RULE GROUPINGS IN EXPERT SYSTEMS USING NEAREST NEIGHBOUR DECISION RULES, AND CONVEX HULLS.

Stergios Anastasiadis

McGill University, School Of Computer Science,
3480 University Avenue,
Montreal, Quebec, Canada, H3A 2A7.
stergios@bnr.ca
calserg@homer.cs.mcgill.ca

Abstract. Expert System shells lack in many areas of software engineering. Large rule-based systems are not semantically comprehensible, difficult to debug, and impossible to modify or validate. Partitioning a set of rules found CLIPS, into groups of rules which reflect the underlying semantic subdomains of the problem, will address adequately the concerns stated above. In this paper we introduce techniques to structure a CLIPS rule-base into groups of rules that inherently have common semantic information. The concepts involved are imported from the fields of A.I., Pattern Recognition, and Statistical Inference. Techniques focus on the areas of feature selection, classification, and a criteria of how "good" the classification technique is, based on Bayesian Decision Theory. We discuss a variety of distance metrics for measuring the "closeness" of CLIPS rules and describe various Nearest Neighbor classification algorithms based on the above metrics.

INTRODUCTION

Knowledge is a collection of related facts that can be used in inference systems such as CLIPS. A production rule is a method for representing knowledge, among others that exist in the world of knowledge representation, such as inclusion hierarchies, mathematical logic, frames, scripts, semantic networks, constraints, and relational databases[10]. In CLIPS this method of representation is particularly appropriate since knowledge is "action-oriented". Although production rules lend themselves to be used in inference systems, validation and verification of syntactic and semantic correctness is difficult to achieve.

Rule-based systems have been developed with a very narrow scope. They don't allow interaction with other types of knowledge and mechanisms used to derive the required results are driven by pure syntactical procedures. Therefore, to move away from isolation, rules must not embody all of the knowledge in a system and furthermore, rules must encapsulate semantic information that would make the derivation process more sophisticated and efficient.

Production systems are not developed one rule at a time, but the complexity and success of the system relies on the inter-dependencies and interactions between rules[7]. Small systems might contain a couple of hundred rules, resulting in a total of many hundreds and/or thousands of patterns that must be matched in order to logically proceed to another derivation. This results in exponential growth of pattern matching without taking advantage of the dependencies and interactions that exist between rules ; which dependencies and interactions were the reasons why the system was established to begin with.

CLIPS does provide saliences for rules as a semantic feature of the rule-base. This alone does not help in removing redundancies or improving the validity of the system. Grouping the rule-base into groups of related rules will extract the underlying domain knowledge from the rule-base. Attaching then priorities to each group, and to each rule in a group will further help the system in its derivation process, in its ability to distinguish between various types of knowledge [2,3], and in its ability to exchange and communicate with other types of knowledge in order to validate, verify, and explain its conclusions [1].

In this paper we will present a mechanism to measure distances between CLIPS rules, and using those results we will present classification schemes that will insert rules into the appropriate group.Then we will also show how accurate the classification schemes are by providing a statistical estimation of misclassification. This estimation will help when one has already classified the rules into groups, and requires to insert a new rule into the "best" group. Finally, we will make some concluding remarks and some possible extensions to the scheme.

OVERVIEW OF METHOD

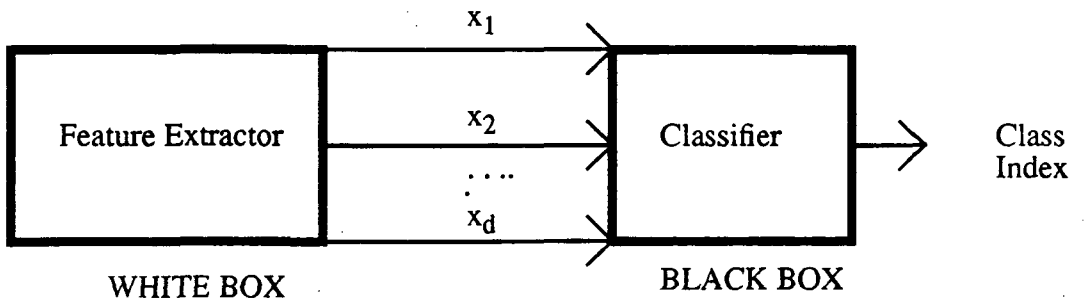
The general approach is to select the crucial information from a rule, and classify it in the appropriate group. Therefore, feature selection, extraction, and classification is the logical route followed by this scheme. One can either allow the system to generate the groups on its own, or the user can choose a rule that represents a group, and then the system classifies the other rules accordingly.

