Rule Acquisition with a Genetic Algorithm

Robert Cattral
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, On K1S 5B6
rcattral@scs.carleton.ca

Franz Oppacher
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, On K1S 5B6
oppacher@scs.carleton.ca

Dwight Deugo
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, On K1S 5B6
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. RAGA differs from a standard Genetic Algorithm in several crucial respects, including the following: (i) its 'chromosomes' are variable-length symbolic structures, i.e. association rules that may contain n-place predicates ($n \geq 0$), (ii) besides typed crossover and mutation operators, it uses macromutations as generalization and specialization operators to efficiently explore the space of rules, and (iii) it evolves a default hierarchy of rules. Several data mining experiments with the system 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 [Berry, Linoff 97]). 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 (Michalski et al. 98).

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 (Holland 75, Mitchell 96) 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 5 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 4 characterizes the system's peculiar type of evolution, section 5 reports some experimental results and Section 6 concludes.

2 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

$$If\ X_1 \wedge X_2 \wedge \ldots \wedge X_n,\ then\ Y.$$

In RAGA different rules will often have a different number of conjuncts in the antecedent and in the consequent, and a user-supplied parameter limits the

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

0-7803-5536-9/99/$10.00 ©1999 IEEE

maximum number of such conjuncts in the antecedent and/or the consequent (see section 3 below). 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.

Consider the following example rule: "If a customer buys eggs, then there is an 85% chance that bacon will be bought during the same transaction. This occurs in 10% of all purchases." To assess the quality, i.e., accuracy and importance, of such a rule, its confidence and support are determined.

The *confidence* for a given rule is a measure of how often the consequent is true, given that the antecedent is true. If the consequent is false while the antecedent is true, then the rule is also false. If the antecedent is not matched by a given data item, then this item does not contribute to the determination of the confidence of the rule. In the example rule above, the confidence is 85%.

The *support* indicates how often the rule holds in a set of data. This is a relative measure determined by dividing the number of data that the rule covers, i.e., that support the rule, by the total number of data in the set. In the previous example, the support is 10%.

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[2], useful rules will be missed. Thus, when the confidence target is set too high, potentially useful rules will be crowded out by redundancies and tautologies. The proper support level is best determined by running several analyses on the same data with different settings.

The antecedents and consequents in association rules can be conjunctions ($\wedge$) and negations ($\neg$) of expressions that are built up from predicates[3] 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* $X > 85\%$, *then* $Y > 10\%$).

The ability to generate negated expressions is not enabled by default because uncontrolled use of negations not only increases the search space but often leads to the production of useless rules. For example, while ***If*** $X \wedge \neg Y$, ***then*** Z may be a good rule, the fitness function should penalize ***If X, then*** $\neg Y \wedge \neg Z$ 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 Support and confidence targets are invariably set to 100% in classification tasks.

3 Before an n-place predicate other than a comparison operator can be used in a learning task an evaluation procedure has to be written for it.

3 Configuring the System

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.

For each variable or predicate being defined the user must specify the following: an identifying label and an English translation (for reporting purposes); the type of variable and, if appropriate, its range and precision; the allowable comparison operators; and the formula for a calculation (if it is a derived field) or the program for an evaluation (if it is a predicate). Optionally, variable class restrictions and rule position conditions may be defined. Variable class restrictions narrow search spaces by preventing variables of given types from being compared to one another. Rule position conditions fix numbers or types of elements in the antecedent or consequent; in their absence, the search is undirected.

User-specifiable options for the Genetic Algorithm component include the following:

Fitness function: see section 4 below.

Population size: between generations the population can grow to more than twice its preset size because for each rule with a confidence strictly between 0 and 100, RAGA copies the rule itself as well as two macromutations of it (a generalization and a specialization) into the next generation.

Initial and ***overall maximum antecedent size***: this specifies the maximum number of conjunctions in the antecedent, in the initial population and in the course of evolution.

Initial and ***overall maximum consequent size***: for classification, both parameters are 1.

Crossover and ***mutation rates***: crossover is typed with a standard rate of 0.8. *Micromutations* operate within conjuncts and *macromutations* achieve generalization and specialization by deleting and adding conjuncts. The rates of both types of mutations vary within preset bounds (0.06 to 0.75 and 0.01 to 0.1, respectively).

Termination condition: processing stops after a specified number of generations, or - in classification tasks - when full data coverage is achieved, or - in unsupervised tasks - when a specified percentage of rules achieve the specified minimum fitness.

