A Hybrid Intelligent Architecture and Its Application to Water Reservoir Control

Ismail Taha and Joydeep Ghosh *

Department of Electrical and Computer Engineering,

University of Texas, Austin, TX 78712-1084

E-mail: {ismail,ghosh}@pine.ece.utexas.edu

December 8, 1995

*This research was supported in part by ARO contracts DAAH04-94-G-0417 and 04-95-10494. Ismail Taha was also supported by the Egyptian Government Graduate Fellowship in Electrical and Computer Engineering.

Abstract

Measured inputs in control domains are often continuous. A discretization function is needed to map continuous inputs into multiple intervals or ranges of input values, so that they can be used as symbolic inputs to a rule-based system. The discretization parameters used to determine each interval play a critical role in the overall effectiveness of a rule-based system. This paper introduces a **Hybrid Intelligent Architecture (HIA)** that exploits the complementary features of expert systems and connectionist architectures to revise the initial domain knowledge and enhance its input characterization. HIA has three building blocks: a Knowledge-Based module, a Statistical module and a Connectionist Architecture module. A well defined format is used to describe the initial knowledge acquired from the application domain in a rule-based format and to enable its mapping into a uniform, three layer network. The statistical module updates both the rule-based and the connectionist subsystems by observing certain correlations among input-input and input-output pairs. The backpropagation algorithm is augmented to allow refining of input discretization functions and extracting new domain knowledge. A successful application of the proposed concepts to control the water reservoirs of Colorado river near Austin, is described. Moreover, the public domain breast cancer data set is used to show the importance of the statistical module in extracting rules when no prior domain knowledge is available.

1 Introduction

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 [17, 25]. Expert systems can also suffer from the brittleness of rules whenever rule antecedents must be exactly satisfied before the corresponding consequents become true [26]. Though substantial progress has been made over the past decade on alleviating these problems, the capabilities of symbolic representation and processing are still limited when domain theories are noisy. 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 [15]. However, they cannot as readily incorporate domain knowledge or provide a symbolic explanation of results. Such observations have motivated several studies of Hybrid Intelligent Systems that exploit the complementary features of expert systems and neural network paradigms [28, 16, 5, 8, 18, 12, 33].

In this paper, we augment a knowledge base system with a connectionist model and a statistical model to help the former refine its domain knowledge and improve its performance and robustness. The resulting **Hybrid Intelligent Architecture (HIA)** is shown in Figure 1. A key feature of this HIA is the ability to change the characterization of continuous valued inputs, based on training data. The rule based system represents the initial domain theory extracted from domain experts in a rule-based format. The extracted 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 the expert system. Therefore, it supplements the expert system with numerical computations [14].

Before training the initial connectionist architecture, a new 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 [23, 31, 11, 1]. 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 Hybrid Intelligent Architecture was implemented to control the water reservoirs of the Colorado river around Austin. Section 2 - 7 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. Section 8 describes the water reservoirs control application. In the concluding section we comment on the relation between the proposed approach and some other hybrid and fuzzy approaches, summarize the significance of the current work and point to future directions.

2 Representation of Initial Domain Knowledge

As depicted in Figure 2, the available domain theory 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 the proposed architecture is the knowledge based module. The knowledge based module 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 which can be acquired easily from the domain without consuming much time or effort. The basic components needed to build the DORs are:

1. The domain objects and their attributes.

2. The relationship between the domain objects.

3. 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 given below is a general rule based format that can be used to represent any rule-based system[1]:

[1]The DOR supports rule-based systems with and without certainty factors. In case of rule-based systems without certainty factors, w is "1" in each rule.

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

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

$Simple\text{-}Condition ::= Boolean\text{-}Expression \mid Negated\ Simple\text{-}Condition$

The $*$ means that the term $[OR\ Compound\text{-}Condition]$ can be repeated $n \geq 0$ times. The $+$ means that the rule must have at least one consequent. In real applications, rules are not always fully true. Therefore, each rule represented by the DOR format has an attached value, w, which indicates the measure of belief (or disbelief, if the value of w is negative), in the rule consequent provided the premise (left hand side) of the rule is 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 leads to a simpler uniform connectionist network, as seen later.
- The DOR format does not have any restriction on the number or the order of the rule operators.
- The DORs are not be converted to any other format before mapping them to the corresponding initial connectionist architecture irrespective of the number or the order of the operators in each rule. Another approach of representing domain operational rules is to convert the operational rules to their primitives to simplify the mapping operation[8]. This approach simplifies the mapping phase but it restricts the generated connectionist architecture from learning new concepts in the training phase. This approach requires the interruption of the training phase and the addition of hidden nodes and links, while our relaxed approach avoids this overhead.

3 Extraction of Supplementary Rules

In many application domains, extracting complete domain operational rules from domain representatives suffer from the knowledge acquisition bottleneck [25, 17]. Therefore, we need to seek another source of information to get more prior knowledge from the application domain. A very rich and always accessible source of information is the data sets that we can use as either training or validation data sets to the connectionist architecture.

