Inducing and Using Decision Rules in the GRG Knowledge Discovery System

Ning Shan,[1,2] Howard J. Hamilton,[2], and Nick Cercone[2]

[1] Macro International Inc., 11785 Beltsville Drive, Calverton, MD, USA 20705, E-Mail: ning@cs.uregina.ca

[2] Department of Computer Science, University of Regina, Regina, SK, Canada S4S 0A2, E-Mail: {hamilton,nick}@cs.uregina.ca

Abstract. We are developing the GRG knowledge discovery system for learning decision rules from relational databases. The GRG system generalizes data, reduces the number of attributes, and generates decision rules. A subsystem of this software learns decision rules using familiar and novel rule induction techniques and uses these rules to make decisions. This paper provides an overview of GRG, describes those aspects of the system most relevant to creating and using decision rules, and compares it to other machine learning approaches.

1 Introduction

The GRG software system is being designed for knowledge discovery from large relational databases (Shan et al. 1996). *Knowledge discovery* in databases (KDD) is "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" (Piatetsky-Shapiro et al. 1991). Knowledge discovery integrates techniques from statistics, machine learning, logic, and rough sets (Fayyad et al. 1996a; Fayyad et al. 1996b; Ziarko 1994). The central question in knowledge discovery research is how to turn information, expressed in terms of stored data, into knowledge expressed in terms of generalized statements about characteristics of the data. Including a machine learning technique in a knowledge discovery system requires addressing the issues of computational efficiency (to deal with the large amounts of data in databases) and robustness (to cope with missing or "noisy" data). In GRG, efficiency is obtained by generalizing the data to reduce its size and robustness is obtained as described in this paper.

Each of the three steps in the GRG system, *information* **g***eneralization*, *information* **r***eduction*, and *rule* **g***eneration*, transforms the data from the database to a higher level of abstraction. By integrating three approaches, we significantly reduce the computational complexity of analyzing large databases. In the first step, an *attribute-oriented concept tree ascension technique* (Cai et al. 1991) generalizes the data (either a complete relational database or a relation extracted by a query from a database). An $O(n)$ generalization algorithm is used for efficiency (Carter and Hamilton 1997). This generalization loses some information but substantially improves the efficiency of the following steps.

Feature selection (or *attribute reduction*) is the problem of choosing a small subset of attributes that ideally is necessary and sufficient to describe the concept. Feature selection accelerates learning and improves learning quality. We need a reliable and practically efficient method to eliminate irrelevant attributes. In the second step of GRG, a *reduction technique* (Pawlak 1991; Shan et al. 1995) generates a minimalized version of the data, called a *reduct*, which contains a minimal subset of the generalized attributes and a minimum number of distinct rows (tuples) for that subset. Details are given by Shan et al. (1996).

In the third step of GRG (see Section 3), a set of decision rules is derived from the reduct. Many approaches are based on "divide-and-conquer" (Quinlan 1986) or "separate-and-conquer" (Clark and Niblett 1989; Michalski 1983). The former recursively partitions the instance space until each remaining, small instance space belongs to a roughly uniform concept. The latter induces one rule at a time and removes the instances covered by this rule until no more rules can be generated. Such methods suffer from the *splintering problem*, where decisions are made with decreasing statistical support as the size of the sample dwindles. Statistical anomalies become harder to weed out, and noise sensitivity increases. Overfitting occurs, incorrect rules may be generated, and prediction accuracy is decreased (Domingos 1995; Holte et al. 1989). The "conquer-without-separating" strategy used in GRG and other systems (Domingos 1995; Ziarko and Shan 1995) can alleviate the splintering problem. It uses statistical measures to combat noise, because each rule is generated while taking into consideration the entire data set. Missing values are handled without being replaced by artificial values, since each equivalence class is accounted for as a specific rule being generated.

The decision rules embody the general patterns within the database, and can be used to interpret and understand the active mechanisms underlying the database. The technique for making decisions with these rules is described in Section 4. An empirical comparison between GRG and C4.5 on well-known machine learning problems is given in Section 5, and conclusions are drawn in Section 6.