Elitism and ***classification elitism***: when *elitism* is selected, *deviation(%)* specifies how far below the best fit individual a rule can be to guarantee its survival into the next generation, and *max copy(%)* limits the number of

candidates for elitism copying. When *classification elitism* is selected, every rule that covers some data element not covered by any other rule is copied to the next generation. This assures that minimum coverage (see section 4 below) will not decrease between generations.

Extra fitness rewards: the user can specify a reward constant to promote *parsimony*, i.e., to prefer shorter rules to longer ones, thus implementing a form of the *minimal description length principle*, and can also force the fitness function to take the *data coverage* (see section 4 below) into account. This latter option encourages the formation of rule niches.

Intergeneration processing: this 'edits out' various redundancies, easily detectable tautologies, and contradictions to speed up the search. For example, the rule ***If*** $(A > 5) \wedge (A > 6)$ ***, then*** $(B = 5) \wedge (B \leq 30)$ will be replaced by ***If*** $(A > 5)$ ***, then*** $(B \leq 30)$ or by ***If*** $(A > 6)$ ***, then*** $(B = 5)$, depending on whether the user prefers to drop specific or general conditions, respectively. Similarly, the rule ***If*** $(A = 5) \wedge (A \geq 5)$ ***, then*** $(B = 5) \wedge (B > 30)$ will be replaced by ***If*** $(A \geq 5)$ ***, then*** $(B > 30)$ or by ***If*** $(A = 5)$ ***, then*** $(B = 5)$. While these techniques effectively reduce the search space, they significantly increase the risk of prematurely losing important genetic material in small populations.

4 The modified GA component 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. 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 initial population is generated randomly but we have found it occasionally useful to seed the initial population, for purposes of rule refinement, with previously learned rules. As noted above, we allow the population to grow between generations to more than twice its preset size by enforcing generalizing and specializing macromutations on rules with non-zero confidence. To further reduce the chance of premature convergence our system also disallows duplicate members.

The rule-generating algorithm is quite restrictive in a manner reminiscent of typed Genetic Programming (Montana 95) to cope with the relatively unconstrained representation. It takes syntactic precautions to prevent comparison of variables of different types or from different classes, of constants with other constants, and of variables with themselves, and it ensures that comparison operators are inserted as specified and that Boolean variables do not appear both negated and unnegated in the same rule. Besides such validity checks it also performs simple redundancy checks to ensure, for example, that the same comparison does not exist in both antecedent and consequent. This process of enforcing correctness and eliminating redundancy may seem unnecessary because poor rules can be expected to be eliminated during selection. However, several experiments convinced us that the implemented restrictions, besides obviously narrowing the search space, frequently yield better rule sets.

Since rule validity is explicitly taken care of during rule generation, rule fitness depends primarily on confidence and support, and optionally on a parsimony reward and - for classification tasks - on a reward based on data coverage. For each proposed rule, confidence and support are calculated based on the available data (see section 2). If a rule's confidence = 0 or if it covers no new data, it gets 0 fitness. Otherwise the distance of its confidence and support from the target values is determined, using a quasi-Euclidean distance:

$$Fitness := 100 - \text{sqrt}[\ (\text{confidence} - \text{confidence target})^2 + (\text{support} - \text{supportTarget})^2\ ]$$

The resulting value, normalized as a percentage, may be adjusted by rewarding parsimony, i.e., by adding a small reward constant for each conjunct by which the maximum antecedent size exceeds the rule's antecedent size, so that

$$fitness := fitness + (maxAntSize - antSize) * reward$$

In classification tasks, this fitness value is further adjusted by taking the rule's data coverage into account. The rules are maintained in descending order of fitness. When several rules all cover the same data, only the rule with the highest confidence and support gets credited for these data. This encourages an extensive exploration of the data set. The *data coverage* of a rule is the percentage of data covered by it for which it receives credit because it has the highest combined confidence and support value among rules covering these data. For example, if a rule has 50% coverage, then it was credited as the preferred rule for half the examples covered by it; the other half are covered by rules preceding it in the fitness-sorted rule list.

Two measures of *data coverage* - perfect coverage and estimated coverage - are used during classification to promote the automatic formation of a *default hierarchy*. A rule is considered to be perfect if it has 100% confidence, and the data it covers do not overlap with data covered by any previous rule (in the list of rules sorted by descending fitness). *Perfect coverage* is the percentage of the data set that is covered by perfect rules. If the perfect coverage is 100%, then the entire rule set is made up of non-overlapping, 100% accurate rules. Since data coverage is a component of rule fitness, rules that add to the perfect coverage are at the beginning of the rule list and form the top of the default hierarchy; they can be interchanged without affecting the outcome of the classification task.