A common approach of extracting information from the available data sets is to investigate the statistical nature of these data sets. We have investigated two different statistical approaches to extract prior domain knowledge from the available data sets. As shown in Figure 5, 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 [6, 13].

These two approaches can be done independently and their results can be combined with the extracted rule base. In the following subsection we present how supplementary correlation rules can be extracted from the available data sets and 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.

3.1 Extraction of Correlation Rules

The correlation approach of the statistical module analyzes the 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).

Assume that X represents an input feature vector and Y represents an output decision vector, let

$$CC1 = \frac{CO(X)}{\sqrt{\sigma}_X} \tag{1}$$

and

$$CC2 = \frac{CO(X,Y)}{\sqrt{\sigma_X \dot{\sigma}_Y}} \tag{2}$$

where CC1 in Equation 1 represents the correlation coefficients matrix between the elements of the input feature vector X, CC2 in Equation 2 represents the correlation coefficients matrix between the input and the output vectors X and Y, and the CO($\cdot$) function represents the covariance matrix. After computing the correlation coefficients matrices (CC1 and CC2), a threshold value, θ, is chosen based on the application domain (usually between 0.85 and 0.99). Based on the chosen threshold θ and the correlation coefficients matrices CC1 and CC2, 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. If the correlation coefficient between any pair of parameters (say X and Y) is greater than θ then create the COR:

   $$X \quad \xrightarrow{w} \quad Y \tag{R1}$$

   where the value of w represents the confidence level of the generated rule which equals the value of the correlation coefficient θ between the two parameters X and Y.

2. If the correlation between the two parameters X and Y is less than $-\theta$ then create the COR:

$$"NOT" \quad X \quad \xrightarrow{w} \quad Y \qquad \text{(R2)}$$

where the value of w equals to the magnitude of θ.

3. If the correlation coefficient between any two parameters is greater than $-\theta$ and less than $+\theta$, no COR 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 DOR format to be matched with the rule based module. Hence, they (the COR and the DOR) can be integrated together. The CORs generated by the statistical module can be used as a constraint or as supplementary rules to the DORs and hence helps in creating an initial connectionist architecture with more prior domain knowledge. As an example, assume that the knowledge based module extracts Rule (R3) and the statistical analysis module extracts Rule (R4) based on the correlation between X1 and X2. Therefore, Rule (R4) can be used to simplify the previous DOR Rule (R3) and a new Rule (R5) will replace both Rule (R3) and Rule (R4).

$$X1 \quad AND \quad X2 \quad \xrightarrow{w1} \quad Y \qquad \text{(R3)}$$

$$X1 \quad \xrightarrow{w2} \quad X2 \qquad \text{(R4)}^2$$

$$X1 \quad \xrightarrow{w3} \quad Y \qquad \text{(R5)}^3$$

[2] Note that w2 should be >= .85

[3] $w3 = w1 * w2$

Note that the new Rule (R5) does not depend on X2 which is highly correlated with X1 based on the COR extracted by the statistical module. Moreover, the logical interpretation of Rule (R5) results from combining the logical meaning of both Rules (R3) and Rule (R4) which can be interpreted as:

"If X1 is true Then X2 is true (with w2 confidence measure) AND If X1 AND X2 are both true then Y is true with a confidence level = w1". This interpretation can be simplified to: *"If X1 is true; which implicitly implies that X2 is true; then Y is true with confidence level* $= w1 * w2$*"* which represent the semantics of Rule (R5).

Another useful way of using the extracted CORs can be described by the next example: If the knowledge based module extracts Rule (R3) and the statistical module extract Rule (R6) then Rule (R3) can not be fired any more whatever the value of X1 (i.e., in either cases if X1 is true or false) because Rule (R6) is considered as a constraint rule to the DOR Rule (R3).

$$NOT \quad X1 \quad \xrightarrow{.96} \quad X2 \qquad \text{(R6)}$$

In other cases the statistical module can extract CORs which are not exist in the DORs. As an example, if the statistical module extracts Rule (R7)

$$X3 \quad \xrightarrow{.89} \quad Y \qquad \text{(R7)}$$

and there was no other rules in the DOR to represent any logical relationship between $X3$ and Y, then the generated COR (R7) is added to the extracted DORs.

Based on the previous cases, the statistical module provides the DOR with either a constraint or supplementary CORs. Moreover, if there was no DORs extracted from the application domain, the CORs will help creating DORs as an initial rule based.

After integrating the extracted rules by the knowledge base module (i.e., the DORs) and the extracted rules by the statistical module (i.e., the CORs) in the DOR format, the Nodes-Links Algorithm (NLA), described in Section 4 is used to map these integrated rules into an initial connectionist architectures.

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 techniques in multivariate analysis for this purpose [13, 6].

As a preprocessing step, the correlation matrices represented by CC1 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 the input to the constructed neural network.

The experimental results presented in Section 9 illustrate how we used the statistics of a public domain data set to extract more prior domain knowledge and to reduce the complexity of the initial connectionist architecture.

4 The Node Links Algorithm (NLA)

The NLA utilizes a set of mapping principles to map the initial domain theory, represented in the DOR 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. In this case, the NLA creates a hidden node with one link to a single input node. 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 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[10, 35, 5].

