value of the corresponding output node of the neural net before thresholding).
   (a) If the extracted rule-base is indicated by Rule 1, but the neural network output is significantly high, then choose the neural network instead to provide the final decision. Also, report that

the extracted rule-base was not able to identify this case, so that a new rule can be asserted in the current knowledge base to handle such cases in the future.

(b) If the neural network is indicated by Rule 1, but the network output is significantly low, then choose the extracted rule-base instead to provide the final output of this case. Also, report that the neural network was not able to identify this case, so that it can be retrained. This case can also be applied if the difference between the two highest activation values of the neural network output nodes is not significant.

This simple heuristic criterion of selecting one of the two mismatched output decisions was applied for all the three architectures and their corresponding set of extracted rules using the breast cancer problem. The implementation results are given in Table 6.

## 10 Application: Controlling Water Reservoirs

There are several dams and lakes on the Colorado river near Austin. The decision of specifying the amount of water that should be released from any of these dams and lakes is a complicated process. The Lower Colorado River Authority (LCRA) determines this decision for each dam or lake based on the current elevation of the water in the lake, the inflow rate from upstream dams and lakes, the outflow rate from the current lake, the predicted weather (rain fall rate), the predicted elevation of the downstream dams and lakes, and many other factors.

The LCRA uses upstream and downstream gages to monitor and forecast lake levels. There are two main modes of operation for the Highland Lakes: the first is a daily operation in which downstream demands for water are met by daily releases from Lake Buchanan and Travis to supplement the flow of the lower river. The second is for flood control, which primarily concerns Lake Travis since it is the only reservoir with a dedicated flood pool. When the Colorado river downstream from Highland Lake approaches the warning stage at any of the following downstream gages, the rules of the flood control operating mode are used to determine water release.

We acquired the 18 main operational rules for controlling the flood gates on the Colorado river in the greater Austin area from different documents issued by the LCRA after the 1991 Christmas flood around Austin [19]. Two of the rules regulating the control of Mansfield Dam on Lake Travis, expressed in DOR format, are:

```
If Projected-Level(Lake-Travis,t0) >= 710 AND
            Projected-Level(Lake-Travis,t0) <= 714 AND
            Projected-Level(Lake-Austin,t1) <= 24.8 AND
            Projected-Level(Bastrop,t2) <= 26.7
      Then
               Open-Up-To 10 Flood-Gates.

If Projected-Level(Lake-Travis,t0) <= 710 AND
            Projected-Level(Lake-Travis,t0) >= 691 AND
            Projected-Level(Lake-Austin,t1) <= 20.5 AND
            Projected-Level(Bastrop,t2) <= 25.5
      Then
               Open-Up-To 6 Flood-Gates.
```

Lake Austin and Bastrop are downstream from Mansfield Dam, $t0$ is the time when the water level at Lake Travis is measured, and $t1 - t0$ and $t2 - t0$ are the approximate times taken for the released water from Lake Travis to reach Lake Austin and Bastrop respectively (approx. 2 hrs and 24 hrs). All values are measured in feet above mean sea level.

## 10.1 Implementation Results

The HIA knowledge based module was used to implement the extracted domain knowledge in the DOR format. A data set representing 600 patterns was gathered from the LCRA historical records of the different dams and lakes. The acquired data set was divided into two sets. The first one had 400 patterns and was used as the training set; the second served as the validation set. Each pattern includes the measured elevation of three lakes at some given time and the corresponding best decision based on the extracted domain knowledge represented by the DOR.

The coarse coding scheme described in Section 5 was used to discretize each input (measured elevation $z$) into a multi-interval input vector ($X$). Each discretized interval $i$ corresponds to one specific DOR $Simple$-$Condition$ and has an initial mean $\mu_i$ and standard deviation $\sigma_i$. The Node-Links Algorithm (NLA) was used to map the DORs into a one hidden layer initial connectionist architecture. The resulting architecture has 23 input nodes (representing the different discretized intervals), 18 hidden nodes (representing either AND or *self-anded* soft conjunction concepts), and 8 output nodes representing the possible decisions at any time (i.e., how many flood gates should be opened).