The important characteristics that are going to be used from a rule are inserted in a feature vector X. The required parsing of the rule is done in order to fill in the vector. Information that is selected from a rule are patterns from the antecedent, consequent, or from both. Hence, each rule has its own feature vector,

$$X_i = (x_1, x_2, \ldots, x_d) ,$$

where d is the feature vector size, and i is the rule number. The feature vector size has a value that is set before run-time, and depends on how complex the rules are. The more patterns that exist, the higher value d will obtain.

The general flow is as follows :



The "WHITE BOX" of the feature extractor is application dependant and produces the feature vector for a given rule. The "BLACK BOX" of the classifier is a universal method comprised of a discriminant function and a maximum selector that returns some class index indicating which group the rule has been put in. The "BLACK BOX" classifier is shown in Figure 1.

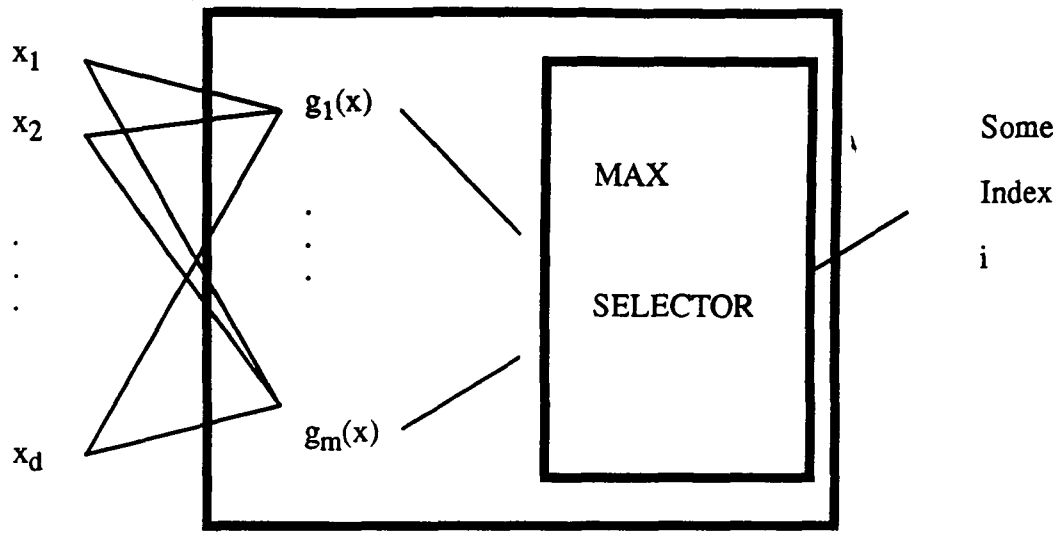


Figure 1 : "BLACK BOX" Classifier.

The discriminant function indicates the confidence that a feature vector comes from a particular class or group of rules. In Figure 1 the function $g_i(x)$ is the discriminant function, and m is the number of classes or group of rules that exist in the system. The selector is used as a "voting" mechanism to finally determine where the rule belongs.

Graphically, decisions are represented as regions, and the boundaries between regions are determined by the difference among the discriminant functions that represent each region. For example given two feature vectors X_1, X_2, and regions R_1, R_2, then the boundaries are found through the equation :

$$g_1(X) - g_2(X) = 0$$

The regions might look like what is shown in Figure 2.