5 Connectionist Input Representations

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 the discretized intervals. In a basic symbolic system, exactly one of the x_is 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 [34].

A Gaussian function with mean μ_i and standard deviation σ_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. 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 $x_2 = 0.75$. 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.

An alternative approach to discretize such continuous input is to use the standard fuzzy logic approach[24, 4], as exemplified by Figure 6 where five fuzzy states are used to represent a variable T, temperature, within a continuous range between $T1$ and $T2$. Each state is represented by a membership function curve, usually triangular in shape, except for the two boundary states which are represented by trapezoidal-shapes. Any value of T is fuzzified using the five states of Figure 6 to specify the membership value of the measured input in each state, with at most two states having a value > 0.

There are two main differences between the standard fuzzy approach and our coarse coding approach. The first difference lies in the membership function used. Our approach uses a smooth and differentiable Gaussian function to represent each state of the measured input variable instead of a triangle-shape membership curve. The mean and the variance of each Gaussian curve in each state (i.e. interval) can be customized and adapted for improved performance. The second difference lies in the output representation. In the fuzzy logic approach a defuzzification phase is used at the end to provide the system user with a single value output. However, in our coarse coding approach there is no need to this phase because the coarse coding fuzzification process is used as a front end to the initial connectionist architecture which was mapped from the domain operational rules, as shown in Figure 7.

6 Refining Input Characterization

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

$$x_i = f_i(z) = e^{-\frac{1}{2}(\frac{z-\mu_i}{\sigma_i})^2} \tag{3}$$

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. Moreover, the ABA is used to refine the initial discretization parameters μ_i and σ_i for each interval i. The ABA calculates the stochastic gradient descents of the output error with respect to μ_i and σ_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 (μ_i, σ_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 μ_i and σ_i:

$$\frac{\partial E}{\partial \mu_i} = \frac{\partial E}{\partial f_i(z)} \cdot \frac{\partial f_i(z)}{\partial \mu_i} = \frac{1}{\sigma_i^2} \cdot (z - \mu_i) \cdot \sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}} \tag{4}$$

$$\frac{\partial E}{\partial \sigma_i} = \frac{\partial E}{\partial f_i(z)} \cdot \frac{\partial f_i(z)}{\partial \sigma_i} = \frac{(z - \mu_i)^2}{{\sigma_i}^3} \cdot \sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}} \tag{5}$$

where the term $\sum_{j=0}^{h-1} w_{ij} \cdot \frac{\partial E}{\partial w_{ij}}$ represents the gradient decent 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:

$$\mu_{inew} = \mu_{iold} - \eta \cdot \frac{\partial E}{\partial \mu_i} + MomentumTerm \tag{6}$$

$$\sigma_{inew} = \sigma_{iold} - \eta \cdot \frac{\partial E}{\partial \sigma_i} + MomentumTerm \tag{7}$$

7 Rule Extraction from Trained Network

The rule extraction module maps the trained connectionist architecture back into a rule based format. This mapping is much harder than the mapping from 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, (iii) new concepts may get extracted. An efficient rule extraction module should be able to deal with these three issues.

The key motivation for the rule extraction module is to improve the power of explanation through the updated expert system. This explanation power is required in many real applications. It helps the user to find out why a specific decision was taken by the system. A good summary of different extraction techniques can be found in [1].

Our rule extraction algorithm first searches for combinations of weights on small groups of afferent links for each hidden node, that exceed the threshold value of that hidden node. Then, it constructs a rule corresponding to each weight combination. Such an extracted rule is called a hidden rule and can be interpreted as: If the premise (combination of a set of anded input nodes) of the rule is true then this hidden node is active. Similarly, it extracts output rules by examining weight combinations from hidden to output nodes. The final rules are formed by substituting each hidden node in an output rule (i.e. in the premise of this output rule) by the corresponding hidden rule premise of this hidden node. To limit extracted rules to efficient ones, only weight values of afferent links to a hidden or an output node which exceeds the average weight value of afferent links to that node, are considered. Another heuristic which can be used to cut down the complexity of extracted rules is to limit the maximum number of conditions in rule premises[7]. The main difference between the two approaches is that the latter is domain dependent.

8 Controlling Water Reservoirs

There are several dams and lakes on the Colorado river near Austin, as shown in Figure 8. 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, (see Figure 8), the rules of the flood control operating mode are used to determine water release.

We acquired the 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]. Figure 9 shows the regulation rules for controlling Mansfield Dam[4]. For example, the upper rule for Mansfield dam (see Figure 9), expressed in DOR format is:

If Projected-Level(Lake-Travis,t0) >= 710[5] 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.

8.1 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

[4] Reprinted from the LCRA documentation of the 1991 Christmas flood.

[5] All values are measured in feet above mean sea level

(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 μ_i and standard deviation σ_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 4 through 7, to refine the initial discretization parameters (μ_i and σ_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 mean (μ) of seven different intervals were shifted and the standard deviation (σ) 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). The

Table 1: Test results of Colorado River Problem

No. of epochs	HIA with refined DOR		MLP initialized randomly		MLP with initial DOR	
	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

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[6] provides a summary of the testing results. 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.