During the training phase, the Augmented Backpropagation Algorithm (ABA) with momentum term stochastically searched the weight space to find the optimal weights and also used the partial derivatives of the output error, as described by Equations 2 through 5, to refine the initial discretization parameters ($\mu_i$ and $\sigma_i$) of the 23 input nodes. Note that each input node in the mapped architecture corresponds to a discretized interval.

By the end of the learning process, we found that the means ($\mu$) of seven different intervals were shifted and the standard deviations ($\sigma$) of four of them were also significantly changed from their initial values. The change in the values of the discretization parameters readily reflects how the ABA exploits the training data set to refine the initial domain knowledge represented by the rule based module. Actually, any change in the discretization parameters directly affects the input of the connectionist architecture and hence its output decisions. Therefore, any refinement in these parameters enhances the output performance of the trained connectionist architecture.

After the training phase, the validation set (200 patterns) was used to observe how the HIA will perform in real flood situations. The decisions taken by the HIA, which uses the ABA for training the mapped connectionist architecture and also for refining domain parameters, were compared with the decisions taken by two different connectionist architectures. The first one is an MLP (with one hidden layer) initialized randomly without any prior domain knowledge and trained by the conventional backpropagation with a momentum term. The second is an MLP (also with one hidden layer) initialized by the same initial rules that were used to initialize the HIA and trained by the conventional backpropagation with a momentum term (i.e., no refinement of discretization parameters was done).

Table 1. Test results of Colorado river problem.

| No. | HIA | | MLP | | MLP | |
|---|---|---|---|---|---|---|
| of | with refined DOR | | initialized randomly | | with initial DOR | |
| epochs | MSE | % match | MSE | % match | MSE | % match |
| 1 | 0.096 | 76.176 | 1.367 | 64.0 | 0.344 | 70.9 |
| 10 | 0.015 | 93.323 | 0.889 | 68.5 | 0.248 | 72.6 |
| 20 | 0.014 | 94.234 | 0.334 | 71.7 | 0.103 | 74.7 |
| 30 | 0.013 | 94.234 | 0.233 | 72.38 | 0.094 | 76.3 |

The output decisions taken by the three architectures (to specify how many flood gates should be opened at a given time) were compared with the desired decisions that should be taken in each situation based on LCRA operational rules. Table 1 provides a summary of the testing results. The presented results are the average of 10 runs. They represent the performance of each architecture when the test data sets were applied. Two parameters are presented for each architecture, the mean square error and the percentage of patterns for which the maximum output value matched with the desired decision based on the LCRA operational rules. Note that the LCRA selects only one decision per pattern, while the neural network indicates support for each decision by the value of the corresponding output. To compute MSE, we used a target value of “1” for output node corresponding to the desired decision and “0” for all other outputs. The implementation results depicted in Table 1 show that:

- The randomly initialized MLP did not perform well compared with the other two architectures.
- The MLP initialized by the initial domain rules and trained with the conventional backpropagation performed somewhat better than the randomly initialized MLP. However, the performance of this architecture did not improve much on further training.
- The HIA did perform much better than the other two architectures and its performance improved significantly by increasing the number of training epochs.

The improved performance of the HIA is mainly due to the continuous refinement of the discretization parameters during each training epoch.