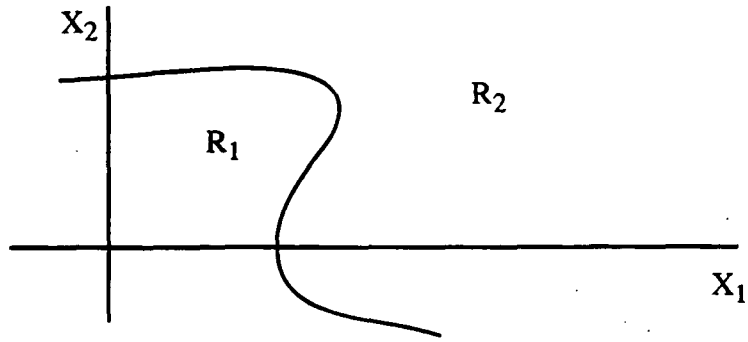


Figure 2 : Boundaries between regions.

DISTANCE METRICS

Different metrics capture different information from the rule-base of an expert system, and each rule-base lends itself to a particular distance metric. In CLIPS a rule-base is made up of rules with antecedents and consequents. Each antecedent or consequent is composed of patterns which match facts in working memory during run time. Each pattern is further divided into tokens, which in CLIPS can be a word, string or number.

As part of our formulation reserved words are ignored. All strings and variables are suppressed. Strings convey little domain knowledge, and variables have values only during run time, are local to a rule, and hence cannot carry along information when we are statically relating rules.

Here are the four distance metrics that will be used in our classification scheme :

$$\mathrm{Dall}(\mathrm{Rule}(1,2)) = \frac{\mathrm{NumOfTokensInRule1} + \mathrm{NumOfTokensInRule2}}{\mathrm{NumOfCommonTokensFromRule1andRule2}}$$

$$\mathrm{Dcon}(\mathrm{Rule}(1,2)) = \frac{\mathrm{NumOfTokensInConOfRule1} + \mathrm{NumOfTokensInConOfRule2}}{\mathrm{NumOfCommonTokensInConFromRule1andRule2}}$$

$$\mathrm{Dant}(\mathrm{Rule}(1,2)) = \frac{\mathrm{NumOfTokensInAntOfRule1} + \mathrm{NumOfTokensInAntOfRule2}}{\mathrm{NumOfCommonTokensInAntFromRule1andRule2}}$$

$$\mathrm{Dca}(\mathrm{Rule}(1,2)) = \frac{\mathrm{NumOfTokensInConOfRule1} + \mathrm{NumOfTokensInAntOfRule2}}{\mathrm{NumOfCommonTokensInConFromRule1andAntFromRule2}}$$

If any of the denominators of the above metrics is zero, then the value will be infinity. This is the case because we can see that the commonality between two rules is inversely proportional to the distance between them. Hence, a small distance indicates that the two rules have many patterns in common.

The Dca metric is used when the fundamental relationships between the rules of the system drive the derivation process from the consequent of one rule to the antecedents of other rules (refer to example 1). If in a rule-based system a large amount of knowledge is present in the antecedent of the rules, then the Dant metric would be appropriate. Similarly for the consequent part, then Dcon would be the metric of interest. Finally, it might be the case, as in diagnostic systems, that rule-interdependency does not allow one to use one of the above three metrics. Both the antecedent and the consequent of each rule play a different role each time a new derivation is to be made. So, since we want to take both sides into consideration, the Dall metric will be used (refer to example 2).

The following examples illustrate how the above metrics are used, and how one makes the decision on using a particular metric :

EXAMPLE 1 :

(defrule rule1 (a) => (assert (b))
(defrule rule2 (b) (c) => (assert (d))
(defrule rule3 (d) (e) => (assert (f))

Dca(1,2) = 3/1= 3; Dant(1,2) = Dcon(1,2) = INFINITY; Dall(1,2) = 5/1 = 5.
Dca(1,3) = Dant(1,3) = Dcon(1,3) = Dall(1,3) = INFINITY.
Dca(2,3) = 3/1 = 3; Dant(2,3) = Dcon(2,3) = INFINITY; Dall(2,3) = 6/1 = 6.
Therefore, Dca would be the appropriate metric to use.