[6]The presented results are the average of 10 runs. They represent the performance of each architecture when the test data sets were applied.

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.

8.2 Rule Extraction

After the learning phase, the connectionist module of HIA was examined by the rule extraction module 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.

9 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 data set is available[22, 2]. This data set has nine inputs and two output classes. Each instance belongs to one of two possible classes: benign or malignant. The 683 available instances were randomly divided into a training data set of size 341 and a testing set of size 342. For this problem three different architectures were designed and implemented:

1. **The first architecture:** A randomly initialized MLP with one hidden layer. Hidden nodes were represented by sigmoid function. The backpropagation algorithm with momentum term was used to train the constructed network.

2. **The second architecture (HIA):** This architecture was generated as follows:

   - The statistical module was used to calculate the correlation matrix between input-input and input-output parameters.
   - Based on the correlation matrices calculated by the previous step, a set of CORs were generated.
   - The Nodes Links Algorithm was applied to map the extracted CORs into an initial neural network architecture. Note that the number of the hidden units in this architecture were obtained based on the mapping procedure of NLA.
   - The generated connectionist architecture was initialized using the NLA.
   - The resulting architectures were trained by the backpropagation algorithm with momentum term.

3. **The third architecture:** This architecture was generated as follows:

   - The principal Component Analysis Algorithm was followed to get the covariance matrix between the nine input parameters of the problem.
   - Based on the results of the previous step, a randomly initialized MLP architecture was constructed for each problem. Note that the number of hidden units for each problem were selected after several experiments to give the best results.
   - The resulting architectures were trained by the backpropagation algorithm with momentum term.

Table 2 compares the performance of the three architectures on the test data. The presented results are the average of 10 runs. The benefits of using the prior domain knowledge extracted by the statistical module are quite evident.

Table 2: Test results for the Breast Cancer data set.

Arch.	No. of nodes		Training time	Test results	
	input	hidden	in epochs	MSE	Class. rate
MLP	9	9	1000	0.119	93.27
HIA	8	9	300	0.059	96.49
PCA-MLP	6	9	1000	0.079	95.95

10 Discussion

10.1 Relation to other approaches

Several researchers have investigated the design of hybrid systems that combine expert and connectionist subsystems[27, 33, 8, 16, 12]. The typical result is a Knowledge Based Neural Network (KBNN) system with four phases:

- 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 a 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) [32] 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 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[21]. 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 example of hybrid systems 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[3], or to refine and extract fuzzy rules[30, 29, 20]. 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 non trivial. 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 Hybrid Intelligent Architecture (HIA) described in this paper is 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 reduces the overall learning time and enhances system performance; (4) it combines a statistical module along with its knowledge based and neural network modules to extract prior domain knowledge when such knowledge is not available; (5) HIA is a complete hybrid model.

10.2 Concluding Remarks and Future Work

HIA was successfully applied to the problem of controlling the flood gates of Colorado river in greater Austin. The obtained results show that refining the input characterization along with the domain knowledge incorporated in a connectionist model can substantially enhance the generalization ability of that model. Moreover, it shows the capability of HIA to extract new and useful operational rules from the trained connectionist module.

A classification problem was also presented to exemplify 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.

The current method of rule extraction from a trained neural network can be further refined to make it more efficient. The objective of extracting rules from a trained neural network can be extended beyond the issue of providing an explanation capability to the trained neural network. It can be used to examine the correctness of the trained network. Moreover, it can be used to maintain the truth represented in the trained network.

The last module in HIA is the integrated decision maker. Currently, this module simply provides the user with both expert system and neural network outputs. We are exploring methods of integrating the two sources to obtain more accurate and reliable outputs with associated confidences. We are also implementing a multi-sensor fusion platform using HIA concepts, as part of a robust automatic target recognition system. Such real-world applications are needed to further substantiate the usefulness of hybrid systems.

Acknowledgment:

We are thankful to Prof. B. Kuipers for bringing the Colorado river problem to our attention.

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, Neurocomputing Research Center, Queensland University of Technology, Queensland, Australia, 1995.

[2] K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. In *Optimization Methods and Software 1*. Gordon and Breach Science Publishers, 1992.

[3] 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.

[4] E. Cox. Fuzzy fundamentals. *IEEE Spectrum*, pages 58–61, October 1992.

[5] J. Fletcher and Z. Obradovic. Combining prior symbolic knowledge and constructive neural network learning. *Connection Science*, 5(3-4):365–375, 1993.

[6] J.H. Friedman. An overview of predictive learning and function approximation. In V. Cherkassky, J.H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, pages 1–16. Springer Verlag, 1994.

[7] L.M. Fu. Rule learning by searching on adapted nets. In *Proceedings of the Ninth National Conference on Artificial Intelligence (Anaheim CA)*, pages 590–595, 1991.

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

[9] L.M. Fu. *Neural Networks in Computer Intelligence.* Mc Graw-Hill, Inc., 1994.

[10] S.I. Gallant. Connectionist expert systems. *Communications of the ACM*, 31(2):152–169, 1988.