2 Preliminary Stages

The GRG system investigates the relationship between two, user-defined groups of attributes, referred to as *condition attributes* C and *decision attributes* D. Given a set I of input instances (rows), the following preliminary steps are performed: (1) generalize all values according to user-defined concept hierarchies (Ning et al. 1996) and discretization functions (Ning et al. 1997); (2) convert all multivalued attributes in I into binary attributes (let m be the number of condition attributes in the result); (3) create separate decision tables $DT_1, \ldots, DT_d$ for each of d decision attributes; (4) delete all duplicate instances (rows) from $DT_1, \ldots, DT_d$; (5) delete redundant condition attributes from each decision table using the attribute reduction procedure described by Ning et al. (1996). The result of these five preliminary steps is to decomposes the original problem, with d possible values for the decision attribute (or combination of decision attributes), into d subproblems, each with a binary decision attribute.

3 Rule Generation

Rule generation is a key step in GRG. Each instance (row) in a decision table D_i can considered as a specific rule that matches only one equivalence class. When taking this view, we refer to the decision table D_i as a set R of rules. Such a rule can be generalized by removing conditions from the condition part of the rule. GRG's rule generation first checks whether a rule can be made more general by eliminating irrelevant attribute values. An attribute value in a rule is *irrelevant* if it can be removed from the rule without decreasing its expected classification accuracy, as computed from D_i. The resulting rules are called *maximally general rules* (or *minimal rules*) because each rule has the minimum number of conditions required to preserve the classification accuracy of the rule.

More technically, if r_i and r_j are valid rules where $cond(r_i) = cond(r_j)$ and $dec(r_i) = dec(r_j)$, then r_i and r_j are *logically equivalent rules*, where *cond* and *dec* give the values of the condition and decision attributes, respectively. If r_i and r_j are valid rules where $cond(r_i) \subset cond(r_j)$ and $dec(r_i) = dec(r_j)$, then r_j is *logically included* in r_i. If r_i and r_j are valid rules where $cond(r_i) \subseteq cond(r_j)$ and $dec(r_i) \neq dec(r_j)$, then r_i and r_j are *decision inconsistent.*

To obtain a set of maximally general rules R', each rule $r \in R$ is considered for dropping conditions. The algorithm initializes R' to empty and copies one rule $r_i \in R$ to rule r. One by one, each condition is dropped from rule r to create a new rule, and then this rule is checked for decision consistency with every other rule $r_j \in R$. If rule r is inconsistent, then the dropped condition is restored. After all conditions have been examined, the resulting rule r is a maximally generalized rule. Before rule r is added to R', it is checked for redundancy. After all rules have been processed, R' contains a set of maximally general rules that are as general as possible but retain the same classification accuracy as R.

A specific rule $r \in R$ may correspond to more than one maximally general rule. The maximally general rule derived depends on the order in which the attributes are processed. Thus, the maximally general rule obtained may not be best with respect to the conciseness or the coverage of the rule. Rather than evaluate all $2^m - 1$ possible subsets of conditions for a rule with m conditions, we use a heuristic solution based on significance values assigned by an evaluation function to every condition, before starting the process of dropping conditions. The significance value for a condition represents the relevance of the condition for the particular class. Higher significance values indicate greater relevance. In *post-pruning* (Quinlan 1993), conditions with *lower* significance values are dropped *first.* One evaluation function for a condition c_i of a rule is: $\mathrm{SIG}(c_i) = P(c_i)(P(D|c_i) - P(D))$, where $P(c_i)$ is the probability of occurrence of the condition c_i; $P(D|c_i)$ is the conditional probability of the occurrence of the concept D conditioned on the occurrence of the condition c_i; $P(D)$ is the proportion of the concept D in the database (Ziarko and Shan 1995).

An alternative evaluation function, introduced here, is: $\mathrm{SIG}'(c_i) = P(cond - \{c_i\})(P(D|cond - \{c_i\}) - P(D))$. Conditions with *higher* significance values are tested for dropping *first.* If a condition is dropped, the significance values of the remaining conditions are updated.

