Using Genetic Algorithms to Evolve a Rule Hierarchy

Robert Cattral, Franz Oppacher, and Dwight Deugo

Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, On K1S 5B6

{rcattral,oppacher,deugo}@scs.carleton.ca

Abstract. This paper describes the implementation and the functioning of RAGA (Rule Acquisition with a Genetic Algorithm), a genetic-algorithm-based data mining system suitable for both supervised and certain types of unsupervised knowledge extraction from large and possibly noisy databases. The genetic engine is modified through the addition of several methods tuned specifically for the task of association rule discovery. A set of genetic operators and techniques are employed to efficiently search the space of potential rules. During this process, RAGA evolves a default hierarchy of rules, where the emphasis is placed on the group rather than each individual rule. Rule sets of this type are kept simple in both individual rule complexity and the total number of rules that are required. In addition, the default hierarchy deals with the problem of overfitting, particularly in classification tasks. Several data mining experiments using RAGA are described.

1 Introduction

Data mining, also known as KDD, or *Knowledge Discovery in Databases*, refers to the attempt to extract previously unknown and potentially useful relations and other information from databases and to present the acquired knowledge in a form that is easily comprehensible to humans (for example, see [1]). It differs from classical machine learning mainly in the fact that the training set is a database stored for purposes unrelated to training a learning algorithm. Consequently, data mining algorithms must cope with large amounts of data, various forms of noise and often unfavorable representations. Because of the requirement of comprehensibility, i.e., that the system be able to communicate the results of its learning in operationally effective and easily understood symbolic form, many approaches to data mining favor symbolic machine learning techniques, typically variants of AQ learning and decision tree induction [2].

RAGA meets the comprehensibility requirement by working with a population of variable-length, symbolic rule structures that can accommodate not just feature-value pairs but arbitrary n-place predicates ($n \geq 0$), while exploiting the proven ability of the Genetic Algorithm (e.g. [3]) to efficiently search large spaces.

Most extant data mining systems perform *supervised learning*, where the system attempts to find concept descriptions for classes that are, together with preclassified examples, supplied to it by a teacher. The task of *unsupervised learning* is much more demanding because here the system is only directed to search the data for interesting associations, and attempts to find the classes by itself by postulating class descriptions for sufficiently many classes to cover all items in the database. We would like to point out that the usual characterization of unsupervised learning as learning without preclassified examples conflates a variety of increasingly difficult learning tasks. These tasks range from detecting potentially useful regularities among the data couched in the provided description language[1] to the discovery of concepts through conceptual clustering and constructive induction, and to the further discovery of empirical laws relating concepts constructed by the system. As will be shown in section 6 below, RAGA is capable of both supervised and (the simplest type of) unsupervised learning. However, since we wish to compare our system to others we emphasize in this paper its use in supervised learning

Sections 2 and 3 briefly describe the rules acquired by RAGA and its major parameters, respectively. Section 5 characterizes the system's peculiar type of evolution, section 6 reports some experimental results and Section 7 concludes.

2 Representation of Data: If-Then Rules

An important type of knowledge acquired by many data mining systems takes the form of *if-then rules*. Such rules state that the presence of one or more items implies or predicts the presence of other items. A typical rule has the form:

$$\textit{If } X_1 \wedge X_2 \wedge \ldots \wedge X_n, \textit{ then } Y_1 \wedge Y_2 \wedge \ldots \wedge Y_n.$$

The data stored internally by RAGA represents rules of this type, with a varying number of conjuncts in the antecedent and/or the consequent. (See section 3). Each part of the antecedent as well as the expression in the consequent can contain n-place predicates· If $n = 0$, the expression is a propositional constant; if $n = 1$, the expression has the widely used form of attribute-value pairs. However, RAGA can handle predicates of any arity.

The antecedents and consequents in association rules can be conjunctions ($\wedge$) and negations ($\neg$) of expressions that are built up from predicates and comparison operators ($=, \neq, <, \leq, >, \geq$). Component expressions can involve boolean variables (e.g. *If* $X \wedge Y$, *then* $\neg Z$), integer and real variables and constants (e.g. *If* $X > 98.6$, *then* $Y = 1$), and percentiles and percentage constants (e.g. *If* $X > 85\% \wedge Y < X$, *then* $Z > 10\%$).