[11] C.L Giles and C.W. Omlin. Pruning recurrent neural networks for improved generalization performance. *IEEE Transactions on Neural Networks*, 5(5):848–851, 1994.

[12] 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.

[13] R.C. Gonzalez. *Digital Image Processing.* Addison Wesley, 1993.

[14] R.M. Goodman, C.M. Higgins, and J.W. Miller. Rule-based neural networks for classification and probability estimation. *Neural Computation*, 4:781–804, 1992.

[15] S. Haykin. *Neural Networks A Comprehensive Foundation.* Macmillan, 1994.

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

[17] P. Jackson. *Introduction to Expert Systems.* Addison Wesley, 1990.

[18] R.C. Lacher, S.I. Hruska, and D.C. Kuncicky. Backpropagation learning in expert networks. *IEEE Transactions on Neural Networks*, 3(1):62–72, 1992.

[19] LCRA. 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, 1992.

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

[21] 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.

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

[23] P.M. Murphy and M.J. Pazzani. Id2-of-3: Constructive induction of n-of-m concepts for discriminators in decision trees. In *Proceedings of Eighth International Machine Learning*, pages 183–187, 1991.

[24] D.G. Schwartz and G.J. Klir. Fuzzy logic flowers in Japan. *IEEE Spectrum*, pages 32–35, July 1992.

[25] A.C. Scott, J.C. Clayton, and E.L. Gibson. *A Practical Guide To Knowledge Acquisition.* Addison Wesley, 1991.

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

[27] I. Taha and J. Ghosh. Controlling water reservoirs using a hybrid intelligent architecture. In *Proceedings of the Artificial Neural Networks In Engineering*, pages 63–86, November 1995.

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] L. Wang and J.M. Mendel. Generating fuzzy rules by learning examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, 1992.

[35] A. Wilson and J. Hendler. Linking symbolic and subsymbolic computing. *Connection Science*, 5(3-4):395–414, 1993.

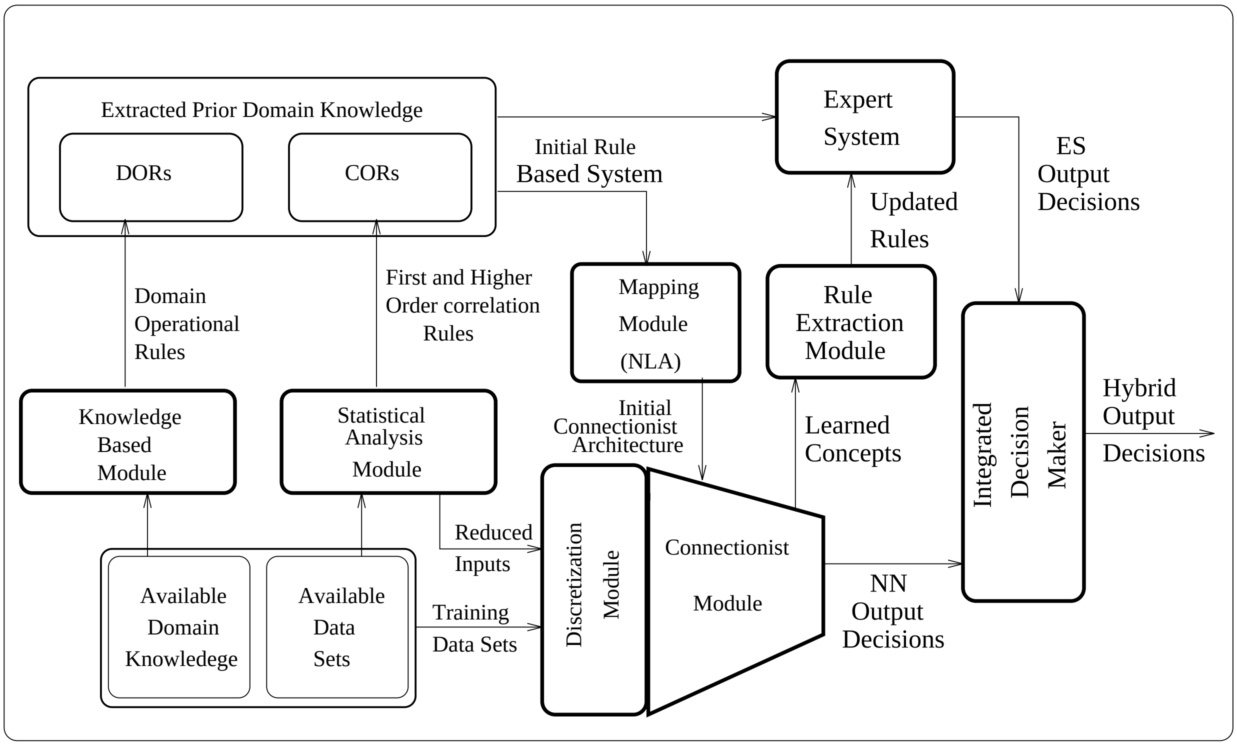

Figure 1: A hybrid intelligent architecture that integrates knowledge based, statistical and connectionist paradigms.