Algorithm GENRULES: Computes a set of maximally generalized rules.
Input: A set R of specific decision rules
Output: A set R' of maximally general rules

$R' \leftarrow \emptyset$
$n \leftarrow |R|$ /* n is the number of rules in R */
for $i = 0$ **to** $n - 1$ **do**
    $r \leftarrow r_i$
    $m \leftarrow |r|$ /* m is the number of attributes in rule r */
    Compute the significance value SIG' for each condition of rule r
    Sort the set of conditions of rule r based on the significance values
    for $j = 0$ **to** $m - 1$ **do**
        Remove the j^{th} condition attribute C_j in rule r
        if r inconsistent with any rule in $R - r_i$ **then**
            Restore the dropped condition C_j to rule r
        endif
    endfor
    Remove any rule $r' \in R'$ that is logically included in rule r
    if rule r is not logically equivalent to or included in any rule $r' \in R'$ **then**
        $R' \leftarrow R' \bigcup r$
    endif
endfor

Suppose there are n decision rules (i.e., n rows) with m attributes in the set of rules R. The computation of the significance value, SIG', for one rule requires $O(mn)$ and the process of dropping conditions of one rule requires $O(mn)$. Thus, finding a maximally general rule for one decision rule requires $O(mn)$ time and finding maximally general rules for n decision rules requires $O(mn^2)$ time. Eliminating redundant rules is $O(n^2)$. The time complexity of our algorithm is $O(mn^2 + n^2) = O(mn^2)$.

4 Decision Making

Given a set of objects described by several attributes, the decision making problem consists of assigning to each object an appropriate decision value, i.e., classifying each object to an appropriate concept. This classification problem has been studied extensively and the approach described in the previous section is but one of many induction techniques suggested for creating decision rules. The general question of *how to use such decision rules* in the process of decision support has received less attention. A simple approach is adopted by many systems. Given an input object, a rule whose conditions are satisfied by this object, is selected. Typically, the first such rule is selected, although some approaches examine the support of the relevant rules, and choose the best supported rule. These systems use a single rule to suggest a decision and tend to minimize the number of rules used. In general, these approaches ignore the fact that the rules produced from data are inherently uncertain and the associated decision probabilities (if

computed at all) are only crude estimates, rather weakly supported by available data.

On the other hand, the approach implemented in GRG and described below treats each rule as a piece of uncertain evidence, which by itself is of little value with respect to decision making, but which jointly with other rules can provide valuable input to the decision process. GRG uses as much evidence as possible in its decision making and tends to maximize the number of rules used. Such evidence can be combined in many ways, and our method is but one possibility.

We first define the criterion $Q(r)$ for rule quality. Let $[D = 1]$ denote the set of rows where the value of the decision attribute is 1, and let $[r]$ denote the set of rows where the values of the condition attributes are consistent with the condition part of rule r. $Q(r)$ is based on estimates of the following conditional probabilities:

(1) $P(r_C|D = 1)$, the probability that the values of the condition attributes for an input object are consistent with the condition part of rule r, if the value of the decision attribute is 1;
(2) $P(D = 1|r_C)$, the probability that the value of the decision attribute D is 1 for an input object, if the values of the condition attributes for the input object are consistent with the condition part of rule r;
(3) $P(r_C|D = 0)$, defined analogously.
(4) $P(D = 0|r_C)$, defined analogously.

Intuitively, if the random events r_C and $(D = 1)$ are related, then measures (1) and (2) should yield high values and measures (3) and (4) should yield low values. Based on this intuition, the quality criterion is defined as:

$$Q(r) = (P(r|D = 1) + P(D = 1|r) - P(r|D = 0) - P(D = 0|r))/2$$

Clearly, $-1 \leq Q(r) \leq 1$. $Q(r)$ can be seen as a measure of the bias of the set $[r]$ towards the set $[D = 1]$. This measure has two extremes: (1) $Q(r) = 1$ if $[r] = [D = 1]$, and (2) $Q(r) = -1$ if $[r] = [D = 0]$.