[1] This is the only type of unsupervised learning with which we have experimented thus far.

The ability to generate negated expressions is not enabled by default because uncontrolled use of negations not only increases the search space but also often leads to the production of useless rules. For example, while ***If X*** $\wedge$ ***¬Y, then Z*** may be a good rule, the fitness function should penalize ***If X, then ¬Y*** $\wedge$ ***¬Z*** as useless in many situations where most items are absent in any given transaction. Although the introduction of constraints and rules governing the use of negation may improve the quality and speed of learning, this has not been explored for the experiments reported here.

2.1 Confidence and Support Factors

In general, a rule will be more relevant and useful the higher its confidence and support are. Considered in isolation, i.e., outside a default hierarchy, rules with low confidence are useless because they are frequently wrong, and rules with low support are useless because they report uncommon combinations of items and are frequently inapplicable. It is important to note, however, that if the targets for support and confidence are set too high in unsupervised data mining, useful rules will be missed. Thus, when the confidence target is set too high, redundancies and tautologies will crowd out potentially useful rules. The proper support level is best determined by varying the settings over several analyses on the same data.

3 User Interface and System Configuration

RAGA is configured through a graphical interface, and can be operated by a novice computer user. It is important, however, that a data mining technician familiar with the domain specifics configure each project.

Before RAGA can perform a rule analysis, the variables and predicates with which rules will be built must be defined, and a number of options controlling the Genetic Algorithm component of the system must be chosen.

Unlike other classification algorithms, RAGA supports comparison of attributes. An example of this would be comparing the length of a rectangle with its width. Queries of this type enable the classification algorithm to search beyond single dimensional vectors. However, comparison of attributes is not always desirable. An example of this would be the comparison of dollar amounts in a sales transaction. While it may be useful to compare the prices of different items in a purchase, it is probably not useful to compare the price of a single item to the cost of the entire order (e.g. is the cost of the soda less than the total bill). In cases where the comparison of attributes is deemed fruitless, variable class restrictions can be imposed to prevent variables of certain types from being compared to one another. This has the additional benefit of further narrowing search spaces.

Rule position conditions fix numbers or types of elements in the antecedent or consequent. This is used in classification tasks, where the attributes are always in the antecedent and the class is alone in the consequent. In the absence of these conditions the search is undirected.

4 Evolving a Default Hierarchy

A default hierarchy is a collection of rules that are executed in a particular order. When testing a particular data item against a hierarchy of rules, the rule at the top of the list is tried first. If its antecedent correctly matches the conditions in the element being tested, this top rule is used. If a rule does not apply, then the element is matched against the rule at the next lower level of the hierarchy. This continues until the element matches a rule or the bottom of the hierarchy is reached.

Rules that are incorrect by themselves can be *protected* by rules preceding them in the default hierarchy, and play a useful coverage-extending role, as in the following example:

If (numberOfSides = 4) ^ (length = width) then class = square
If (numberOfSides = 3) then class = triangle
If (numberOfSides > 2) ^ (numberOfSides < 5) then class = rectangle

If the last rule were used out of order, many instances would be improperly classified. In the current position it covers the remaining data items accurately.

Experimentation has shown that rules at the top of the evolved hierarchy cover most of the data, and rules near the bottom often handle exceptional cases. There is a certain amount of overlap between members of the hierarchy, as opposed to having a mutually exclusive set of rules. The evolving hierarchy tends to produce fewer and less complex rules.

In RAGA, the problem of over-fitting is addressed by deciding how to handle the rules at the bottom of the hierarchy. Often these special cases handle only 1 or 2 out of perhaps 5000 data elements. It is important to note, however, that this type of over-fitting is harmless because these rules are only tried as a last resort. If improper classification is considered more costly than leaving the class unknown, the user would simply ignore, after visual inspection, the lower levels of the rule hierarchy.

5 The Genetic Engine Used in RAGA

In order to apply the Genetic Algorithm (GA) to the task of data mining for rules we found it desirable to modify the traditional GA in a number of respects. Accordingly, the genetic engine used by RAGA is a hybrid GA. Perhaps the most drastic modification concerns our choice of representation.

Unlike the traditional GA whose chromosomes are fixed-length binary strings, the GA in our system accommodates rules of varying length and complexity. These rules are expressed in a nonbinary alphabet of user-defined symbols.