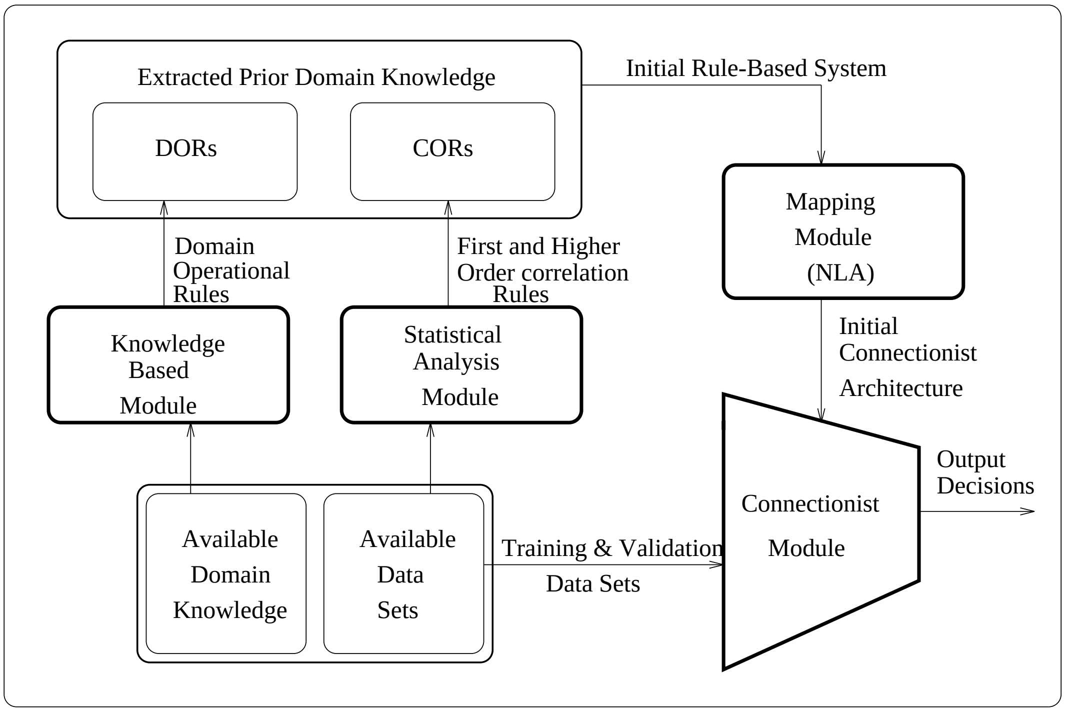

Figure 2: **The main modules of the proposed architecture**

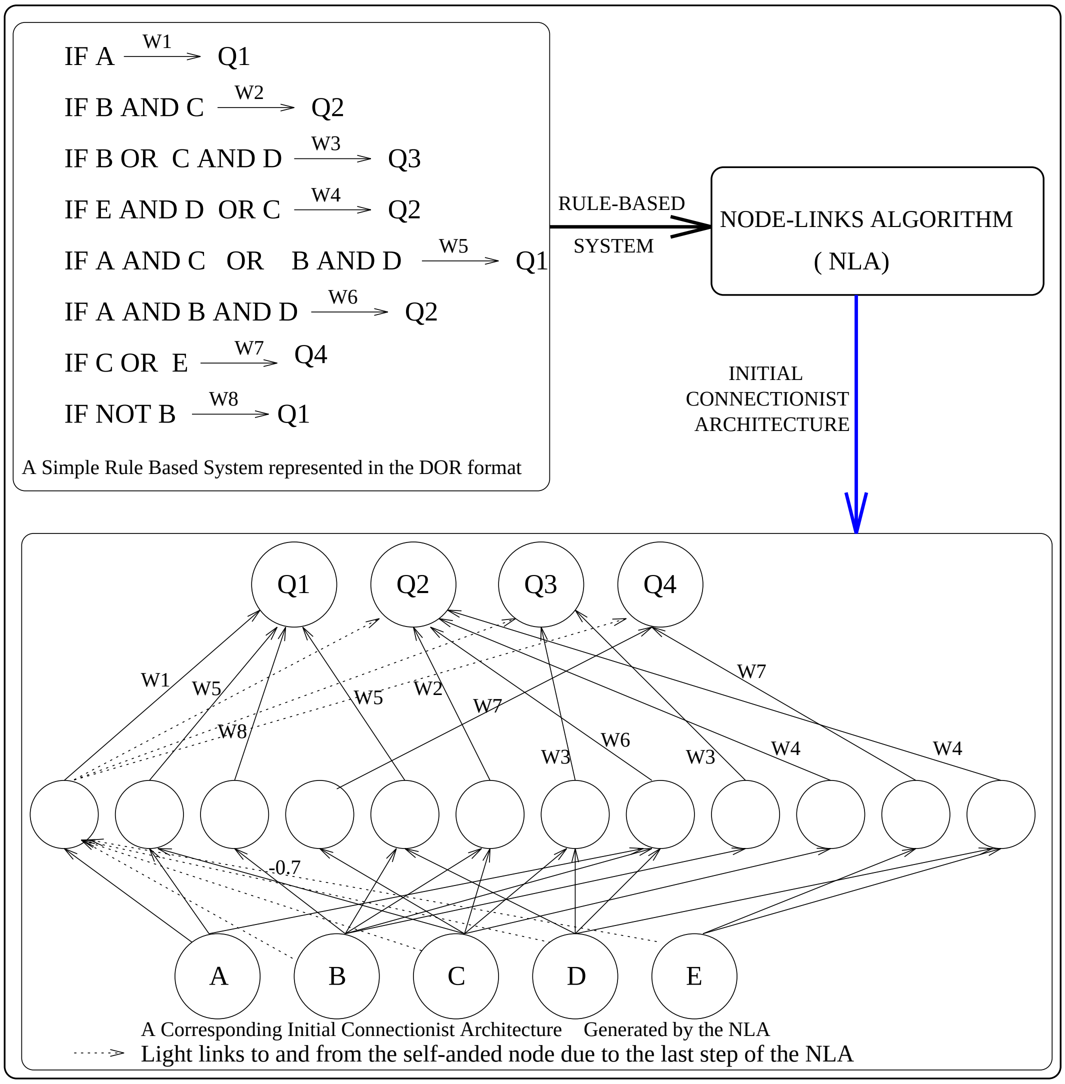

Figure 3: From rule-based system to initial