During rule generation, a collection of d rule sets corresponding to d possible decisions is created. Each set is used in turn as the current set, L_k $(k = 1, 2, \ldots, d)$. To handle objects which do not match any rule, we add a default rule, which predicts the concept that appears most frequently in the database. Let X denote the input vector of condition attribute values, and let $L_k(X) \subseteq L_k$ be a subset of the current set L_k consisting of those rules whose conditions are satisfied by values of the input vector X. To select one of m possible decisions for X, a promising approach is to compute a decision score $S_k(X)$, for every $k = 1, 2, \ldots, d$, as follows:

$$S_k(X) = \sum_{r \in L_k(X)} Q_k(r).$$

This approach sums the rule quality measures $Q_k(r)$ computed for each rule. Then, the decision with the highest score is selected.

As given, this approach is sensitive to the number of rules in each set L_k $(k = 1, 2, \ldots, d)$. To eliminate this sensitivity, we use the normalized decision score, defined as:

$$NS_k(X) = \begin{cases} -\frac{S_k(X)}{N_k} & \text{if } S_k(X) \leq 0 \\ \\ \frac{S_k(X)}{M_k} & \text{if } S_k(X) > 0 \end{cases}$$

where

$$M_k = \sum_{r \in L_k^+} Q_k(r) \quad \text{and} \quad N_k = \sum_{r \in L_k^-} Q_k(r)$$

and

$$L_k^+ = \{r \in L_k : Q_k(r) > 0\},$$
$$L_k^- = \{r \in L_k : Q_k(r) \leq 0\}.$$

With the normalized definition we have $-1 \leq NS_k(X) \leq 1$.

During decision making, the decision with the normalized decision score closest to 1 is selected. That is, in extreme cases:

$NS_k(X) = 1$ indicates a strong "yes" for the decision k;
$NS_k(X) = -1$ indicates a strong "no" for the decision k; and
$NS_k(X) = 0$ indicates insufficient evidence for decision k.

5 Experimental Results

To evaluate the machine learning subsystem of GRG, we measured the prediction accuracy of the learned rules on test examples. We also compared this accuracy to that for C4.5RULE, the "rule" portion of the output generated by C4.5 run with default settings (Quinlan 1993). We chose 20 databases from the UC Irvine repository. Each database is stored and processed as a single relation table.

For our experiments, we used leave-one-out cross validation for each method on each database. Given a data set containing n objects, *leave-one-out testing* removes one object, generates rules using the remaining $n-1$ objects as a training sample, and tests these rules using the removed object as a test object. This procedure is repeated n times, using each object in turn. The prediction accuracy is calculated as the number of correctly classified objects divided by n. Leave-one-out testing is computationally expensive, but it gives a more reliable estimate of prediction accuracy than other approaches. Given the same algorithm and data set, leave-one-out testing always reports the same prediction accuracy. On the other hand, the more commonly used *ten-fold testing* strategy randomly partitions the data set into ten subsets and uses each in turn as a training sample. When applied repeatedly to the same algorithm, this method yields varying results because different random subsets are chosen.

Table 1 shows the results from the experiments described above; details are given by Shan et al. (1997). In each case, the better result is shown in bold face. For each database and algorithm, we report the prediction accuracy as determined by leave-one-out testing. On some databases, the behavior of the

Data Set	Size	#classes	GRG	C4.5
append	106	2	**93.40**	84.91
australian	690	2	82.03	**82.75**
balance-scale	625	3	**79.36**	77.28
breast-cancer(wisconsin)	699	2	**96.28**	96.00
credit-screening	690	2	**84.93**	84.49
german	1000	2	72.30	**72.50**
glass	214	7	**92.06**	71.50
heart	270	2	**77.04**	74.44
housing	506	2	**85.38**	83.20
ionosphere	351	2	93.16	**94.87**
iris	150	3	**96.00**	95.33
lenses	24	3	79.17	**83.33**
lung-cancer	32	3	**59.38**	50.00
pima	768	2	69.53	**74.87**
segment	2310	7	95.76	**96.10**
shuttle	15	2	**66.67**	40.00
soybean-small	47	4	95.74	**97.87**
tictac	958	2	98.75	**99.48**
wine	178	3	**97.19**	92.70
zoo	101	7	**94.06**	93.06

Table 1. Experimental Results

GRG system and C4.5RULE is similar. On 4 out of 20 databases, the GRG system has a considerably higher prediction accuracy than C4.5RULE. Overall, the prediction accuracy is higher for GRG than for C4.5RULE.