### 10.2 Rule Extraction

After the learning phase, the connectionist module of HIA was examined by Partial-RE to find out if any new rules were created during the training. Some of the extracted rules were already there in the initial DORs and others were subsets of existing rules. However, some new and useful rules were also found. For example, in the initial rule based system extracted from the LCRA documents, there was a rule specifying that: if the predicted level of Lake Travis is between 681 and 691 feet msl (mean sea level) then release up to 5,000 cfs (cubic feet per second) if the river, with the release, is no higher than 20.5 ft. at Austin and 25.1 ft. at Bastrop. HIA extracted three new and useful fine rules instead of the previous coarse rule. The first new rule extracted by HIA is: if the predicted level of Lake Travis is between 681 and 683 feet msl then water does not need to be released if the river is no higher than 16.0 ft. at Austin and 18.0 ft. at Bastrop. The second rule is: if the predicted level of Lake Travis is between 683 and 685 feet msl then release up to 3,000 cfs if the river, with the release, is no higher than 16.0 ft. at Austin and 18.0 ft. at Bastrop. The third rule is: if the predicted level of Lake Travis is between 685 and 691 feet msl then release up to 5,000 cfs if the river, with the release, is no higher than 20.5 ft. at Austin and 25.1 ft. at Bastrop. The previous three new extracted rules are useful where they refine the original coarse rule based on the combination of the upstream (Lake Travis) and downstream (Lake Austin and Bastrop) conditions which primarily determine the decision on releasing water. More comprehensive rules can be extracted by applying the Full-RE technique.

## 11 Application of the Statistical Approach

The statistical approaches introduced in Section 3 were not used in the water reservoirs control problem because the extracted DORs were reasonably comprehensive. In this section, we report results of applying the statistical module to the breast cancer classification problem where there is no pre-existing domain knowledge (i.e., no DORs) but a public domain

Table 2. Rules extracted from network *"Cancer-Bin"* by BIO-RE technique.

| No. | Rule Body | Cancer Class | Performance Measures | | |
|---|---|---|---|---|---|
| | | | Sound-ness | Complete-ness | False Alarm |
| $R_1$ | 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 |
| $R_2$ | 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 |
| $R_3$ | 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 |
| $R_4$ | 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 |
| $R_5$ | 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 |
| $R_6$ | If $X_1 \geq 4.1$ and $X_3 \geq 3.0$ | Malignant | 200/239 | 199/239 | 15/683 |
| $R_7$ | If $X_3 \geq 3.0$ and $X_7 \geq 3.3$ | Malignant | 187/239 | 27/239 | 2/683 |
| $R_8$ | If $X_3 \geq 3.0$ and $X_8 \geq 2.7$ | Malignant | 187/239 | 3/239 | 0/683 |
| $R_9$ | If $X_1 \geq 4.1$ and $X_7 \geq 3.3$ | Malignant | 167/239 | 7/239 | 1/683 |
| $R_{10}$ | If $X_1 \geq 4.1$ and $X_9 \geq 1.5$ | Malignant | 100/239 | 1 | 3 |
| $R_{11}$ | Default Class | Benign | 5/444 | 5/444 | 0/239 |
| Total For Benign Rules | | | | 423/444 | 2/683 |
| Total For Malignant Rules | | | | 237/239 | 21/683 |
| Overall Performance% | | | | $96.63\%$ | 3.37% |

data set is available [3], [23].

The Wisconsin breast cancer data set has nine inputs ($X_1 \cdots X_9$) and two output classes ($Benign$ or $Malignant$). The available 683 instances were divided randomly into a training set of size 341 and a test set of size 342. In all experiments, an MLP network is trained using the backpropagation algorithm with momentum as well as a regularization term [11]. The dimensionality of the breast-cancer input space is reduced from 9 to 6 inputs using PCA. BIO-RE, Partial-RE, and Full-RE are used to extract rules from Cancer-Bin, Cancer-Norm, and Cancer-Cont networks respectively, where the first network is trained with a binarized version of the available data, Cancer-Norm is trained with normalized input patterns, and Cancer-Cont is trained with the original data set after dimensionality reduction. Tables 2, 3, and 4 present three sets of ordered rules extracted by the three rule extraction techniques, along with the corresponding performance measures.