Estimated coverage is the estimated percentage of the data set that is correctly covered by the current default hierarchy[4]. When estimated coverage is greater than perfect coverage, some rules overlap in the data they cover, but still produce correct results because of their position in the hierarchy. Rules that are incorrect by themselves but are 'protected' by rules preceding them in the default hierarchy may play a useful coverage-extending role, as in the following example:

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

If the last rule were used out of order, many instances would be improperly classified. As it is, it covers the remaining data items (after applying the first two rules) accurately.

Rules at the top of the hierarchy cover most of the data, and rules near the bottom often handle exceptional cases[5]. Because overlap in the data covered by the rules is admissible, the individual rules can be less complex, and there can be fewer of them. If improper classification is considered more costly than leaving the class unknown, the user would simply not use the entire set of rules. For example, choosing to use only the rules with perfect coverage would often leave items unanswered, but would certainly avoid the problem of over-fitting.

Processing of one generation in RAGA involves 3 steps:
(i) Controlled by two parameters (see section 3), *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.

4 Because minimum coverage depends on all rules, it is always equal to or greater than perfect coverage.
5 Rules at the bottom are often very specific, handling only 1 or 2 out of 5000 data elements. Note that this type of over-fitting is harmless in the context of our system because these rules are only tried as a last resort.

(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.

5 Some Experimental Results

The first data set tested contains 8124 sample descriptions of 23 species of gilled mushrooms in the Agaricus and Lepiota Family (drawn from [Lincoff 81] and presented in [Schlimmer 87]). The data set uses 22 attributes, and classifies each mushroom as either edible (51.8%) or poisonous.

Each of 9 test runs[6] produced between 14 and 25 rules. Each rule set yields 100% accuracy for the entire set. This compares favorably with STAGGER (Schlimmer 87) and HILLARY (Iba et al. 88) 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:

If the *Gill Attachment* is not *Descending* (ie: *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.

The second data set tested is the Vehicle Silhouette data set (Siebert 87). It uses 18 continuous attributes to classify vehicles into four different classes based on their silhouettes. It contains 846 records, with 212, 217, 218, and 199 elements in each of the four classes. Several tests with either 500 or 700 training instances all yielded similar results. Depending on the size of the working population, the accuracy on the training set is between 95% and 100%, but the testing accuracy is only between 62% and 71%. This is a higher error rate than is achieved with C4.5 (StatLog 94),

6 Five test runs used 1000 training instances and the remaining four runs used 7124 instances.

which has an error rate of 6.5% in training, and 26.6% in testing.

Each of our rules covers only surprisingly few instances, so that 60 - 90 rules are required to cover the entire set. This seems rather high, although we have been unable to find out how many rules are needed to achieve the published coverage in other approaches. One example, the rule with the highest coverage (66 records out of 750 training instances) is:

If (*Max length aspect ratio* >= 9) **and** (*Scaled radius of Gyration* > 112) **and** (*Distance circularity* >= 81) **and** (*Scaled variance* <= 403) **then** the *vehicle* is a *Van.*

6 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.

Bibliography

[Berry, Linoff 97] Michael J.A. Berry, Gordon Linoff. Data Mining Techniques. John Wiley & Sons, 1997.

[Mitchell 96] Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, Cambridge, Mass., 1996.

[Cabena et al. 98] Cabena, Hadjinian, Stadler, Verhees, Zanasi. Discovering Data Mining from Concept to Implementation. Prentice Hall, 1998.

[Holland 75] John H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan press, 1975.

[Iba et al. 88] W. Iba, J. Wogulis, P. Langley. Trading off Simplicity and Coverage in Incremental Concept Learning. In: Proceedings of the 5th International Conference on Machine Learning, 73-79. Morgan Kaufmann, Ann Arbor, Michigan, 1988.

[Koza 92] John R. Koza. Genetic Programming: On the programming of computers by means of natural selection. MIT press, Cambridge, Mass., 1992.

[Lincoff 81] G. H. Lincoff. The Audubon Society Field Guide to North American Mushrooms. Alfred A. Knopf, New York, 1981.

[Michalski et al. 98] R. Michalski, I. Bratko, M. Kubat, Machine Learning and Data Mining. Wiley, New York, 1998.

[Schlimmer 87] J. S. Schlimmer. Concept Acquisition Through Representational Adjustment (Technical Report 87-19). Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, 1987.

[StatLog 94] P.Brazdil, J.Gama. LIACC, University of Porto Rua Campo Alegre 823 4150 Porto, Portugal. 1991-1994.

[Siebert 87] J.P. Siebert. Turing Institute Research Memorandum TIRM-87-018 "Vehicle Recognition Using Rule Based Methods". Turing Institute, Glasgow, Scotland, 1987.