6 Conclusions

We have described the techniques used to create and use decision rules in the GRG knowledge discovery system. Experiments show that, when tested on 20 databases, GRG has generally higher prediction accuracy than C4.5RULE. We do not conclude that GRG is better for all applications than C4.5RULE. Different systems will be suitable for different real-world domains. With the growth of applications of knowledge discovery from databases, a variety of approaches are needed, and the techniques presented here provide a useful addition.

In future work, we will study a variety of measurements for attribute selection. Rule generation may be improved by allowing the generation of more general rules. As well, better results may be obtained by relaxing the current requirement that a general rule have the same classification accuracy as a specific rule. Finally, using partial matching instead of full matching might improve the prediction accuracy.

We have presented an initial description of the theoretical properties and empirical performance of the GRG system considered as a machine learning sys-

tem. Further investigation is warranted to identify situations in which the system should perform well and appropriate settings for parameter values. Additional experiments on large data sets are required to determine how well the GRG system scales up.

References

Cai, Y., Cercone, N., Han, J.: "Attribute-Oriented Induction in Relational Databases," in Piatetsky-Shapiro, G. and Frawley, W.J. (eds.), *Knowledge Discovery in Databases*, AAAI/MIT Press (1991) 213–228

Carter, C.L., Hamilton, H.J.: "Efficient Algorithms for Attribute-Oriented Induction," *IEEE Trans. on Knowledge and Data Engineering* (1997) in press

Clark, P., Niblett, T.: "The CN2 Induction Algorithm," *Machine Learning* **3** (1989) 261–283

Domingos, P.: "Rule Induction and Instance-Based Learning: A Unified Approach," *Proc. IJCAI-95* (1995) 1226–1232

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press (1996)

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *CACM* **39**(11) (1996) 27–34

Michalski, R.S.: "A Theory and Methodology of Inductive Learning," in Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., *Machine Learning: An Artificial Intelligence Approach*, Vol. 1 (1983) 83–134

Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning About Data*, Kluwer Academic (1991)

Piatetsky-Shapiro, G., Frawley, W.J., Matheus, C.J.: "Knowledge Discovery in Databases: An Overview," in Piatetsky-Shapiro, G. and Frawley, W.J. (eds.), *Knowledge Discovery in Databases*, AAAI/MIT Press (1991) 1–27

Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA (1983)

Quinlan, J.R.: "Induction of Decision Tress," *Machine Learning* **1** (1986) 81–106

Shan, N., Ziarko, W., Hamilton, H.J., Cercone, N.: "Using Rough Sets as Tools for Knowledge Discovery", in *Proceedings of the 1st International Conference on Knowledge Discovery & Data Mining (KDD-95)*, Montreal, Canada, August (1995) 263–268

Shan, N., Hamilton, H.J., Cercone, N.: "GRG: Knowledge Discovery Using Information Generalization, Information Reduction, and Rule Generation", *International Journal on Artificial Intelligence Tools*, **5**(1 & 2) (1996) 99–112

Shan, N., Hamilton, H.J., Cercone, N.: "Inducing and Using Decision Rules in the GRG Knowledge Discovery System: Extended Report," Technical Report, Department of Computer Science, University of Regina, Regina, Canada (1997)

Ziarko, W.: *Rough Sets, Fuzzy Sets and Knowledge Discovery*, Springer-Verlag (1994)

Ziarko, W., Shan, N.: "Knowledge Discovery as a Search for Classification," in *Proc. of 23rd Annual Computer Science Conference Workshop on Rough Sets and Database Mining (CSC'95)* (1995)