Table 3. Rules extracted from network *"Cancer-Norm"* by Partial-RE technique.

| No. | Rule Body | Cancer Class | CF | Performance Measures: Soundness | Performance Measures: Completeness | Performance Measures: False Alarm |
|---|---|---|---|---|---|---|
| $R_1$ | 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 |
| $R_2$ | 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 |
| $R_3$ | If $X_2 \geq 3.0$ and $X_3 \geq 3.0$ | Malignant | 0.99 | 222/239 | 219/239 | 15/683 |
| $R_4$ | 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 |
| $R_5$ | 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 |
| $R_6$ | If $X_1 \geq 4.1$ and $X_2 \geq 3.0$ | Malignant | 0.99 | 198/239 | 2/239 | 0/683 |
| $R_7$ | 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 |
| $R_8$ | If $X_1 \geq 4.1$ and $X_3 \geq 3.0$ | Malignant | 0.99 | 200/239 | 3/239 | 2/683 |
| $R_9$ | 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 |
| Total For Malignant Rules | | | | | 232/239 | 17/683 |
| Overall Performance% | | | | | 96.49% | 3.51% |

Table 5 provides an overall comparison between the performance of 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 the breast cancer data set did not degrade the performance of the trained networks or of the rules extracted by BIO-RE and Partial-RE from these networks *("Cancer-Bin" and "Cancer-Norm" respectively)*. This is due to the fact that the original input features of the breast cancer problem have the same range (1,10). Table 6 shows the impact of the integration

Table 4. Rules extracted from network *"Cancer-Cont"* by Full-RE technique.

| No. | Rule Body | Cancer Class | CF | Performance Measures Soundness | Completeness | False Alarm |
|---|---|---|---|---|---|---|
| $R_1$ | If $X_1 < 8$ and $X_3 < 3$ | Benign | 0.96 | 394/444 | 394/444 | 5/683 |
| $R_2$ | If $X_2 \geq 2$ and $X_7 \geq 3$ | Malignant | 0.83 | 227/239 | 223/239 | 18/683 |
| $R_3$ | If $X_1 < 8$ and $X_7 < 3$ | Benign | 0.75 | 300/444 | 27/444 | 1/683 |
| $R_4$ | If $X_1 \geq 8$ | Malignant | 0.89 | 123/239 | 9/239 | 1/683 |
| $R_5$ | 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 |
| Total For Malignant Rules | | | | | 232/239 | 19/683 |
| Overall Performance% | | | | | $96.19\%$ | $3.80\%$ |

Table 5. Performance comparison between the sets of extracted rules and their corresponding trained networks for the breast-cancer problem.

| | | Neural Network | | Extracted Rules | |
|---|---|---|---|---|---|
| | | ratio | % correct | ratio | % correct |
| 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 |

method. It is important to mention that the limited gains due to the integration is because of the high degree of agreement between the two modules. Only $22$, $20$, and $14$ out of $683$ outcomes were different respectively for the three experiments. The integration mechanism was able to select correctly $20$, $17$, and $14$ of these mismatches respectively.

## 12 Discussion

Symbolic and neural subsystems can be combined in a wide variety of ways [24]. HIA has the flavor of a transformational hybrid system, whose prototype is the Knowledge Based Neural Network (KBNN) that

Table 6. Overall performance of HIA after applying the integration mechanism using the breast-cancer database.

| | #both correct | #both wrong | #disagreed on | #correct decisions on mismatches |
|---|---|---|---|---|
| Cancer-Bin | 647 | 14 | 22 | 20/22 |
| Cancer-Norm | 647 | 16 | 20 | 17/20 |
| Cancer-Cont | 653 | 16 | 14 | 14/14 |

| | Overall Performance | |
|---|---|---|
| | ratio | % correct |
| Cancer-Bin | 667/683 | 97.77 |
| Cancer-Norm | 664/683 | 97.22 |
| Cancer-Cont | 667/683 | 97.77 |

achieves theory refinement in four phases [8], [13], [15], [36], [42]:

- The rule base representation phase, where the initial domain knowledge is extracted and represented in a symbolic format (e.g., a rule base system).
- The mapping phase, where domain knowledge represented in symbolic form is mapped into an initial connectionist architecture.
- The learning phase, where this connectionist architecture is trained by a set of domain examples.
- The rule extraction phase, where the trained (and thus modified) connectionist architecture is mapped back to a rule based system to provide explanation power.

