Evolutionary Data Mining With Automatic Rule Generalization

ROBERT CATTRAL, FRANZ OPPACHER, DWIGHT DEUGO
Intelligent Systems Research Unit
School of Computer Science
Carleton University
Ottawa, On K1S 5B6
Canada

Abstract: This paper describes RAGA, a data mining system that combines evolutionary and symbolic machine learning methods, and discusses recent extensions required to extract comprehensible and strong rules from a very challenging dataset. RAGA relies on evolutionary search to highlight strong rules to which symbolic generalization techniques are applied between generations. We present some experimental results and a comparison of RAGA with other data mining systems.

Key-Words: Data mining, genetic algorithms, rule hierarchies, classification, machine learning, data set

1 Introduction

RAGA [1, 2] is a data mining system that uses a Genetic Algorithm [4] (GA) / Genetic Programming [5] (GP) based engine to extract knowledge in the form of predictive rules. The system was designed for the tasks of both supervised and certain types of unsupervised learning. Most rule induction algorithms perform a local search by selecting one attribute at a time. We prefer evolutionary algorithms because of their more global style of search. Also, many data mining systems applied in practice rely on a representation language restricted to 1-place predicates (for example, See-5 [6]). These systems cannot discover relationships between attributes and are, thus, incapable of finding strong rules in many domains. GP-based systems, on the other hand, are not subject to similar restrictions in expressive power.

Section 1 introduces RAGA and section 2 describes the Poker Hand Dataset and the challenges it poses to data mining systems. Section 3 details how RAGA combines evolutionary and classical machine learning techniques to evolve better and more comprehensible rules. In section 4 we present experimental results including comparisons with other data mining systems, and section 5 concludes.

2 Description of RAGA

RAGA is an evolutionary data mining system that uses a hybrid GA and GP engine. Although the system is fully described in [1] and [2], below is a brief description of several key points.

The quality of a rule depends on several statistical and subjective factors. These include the confidence, the coverage, and how useful and how interesting the rule is within the given domain. Because the latter two features are subjective there is no widely accepted method for determining them, and no common scale to rate or compare them.

Confidence (rule accuracy) is defined to be the percentage of times that the consequent is true given that the antecedent is true. If the consequent is false while the antecedent is true, the confidence for the given rule drops. If the antecedent is not matched by a data item, then this item does not contribute to the determination of the confidence of the rule.

Support (rule coverage) is defined as the number of data elements that are correctly answered using the rule, divided by the total number of elements in the set.

RAGA aims to find rules of the form:
If $\mathbf{X_1 \wedge X_2 \wedge \ldots \wedge X_n}$ ***Then*** $\mathbf{Y_1 \wedge Y_2 \wedge \ldots \wedge Y_m}$.

The symbols $X_1 \ldots X_n$, and $Y_1 \ldots Y_m$ each represent terms within the rule, where a term is a function that either indicates the existence of a value, or performs an operation on two or more attributes. In classification tasks the value for *m* is always 1, while the value for *n* is unbounded.

The genetic engine used by RAGA is a hybrid of GA and GP, with several modifications and additions to the standard models. These additions include a set of contraints that define valid population members, a plug-in style fitness function, specialized crossover and mutation operations, and a non-evolutionary component called *Intergenerational Processing*.

In order for a rule to belong to a population it must first pass tests that determine *uniqueness* and *validity*, according to the language specified by the user. Because it is possible that invalid rules will be generated and evolved throughout the life of the search, each rule is examined before being allowed into the population.

The *plug-in style fitness function* starts by calculating a *raw fitness*, which is an objective measure of how close the confidence and support factors are to target values specified by the user. This result is used as the initial fitness for each rule before several penalty and bonus functions are applied. The functions and parameters that are selected are used to tune the fitness such that it is relative to other rules in the population. The inclusion of different fitness modifiers depends on the definition of the problem. For example, certain fitness plug-ins are designed to maximize data coverage in classification tasks, but would not contribute positively to the results of an unsupervised search.

RAGA uses several fairly standard *crossover* and *mutation* operators whose results must conform to the population validity constraints. Crossover is used to extract and make wider use of important rule fragments, while the mutation operators are used to probe untested solutions and fine-tune existing rules.

The *intergeneration processing* refers to several tasks that are run between generations. These tasks vary depending on user specifications, but can include operations such as trimming the dataset of rules that are logical tautologies or contradictions. Previously in RAGA, the intergenerational processing stage was primarily used to ensure rule validity and to omit rules based on the quality of their output, and to further fine-tune rules to make them more general. Because of the benefits realized during this stage, it has been greatly expanded (see sections 3 and 4).

2 Dataset: Poker Hand Scores

One way to address the problem of evaluating results from a rule induction algorithm is to generate a set of random data that conforms to a pre-specified set of known rules. The data can optionally contain noise, incorrect and missing values, as well as non-essential attributes. The quality of the algorithm can be judged by the correlation between the original rules and the discovered rules. The drawback to this method stems from the fact that these datasets are purely abstract and have no real-world meaning.