EXAMPLE 2 :

(defrule rule1 (a) (b) => (assert (c)) (assert (d)))
(defrule rule2 (a) (e) => (assert (f)) (assert (g)))
(defrule rule3 (c) (d) (h) => (assert (i)))

Dca(1,2) = Dcon(1,2) = INFINITY; Dant(1,2) = 4/1 = 4; Dall(1,2) = 8/1 = 8.
Dca(1,3) = 5/2; Dcon(1,3) = Dant(1,3) = INFINITY; Dall(1,3) = 8/2 = 4;
Dca(2,3) = Dcon(2,3) = Dant(2,3) = Dall(2,3) = INFINITY;
The metric Dcon is ruled out, but Dca and Dant have unstable behaviors. Therefore, Dall would be selected.

CLASSIFICATION

The approaches that will be presented here are algorithmic and graphical. Three classification schemes are introduced, and an attempt will be made to analyze these schemes through some graphical representation. Then the Nearest Neighbor rule will be introduced, along with how the three schemes fit in to the decision making.

The basic assumption here is that initially each rule belongs in its own group, although the user has the ability to create initially his own groups with a representative rule from the rule-base being the initial member of each group. Lets begin with some definitions.

Let $\{X,C\} = \{X_1, C_1; ...; X_N, C_N\}$ be the set of N pattern samples available, where N is the number of rules in the CLIPS rule-base, and X_i and C_i denote, respectively, the feature vector and the label for the classification information of the ith pattern sample. Values for C_i are in the range [1,m], where m is the number of groups present in the rule-base as defined above. It's obvious with the above assumption that $1 <= m <= N$.

In the following schemes training and testing a set of data samples is frequently used. Training a data set means to use a decision rule (and in our case it will be the Nearest Neighbor rule) that will insert each of the feature vectors into the appropriate group. Given this initial classification one takes another set of data, that is possibly mutually exclusive from the training set, and examines if the feature vectors in the set for testing could be classified correctly in the classification scheme derived during the training session.

Here are the three methods for classification :

METHOD 1 : (Resubstitution - R method).[5]

The steps for classification are :

1) The classifier is trained on {X,C}.

2) The classifier is tested on {X,C}.

METHOD 2 : (Holdout - H method).[4]

The steps are :

1) Partition {X,C} into K randomly chosen pairs of sets of equal size

$$\{X,C\}^1_a : \{X,C\}^1_b , \ldots , \{X,C\}^K_a : \{X,C\}^K_b$$

such that for i = 1,...,K , $\{X,C\}^i_a$ and $\{X,C\}^i_b$ are mutually exclusive.

2) For i = 1,...,K , train the classifier on $\{X,C\}^i_a$ and test it on $\{X,C\}^i_b$.

METHOD 3 : (Leave-me-out - U method).[8]

The steps are :

1) Take one pattern sample $\{X_i, C_i\}$ out of {X, C} to create $\{X, C\}_i$.

2) Train the classifier on $\{X, C\}_i$.

3) Test the classifier on $\{X_i, C_i\}$. If X_i is classified into the group associated with C_i, then set $e_i = 0$; otherwise $e_i = 1$, where e_i acts as an error indicator.

4) Do steps 1)-3) for i = 1, ... , N to obtain values for all e_i's.

It turns out that the R-method is very biased as one will see when calculating the probability of misclassification. For the H-method traditionally 50 percent of the available samples have been used for training, and the other 50 percent for testing, and in the U-method when N is particularly large the method works well, but when N is relatively small it is not very reliable. Furthermore, it is an overly optimistic estimate of performance. Although lots of computations are required, the method tends to be unbiased and uses all the available data with high variance in discrete cases.

Apply the Nearest Neighbor rule with one of these methods (see next section on Nearest Neighbor rule), and fill in the values in {X, C}.

Now how can one graphically take this information and check for correctness? This can be done through what is called a convex hull. Let R denote the set of real numbers. Then R^2 denotes the set of all points in the plane. Define a set of points D from R^2 as being a **convex domain** if, for any two points x and y in D, the line segment xy is entirely contained in D. So if S is a set of points from R^2, then the **convex hull** of S is the boundary of the smallest convex domain in R^2 containing S [9]. See Figures 3,4 for an example of a convex domain and a convex hull, respectively.