The main motivation of such KBNN systems is to incorporate the complementary features of knowledge based and neural network paradigms. Knowledge-Based Artificial Neural Network (KBANN) [41] is a notable system that maps domain knowledge, represented in propositional logic (Horn clauses), into a neural network architecture which is then trained using the backpropagation algorithm to refine its mapped domain knowledge. The KBANN algorithm has been applied to two problems from molecular biology and the reported results show that it generalizes better than other learning systems. However, KBANN maps binary rules into a neural network and it is not clear how it can deal with rules with certainty factors. Also, it adds new hidden nodes before the training phase starts, but the difficult question of *how many hidden nodes should be added* is not answered.

RAPTURE is another hybrid system for revising probabilistic knowledge bases that combines connectionist and symbolic learning methods [22]. RAPTURE is capable of dealing with rule based systems that use certainty factors. However, the structure of the mapped network by RAPTURE is domain dependent and the number of network layers, being determined by the hierarchy of the rule base, can become large. Another notable example is the Knowledge Based Conceptual Neural Network (KBCNN) model. KBCNN revises and learns knowledge on the basis of a neural network translated from a rule base which encodes the initial domain theory [8], [9]. In fact, the KBCNN model has some similarities with both KBANN and the RAPTURE.

Researchers have also combined connectionist systems with fuzzy logic to obtain Fuzzy Logic Neural Networks (FLNN). In FLNNs, the neural network subsystem is typically used to adapt membership functions of fuzzy variables [5], or to refine and extract fuzzy rules [20], [37], [38]. A standard fuzzy logic system has four components:

- A fuzzifier, which determines the degree of membership of a crisp input in a fuzzy set.
- A fuzzy rule base, which represents the fuzzy relationships between input-output fuzzy variables. The output of the fuzzy rule base is determined based on the degree of membership specified by the fuzzifier.
- An inference engine, which controls the rule base.
- A defuzzifier, which converts the output of the fuzzy rule base into a crisp value.

Such fuzzy logic systems often suffer from two main problems. First, designing the right membership function that represents each input and output variable may be nontrivial. A common approach is to design an initial membership function, usually triangular or trapezoidal in shape, and subsequently refine it using some heuristic. The second problem is the lack of a learning function that can be adapted to reason about the environment.

These two problems can be alleviated by combining both fuzzy logic and neural network paradigms. In FLNN hybrid systems, a neural network architecture is used to replace the fuzzy rule base module of standard fuzzy logic systems. The Max and Min functions are commonly used as activation functions in this network. Then, a supervised or unsupervised learning algorithm is used instead of the inference engine to adapt network parameters and/or architecture. After training the network with available domain examples, the adapted network is used to refine initial membership functions and fuzzy rule base. Moreover, it may be used to extract new fuzzy rules.

The first five modules of HIA are superficially similar to both KBNN and FLNN hybrid systems. HIA is capable of revising initial domain knowledge and extracting new rules based on training examples. However, it has the following distinguishing features: (1) it generates a uniform neural network architecture because of the constrained DOR format; (2) the neural network architecture generated by HIA has only three layers, independent of the initial rule base hierarchy; (3) it revises the input characterization parameters using a coarse coding fuzzification approach during the training phase which may enhance system performance; (4) it combines a statistical module along with its knowledge based and neural network modules to extract supplementary domain knowledge.

Much of the power of HIA is derived from its completeness. It provides a mechanism for conflict resolution among the extracted rules, and for optional integration of the refined expert system and the trained neural network.

For the problem of controlling the flood gates of the Colorado river in greater Austin, we observe that refining the input characterization along with the domain knowledge incorporated in a connectionist model substantially enhanced the generalization ability of that model. This application also showed the capability of HIA to extract new and useful operational rules from the trained connectionist module. The breast cancer classification problem shows how the statistical module of HIA can enhance the topology of the connectionist module in cases where there is no available prior domain knowledge or in cases where the input features have substantial redundancy. It is remarkable that, using the Full-RE technique, the data set can be characterized with high accuracy using only five rules.