connectionist architecture using the Node-Links Algorithm.

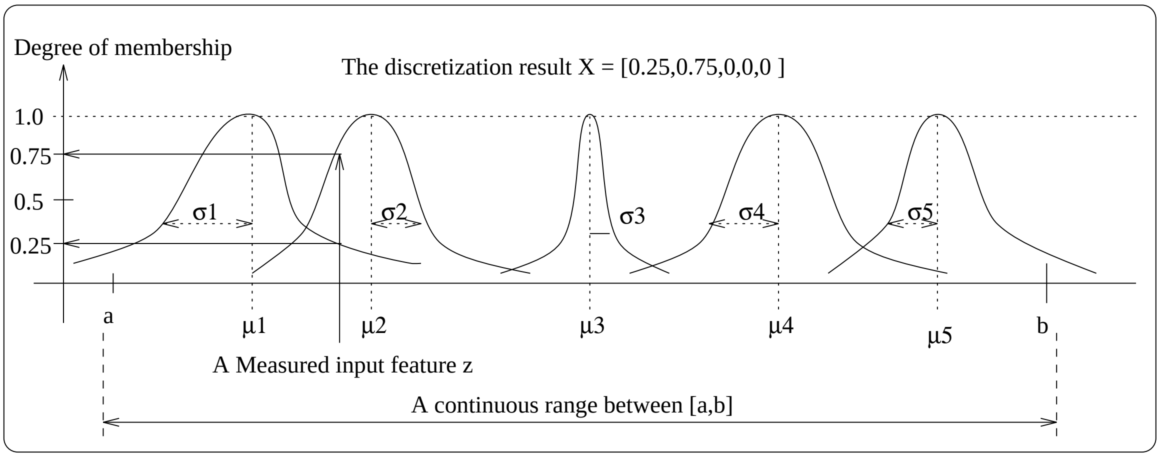

Figure 4: Discretizing a continuous measured input into n intervals using nGaussian functions.