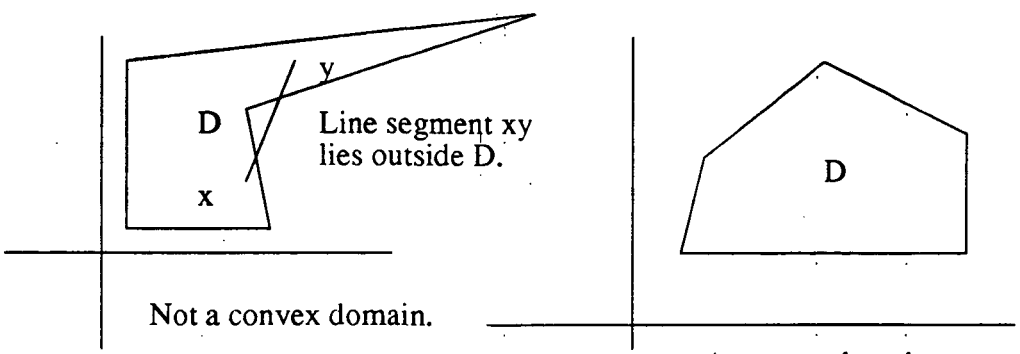


Figure 3 : Convex Domain.

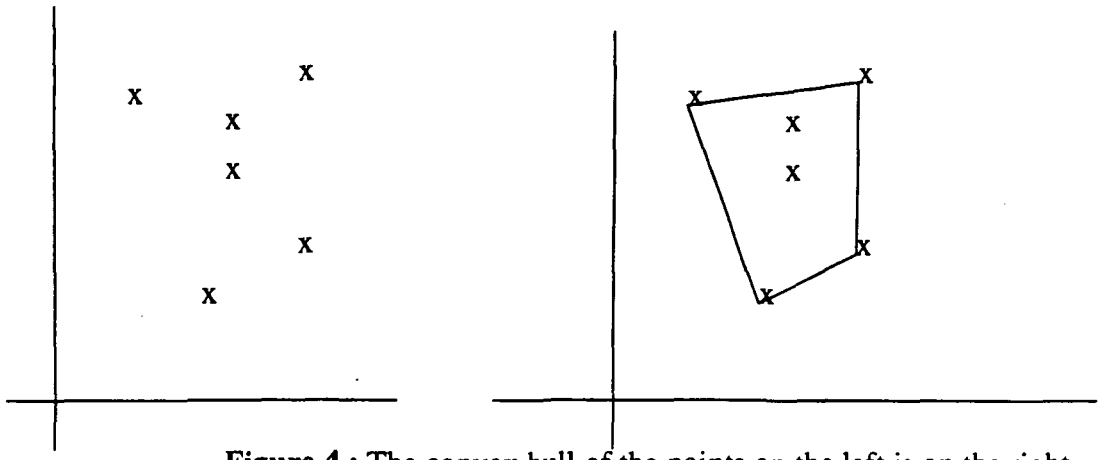
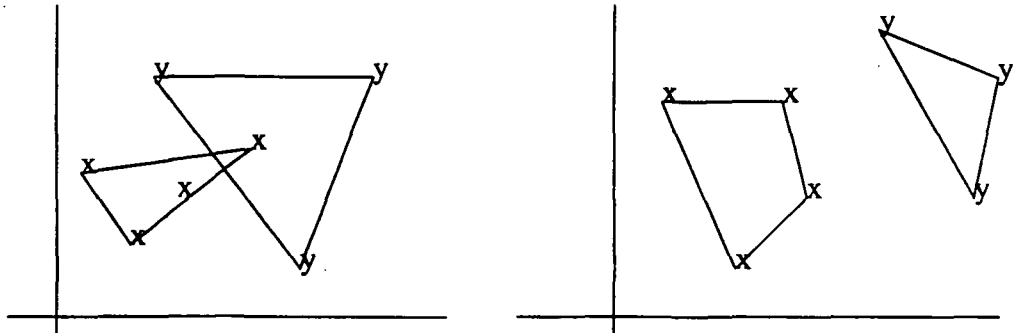


Figure 4 : The convex hull of the points on the left is on the right.