It will be worthwhile to apply HIA to a wider range of problems where both domain knowledge and training data are available, but none is sufficiently comprehensive on its own.

## Acknowledgments

This research was supported in part by ARO contracts DAAG55-98-1-0230 and DAAD19-99-1-0012, and NSF grant ECS-9900353. Ismail Taha was also supported by the Egyptian Government Graduate Fellowship in Electrical and Computer Engineering. We are thankful to Prof. B. Kuipers for bringing the Colorado river problem to our attention.

## References

[1] Aggarwal, J.K., Ghosh, J., Nair, D., and Taha, I. (1996), "A comparative study of three paradigms for object recognition - bayesian statistics, neural networks and expert systems," *Advances In Image Understanding: A Festschrift for Azriel Rosenfeld*, Boyer, K. and Ahuja, N. (Eds.), IEEE Computer Society Press, pp. 241-262.

[2] Andrews, R., Diederich, J., and Tickle, A. (1995), "A survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373-389.

[3] Bennett, K. and Mangasarian, O. (1992), "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software 1*, Gordon and Breach Science Publishers.

[4] Jain, L.C., and Martin, N.M. (Eds.) (1999), *Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms*, CRC Press.

[5] Challo, R., McLauchlan, R., Clark, D., and Omar, S. (1994), "A fuzzy neural hybrid system," *IEEE International Conference on Neural Networks*, Orlando, FL, vol. III, pp. 1654-1657.

[6] Fletcher, J. and Obradovic, Z. (1993), "Combining prior symbolic knowledge and constructive neural network learning," *Connection Science,* vol. 5, no. 3-4, pp. 365-375.

[7] Friedman, J.H. (1994), "An overview of predictive learning and function approximation," *From Statistics to Neural Networks, Proc. NATO/ASI Workshop,* Cherkassky, V., Friedman, J., and Wechsler, H., editors, Springer-Verlag, pp. 1-61.

[8] Fu, L. (1993), "Knowledge-based connectionism for revising domain theories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 173-182.

[9] Fu, L. (1994), *Neural Networks in Computer Intelligence,* McGraw-Hill, Inc.

[10] Gallant, S.I. (1988), "Connectionist expert systems," *Comm. of ACM,* vol. 31, no. 2, pp. 152-169.

[11] Ghosh, J. and Tumer, K. (1994), "Structural adaptation and generalization in supervised feed-forward networks," *Journal of Artificial Neural Networks*, vol. 1, no. 4, pp. 431-458.

[12] Giles, C. and Omlin, C. (1994), "Pruning recurrent neural networks for improved generalization performance," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 848-851.

[13] Glover, C., Silliman, M., Walker, M., and Spelt, P. (1990), "Hybrid neural network and rule-based pattern recognition system capable of self-modification," *Proceedings of SPIE, Application of Artificial Intelligence VIII,* pp. 290-300.

[14] Gonzalez, R. (1993), *Digital Image Processing,* Addison-Wesley.

[15] Hendler, J. (1989), "Marker-passing over microfeatures: towards a hybrid symbolic/connectionist model," *Cognitive Science*, vol. 13, pp. 79-106.

[16] Jackson, P. (1990), *Introduction to Expert Systems*, Addison-Wesley.

[17] Kanal, L. (1974), "Patterns in pattern recognition," *IEEE Trans. Information Theory*, IT-20:697-722.

[18] Lacher, R., Hruska, S., and Kuncicky, D. (1992), "Backpropagation learning in expert networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 62-72.

[19] LCRA (1992). "The flooding of the Colorado: how the system worked to protect central Texas," Technical report, Lower Colorado River Authority, P.O. Box 220, Austin, Texas 78767-0220.

[20] Lin, C. and Lee, C. (1991), "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320-1326.

[21] Liu, H. and Setiono, R. (1995), "Chi2: feature selection and discretization of numeric attributes," *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, pp. 388-391.

[22] Mahoney, J. and Mooney, R. (1993), "Combining connectionist and symbolic learning to refine certainty factor rule bases," *Connection Science*, vol. 5, no. 3-4, pp. 339-364.