The system operates differently depending on whether the current task is classification or otherwise. The primary difference is determining how useful a rule might be, namely its fitness.

The algorithm reads as follows:

Processing of one generation in RAGA involves three steps:

(i) Controlled by two parameters, *ordinary elitism* copies one or more of the current best individuals into the next population to guarantee that the top fitness levels will not drop between generations. *Classification elitism* copies every rule that uniquely covers at least one data item and thus contributes, even if only in a small way, to the set of final rules.

(ii) Next, *fitness proportional selection*, *crossover* and *(macro and micro) mutations* are applied. Until the new population is complete, rule pairs are repeatedly selected and possibly crossed over. Crossover splits rules only between conjunctions. Because of this (and also because of macromutations) rules can grow or shrink during this process. Before the two child rules enter the next phase, they - like all other rules except those copied under elitism - are subjected to micro and macro mutation with bounded rates. Since all rules with positive confidence are macro-mutated, the population size grows during generations.

(iii) Finally, *intergeneration processing* takes place to ensure validity and nonredundancy. Rules may have several comparisons deleted before conforming to what is allowed. If after this point a rule has become invalid or identical to one that already exists in the new population, it is discarded. After enough valid rules have been selected, modified, and inserted into the new population, the evolution for the current generation is complete.

6 Some Experimental Results

The data set tested contains 8124 sample descriptions of 23 species of gilled mushrooms in the Agaricus and Lepiota Family (drawn from [4] and presented in [5]). The data set uses 22 attributes, and classifies each mushroom as either edible (51.8%) or poisonous.

Each of 9 test runs[2] produced between 14 and 25 rules. Each rule set yields 100% accuracy for the entire set. This compares favorably with STAGGER [5] and HILLARY [6], which approach 95% classification accuracy after training on 1000 instances.

Several unsupervised tests were also run on the mushroom data set in an attempt to discover information that is not necessarily related to the predefined classes. When looking for domain specific information that may have nothing to do with edibility (by using rules with 100% confidence and 100% support), we found several rules like the following:

[2] Five test runs used 1000 training instances and the remaining four runs used 7124 instances.

If the *Gill Attachment* is not *Descending* (either: *attached*, *free*, or *notched*), **then** the *Veil Type* is *Partial*.

Unfortunately, we lack the expertise in the given domain to distinguish between interesting domain specific information and well-known facts. In an attempt to automatically discover some facts about edibility, we reduced support to 50%. These tests are difficult to interpret because we lack a tool to compare the results of an undirected and a directed search. We did notice, however, that many of the same attributes used to describe classes are used similarly in the two sets of rules.

7 Conclusion

We have described a flexible new data mining system based on a modified GA. Preliminary experiments show that RAGA's performance compares favorably with that of other approaches to data mining. Unlike the latter, RAGA is also capable of simple forms of unsupervised learning.

In the space of evolutionary approaches, RAGA seems to lie 'half way' between Genetic Algorithms and Genetic Programming: like GP, it uses a variable-length, albeit restricted, representation with a non-binary alphabet, a typed crossover and a macromutation that shares some of the effects with GP crossover; like GA, it uses mutation, and it does not evolve programs. Unlike both GP and GA, it promotes validity and nonredundancy by intergenerational processing on fluctuating numbers of individuals, it implements a form of elitism that causes a wide exploration of the data set, and, by making data coverage a component of fitness, it automatically evolves default hierarchies of rules.

References

1. Berry, Michael J.A., Linoff, Gordon: Data Mining Techniques. J. Wiley & Sons (1997).
2. Michalski, R., Bratko, I., Kubat, M. Machine Learning and Data Mining. Wiley, New York (1998).
3. Mitchell, Melanie: An Introducation to Genetic Algorithms. MIT Press, Mass (1996).
4. Lincoff, G. H. The Audubon Society Field Guide to North American Mushrooms. Alfred A. Knopf, New York (1981).
5. Schlimmer, J. S. Concept Acquisition through Representational Adjustment (TR87-19). Computer Science, University of California, Irvine (1987).
6. Iba, W., Wogulis, J., Langley, P. Trading off Simplicity and Coverage in Incremental Concept Learning. Proceedings of the 5th International conference on Machine Learning, 73-39. Morgan Kaufmann, Ann Arbor, Michigan (1988).