One can transform the set of points in {X, C} using a function F, the details of which are beyond the scope of this paper. So applying F to each element of {X, C} one has :
$F(X_i, C_i) = (i, k)$ meaning that rule i belongs to group k, for every i belonging to [1,N].Hence, we have pairs of numbers that can be graphed in R^2 just like in Figure 4, but the difference here is that one does not construct the convex hull of the entire set, but one builds the convex hulls of each group individually. So, if after classification one finds that the N original rules fall in r (with $1 <= r <= m <= N$) distinct groups, then it must be the case that the r convex hulls that are produced are all **linearly separable** between them (refer to Figure 5 for an example).

Two sets of points are **linearly separable** if and only if there exists a hyperplane H that separates them [9]. Therefore, two sets are separable if there convex hulls are disjoint, and our classification is a success only if all the convex hulls that correspond to the groups that have been derived are disjoint between them.



Group X and Y are not classified properly. Failure.

Group X and Y are classified properly. They have disjoint convex hulls.

Figure 5 : Two separable sets on the right and two non-separable sets on the left.

NEAREST NEIGHBOR RULE

The basic idea behind this concept is given a rule "A" find its distance with all the other rules in the training set. From those distances one chooses the smallest, and classify rule A as being in the same group as the rule for which they have the smallest distance.

Let G = the number of samples in the training set. Therefore, there are N -G samples in the testing data set.The general algorithm general algorithm is shown below :

Assign K = G
CLASSIFY the training set
FOR every rule i in the testing set
    CALCULATE the distance of rule i with all other rules in the training set
    FIND the K smallest distances between rule i and every rule in the training set
    FOR every K-pair of rules (i,j) if rule j belongs to group A, then group A gets a vote
    FIND the group that has the most votes, say A". Then rule i is classified in A".

Many of these steps can be done together, but it's written out explicitly in order for one to understand the algorithm . As one can see by varying K, one can get any of the above methods. Particularly one may also group the rules into a testing set and a training set discretely, choosing possibly mutually exclusive sets. Finally, as mentioned earlier, by classifying a training set one simply results to the default, where each rule belongs in its own group, or the expert selects rules that might be pivotal to a derivation as being the head of a group.

In the experiments that will be discussed the U-method was used. If K = 1, then it's the pure method, otherwise by varying K one is able to classify even with a more unbiased attitude. The ideal value for K, as mentioned in many papers and proven empirically, is :

$$\sqrt{N}$$

ESTIMATION OF MISCLASSIFICATION

For each of the above methods a probability of misclassification will be derived. There exist many probabilities of error that are used in ones analysis. Here we will consider two, and the second one will be chosen for our derivations.

OPTIMAL/BAYES PROBABILTY OF ERROR : It is denoted by P^B_e and is given by :

$$1 - \oint \text{MAXi} \{ P(\frac{X}{Ci}) P(Ci) \} dX$$

$P(X/C_i)$ and $P(C_i)$ are the class conditional probability density function and a priori probability of the ith group, respectively. The maximum value of the product is taken over all i, ie. over all possible groups of rules in the system. This error probability results when one has complete knowledge of the probability density functions with which to construct the optimal decision rule and uses the Bayes decision rule [6].

PROBABILITY OF ERROR : It is denoted by P_e and is the probability of error on future performance when the classifier is trained on the given data set. In practice one usually obtains a data set which is not only finite, but in fact quite small. Frequently no knowledge is available concerning the underlying distributions. In such cases one would like to know what the resulting probability of error is going to be on future pattern samples when the classifier is trained on the given data set. All the above methods train and then test the classification. Using this probability one will see how "good" the original classification was.

Let $P_e[R]$ denote the probability of error for the R-method using the Nearest Neighbor rule. If K denotes the number of incorrect classification attempts in step 2) of the process, then the resulting probability is K / N, where N is the number of rules in the system.