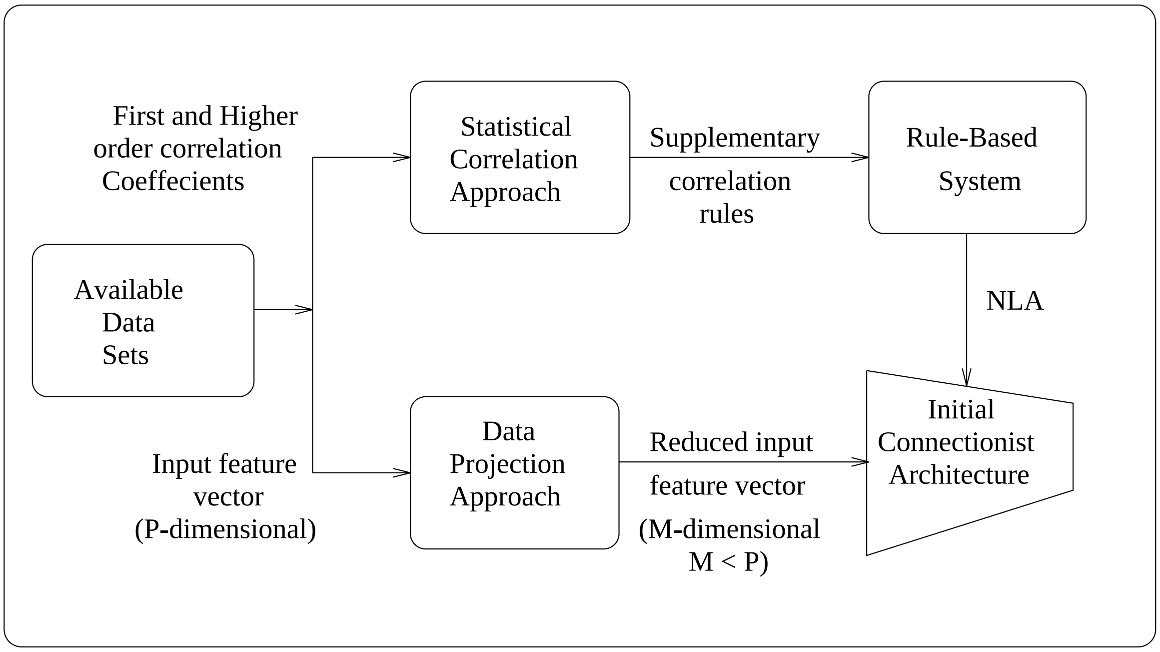

Figure 5: **Two statistical approaches to extract additional domain knowledge**

temperature

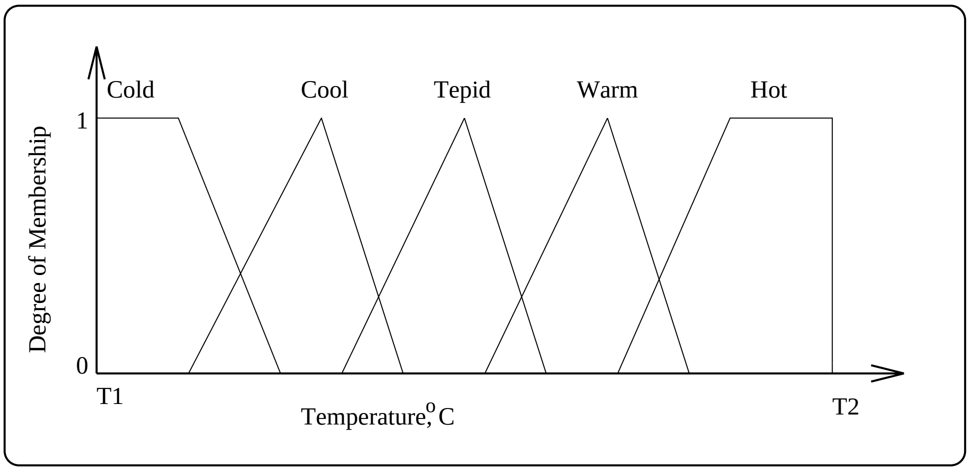

Figure 6: **A typical fuzzy system with five states to fuzzify an input variable T**

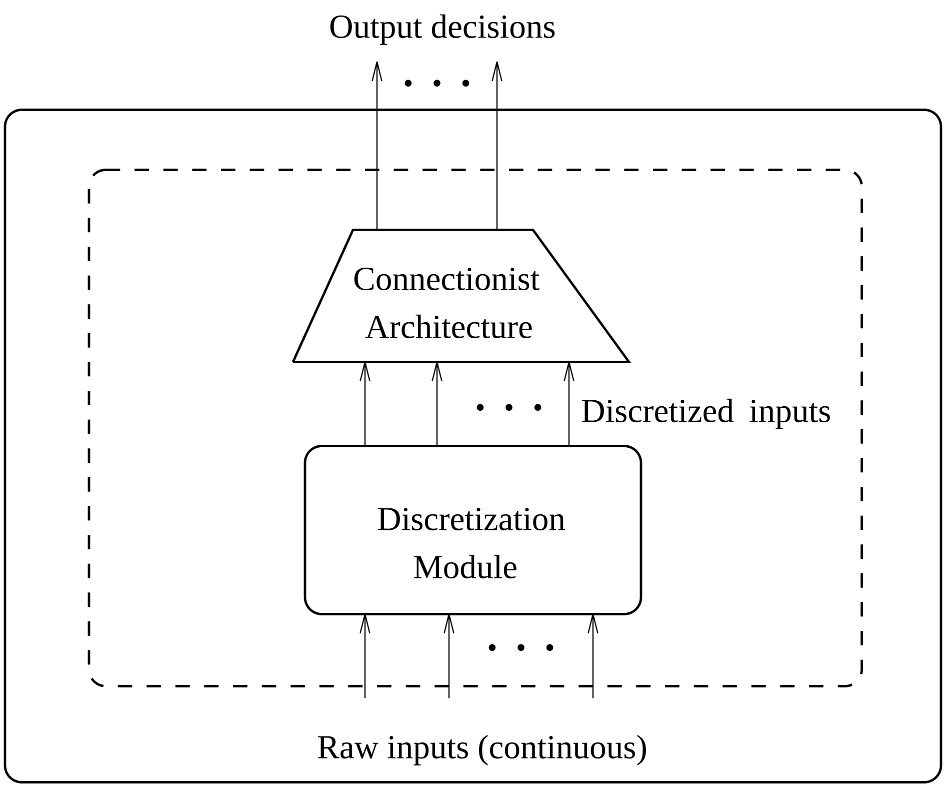

Figure 7: **Using the discretization module as a front end to the connectionist architecture**

Figure 8: How the LCRA managed the historic flood of December 1991. Printed from the LCRA Lake Travis Townhall meetings 1992

Figure 9: Regulation rules for controlling Mansfield Dam. Printed from the LCRA documentation of the 1991 Christmas flood.