[23] Mangasarian, O. and Wolberg, H. (1990), "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, pp. 5, pp. 1-18.

[24] McGarry, K., Wertmer, S., and MacIntyre, J. (1999), "Hybrid neural systems: from simple coupling to fully integrated neural networks," *Neural Computing Surveys*, vol. 2, pp. 62-93.

[25] Medsker, L.R. (1995), *Hybrid Intelligent Systems*, Kluwer Academic, Norwell, MA.

[26] Michalski, R. (1993), “Toward a unified theory of learning: multi-strategy task-adaptive learning,” *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, Buchanan, B. and Wilkins, D., editors, Morgan Kaufmann, San Mateo.

[27] Minsky, M. (1991), “Logical versus analogical or symbolic versus connectionist or neat versus scruffy,” *AI Magazine*, vol. 12, no. 2, pp. 34-51.

[28] Ourston, D. and Mooney, R. (1990), “Changing the rules: a comprehensive approach to theory refinement,” *Proceedings of the Eighth National Conference on Artificial Intelligence*, AAAI Press, pp. 815-820.

[29] Ruddel, R. and Sangiovanni-Vincentelli, A. (1985), “Espresso-MV: algorithms for multiple-valued logic minimization,” *Proceedings of Cust. Int. Circ. Conf.*, Portland.

[30] Scott, A., Clayton, J., and Gibson, E. (1991), *A Practical Guide to Knowledge Acquisition*, Addison-Wesley.

[31] Sharkey, A. (1996), “On combining artificial neural networks,” *Connection Science*, vol. 8, no. 3/4, pp. 299-314.

[32] Sharkey, N. and Sharkey, A. (1994), “Understanding catastrophic interference in neural nets,” Technical Report CS-94-4, Dept. of CS, Univ. of Sheffield, UK.

[33] Sun, R. (1994), *Integrating Rules and Connectionism for Robust Commonsense Reasoning*, John Wiley and Sons.

[34] Taha, I. (1997), *A Hybrid Intelligent Architecture for Revising Domain Knowledge*, Ph.D. thesis, Dept. of ECE, Univ. of Texas at Austin, December.

[35] Taha, I. and Ghosh, J. (1997), “Hybrid intelligent architecture and its application to water reservoir control,” *International Journal of Smart Engineering Systems*, vol. 1, pp. 59-75.

[36] Taha, I. and Ghosh, J. (1999), "Symbolic interpretation of artificial neural networks," *IEEE Trans. Knowledge and Data Engineering*, vol. 11, no. 3, pp. 448-463.

[37] Takagi, H. and Hayashi, I. (1992), "NN-driven fuzzy reasoning," *Fuzzy Models for Pattern Recognition*, Bezdek, J. and Pal, S., editors, IEEE Press, pp. 496-512.

[38] Tazaki, E. and Inoue, N. (1994), "A generation methods for fuzzy rules using neural networks with planar lattice architecture," *IEEE International Conference on Neural Networks*, Orlando, FL, vol. III, pp. 1743-1748.

[39] Tickle, A.B., Andrews, R., Golea, M., and Diederich, J. (1998), "The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1057-1068.

[40] Towell, G. and Shavlik, J. (1993), "The extraction of refined rules from knowledge-based neural networks," *Machine Learning*, vol. 13, no. 1, pp. 71-101.

[41] Towell, G. and Shavlik, J. (1994), "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, no. 1-2, pp. 119-165.

[42] Towell, G., Shavlik, J., and Noordwier, M. (1990), "Refinement of approximate domain theories by knowledge-based artificial neural network," *Proceedings of Eighth National Conference on Artificial Intelligence*, pp. 861-866.

[43] Wang, L. and Mendel, J. (1992), "Generating fuzzy rules by learning examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427.

[44] Wilson, A. and Hendler, J. (1993), "Linking symbolic and subsymbolic computing," *Connection Science*, vol. 5, no. 3-4, pp. 395-414.