Let $P_e[H]$ denote the probability of error for the H-method using the Nearest Neighbor rule. At step 2) of the experiment let $P_e[H]_i$ be the number of errors found when attempting to test $\{X, C\}^i_b$ on the trained classifier $\{X, C\}^i_a$. With i varying from 1, ..., K , then the resulting probability of error is :

$$(1 / K) (P_e[H]_1 + ... + P_e[H]_K).$$

Let $P_e[U]$ denote the probability of error for the U-method using the Nearest Neighbor rule. During the entire process one has been tabulating the errors of misclassification when testing a sample pattern on the trained set (e_i). Therefore, the resulting probability of error is :

$$(1 / N) (P_e[U]_1 + ... + P_e[U]_N).$$

In spite of its advantages with regards to bias, the U-method suffers from at least two disadvantages. Although it is desirable to have the average of errors "close" to the actual error of probability, it is more important to use a method with a small variance. Having confidence about a particular classification of a data set might be preferred to being just unbiased. It has been shown that the U-method has much greater variance than the R-method, making it even more likely that it will be used less, compared to the simplicity of implementation of the R-method. Finally, the U-method requires excessive computation in the form of N training sessions, unless N is small.

EXPERIMENTAL RESULTS

Table 1: U-Method

K	ERRORS
6	2
12	5
24	13
36	24

The analysis of the automobile rule-base (auto.clp) There are 36 rules. The Dcon metric for obvious reasons is used. From the rules, 3 are selected as group leaders. One rule summarizes diagnostics, the second rule captures data collection and the other possible solutions. One uses the Nearest Neighbor rule with the U-method. Various values for K are considered. Observe that the square root of N (ie. 36) is the best solution.

Errors indicate how many misclassifications were attempted. Simialr results are derived with other methods.

CONCLUSIONS

From the information presented it is evident that there is more information encapsulated in the system. Underlying subdomains are created with more semantics attached to them in terms of classification, validation of the semantics of the rule-base is achievable through convex hulls, the rule-base is now more semantically comprehensible, and problems are simpler to debug.

Initially one must select the appropriate distance metric, choose a suitable method of classification depending on the characteristics of the system, and apply the Nearest Neighbor rule to the data. At the same time errors at run time are collected in order to come up with a probability of misclassification.

As an extension to the scheme one might add saliences to each group derived, and to further enhance the system one might add saliences to each rule in a group. Another addition might include for the system to select on its own the best metric after examining the probabilities of error along with the data. It is obvious that this analysis can be executed when entering the CLIPS shell, but as one develops the rule base it is possible to classify one rule at a time, or gather newly defined rules and process them in batch mode. Finally, gathering information on how many times a rule is used, how many times a rule is used to initiate a derivation, and how many times a rule is used to conclude a derivation can prove to be very useful information in a more complex scheme of classifying a rule-based system.

ACKNOWLEDGEMENT

I would like to thank Bell-Northern Research Ltd. for the resources that were used in order for this research to be completed and published.

REFERENCES

[1] S. Anastasiadis (1990). CLIPS Enhanced with Objects, Backward Chaining, and Explanation Facilities. *First CLIPS Conference Proceedings,* Houston, pp. 621-641.

[2] B. Chandrasekharan (1986). Generic tasks in knowledge based reasoning : High-level building blocks for expert system design. *IEEE Expert, Fall 1986.*

[3] W.J. Clancey (1983). The advantages of abstract control knowledge in expert system design. *National Conference on Artificial Intelligence,* pp. 74-78, 1983.

[4] R. Duda, P Hart (1973). *Pattern classification and scene analysis,* New York : Wiley, 1973.

[5] W. Highleyman (1962). The design and analysis of pattern recognition experiments. *Bell System Tech. Journal,* vol. 41, pp. 723-744, Mar. 1962.

[6] R.V. Hogs, E.A. Tanis (1977). Bayesian Decision Theory. *Probability and Statistical Inference*, Macmillan.

[7] M.R. Genesereth, N.J. Nilsson (1987). Knowledge and Belief. *Logical Foundations of Artificial Intelligence,* Morgan Kaufmann.

[8] P. Lachenbruch (1967). An almost unbiased method for obtaining confidence intervals for the probability of misclassification in discriminant analysis. *Biometrics,* vol. 23 , pp. 639-645, Dec. 1967.

[9] F.P. Preparata, M.I. Shamos (1985). Convex Hulls : Basic Algorithms. Extensions and Applications. *Computational Geometry,* Springer-Verlag.

[10] S.L. Tanimoto (1987). Productions and Matching, Knowledge Representation. *The Elements of Artificial Intelligence,* Computer Science Press.