During the analysis phase it is trivial to look for rules that are an identical match, however in cases where the discovered rules are different it is sometimes difficult to determine their worth. For example, if a discovered rule is an optimized version of the original then it may be deemed incorrect because the human analyst cannot see the relationship.

Another problem with synthetic datasets is that they can represent a search space that is unbounded and unfair to some learning algorithms. Similarly, some synthetic datasets cannot be correctly classified by algorithms that are not capable of relational learning. This is shown using the polygon dataset in [1].

The poker hand dataset, which is more completely described below, was generated with the intention that it be difficult[1] to discover solutions yet easy to analyze them. Because of the way that the problem is represented it is difficult to discover rules that can correctly classify poker hands, however the simple nature of the game makes it trivial for the human analyst to validate potential rules objectively. The solution space is bounded because there are a finite number of valid poker hands. A valid hand is restricted to five unique cards in any position drawn from a standard deck of 52 cards.

Further advantages of the poker hand dataset are that it can be readily mapped onto several other real-world problem domains such as resource allocation in a network, and that it facilitates rule evaluation even in unsupervised learning tasks.

Although several different poker hand scoring systems exist we have chosen a model that is used to score the worth of hands in video poker. This system has more ranks (or different scores) than standard poker because certain hands score higher based on the possibility of a higher payout, rather than simply accounting for winning or losing combinations. In some cases a hand is considered good because it represents a solid stepping stone

[1] The difficulty of the task can be estimated from the astonishingly poor performance of other data mining systems such as See-5 (see section 4).

towards a winning hand.

The scores are ordered such that they increase with the potential payout. For example, a Royal Flush has the greatest payout. Normally the hands that are guaranteed to pay out are at the top of the scale, with one exception: Score = 10. A hand that is a single card away from a Royal Flush is more important than some other winning hands simply because the payout is so large that it is worth giving up the guaranteed win for the chance.

The example hands show all of the important cards in bold face and sorted, however the actual ordering of the cards within the hands does not matter. For example, a three-of-a-kind is obtained regardless of the positions the three like-valued cards are in.

The following table describes the 17 different ways that we score poker hands:

Score	**Name**	**Description**	**Example**
16*	Royal Flush	Ace → Ten of same suit	**AH KH QH JH 10H**
15*	Straight Flush	Five sequential cards, same suit	**4C 5C 6C 7C 8C**
14*	Four of a kind	Four of the same card	**2H 2D 2S 2C** 8S
13*	Full house	Three of a kind plus one pair	**3D 3C 7H 7S 7D**
12*	Flush	Five cards of the same suit	**2C 3C 6C 9C AC**
11*	Straight	Five sequential cards	**3C 4C 5D 6H 7D**
10	Four to RF	Four cards towards Royal Flush	**AD KD QD JD** 4S
9*	Three of a kind	Three equal cards	**5H 5S 5D** 3C 7H
8*	Two pairs	Two pairs of equal cards	**4H 4S 9D 9S** 7C
7*	One high pair	Two equal cards (Jacks or better)	**QH QD** 2C KH 9S
6	Four to Flush	Four cards of the same suit	**2C 5C 7C 9C** 4H
5	Four to Straight	Four sequential cards (2 < c < A)	**3H 4D 5D 6C** AS
4	Three to RF	Three cards towards Royal Flush	**KD QD JD** 9S 8H
3	One low pair	Two equal cards (Ten or less)	**5H 5C** 7D JS 2D
2	Two high cards	Two cards (Jacks or better)	**JD QC** 2H 9C 5D
1	One high card	One card (Jack or better)	**AS** 4H 8D 3S 10H
0	Nothing	No useful cards in the group	2S 3D 6H 9C 10H

Table 1: Video poker scores

3 What is new in RAGA?

The original system uses a purely evolutionary search, and is enhanced by performing several types of intergenerational processing (for example, rules deemed invalid are modified by systematically deleting terms from the antecedent, consequent, or both. If this does not render the rule valid because of the rule structure or an excess of similar members in the population, a complete substitution is made). Because of the success realized with intergenerational processing (faster discovery of stronger rules), we decided to expand its role to include machine learning techniques, in particular, rule generalization techniques that can be used to abstract higher level rules from groups of related and more specific rules.

The most recent work in RAGA was the design and implementation of a system to examine and generalize sets of rules based on functions contained within a library. These functions include operations such as combining commonly appearing sets of items together into new symbols, and the examination of rule groups with common themes (eg: identical consequent) for the purpose of introducing higher level functions. One benefit to this operation is that the results are more comprehensible.

An example of how generalization can improve comprehensibility can be seen in the poker hand dataset. The example rules below classify some subset of the data as a three-of-a-kind. Notice that these rules are not 100% confident over all possible hands because they do not exclude the case of a Full House, however the addition of new terms to achieve perfect accuracy is not necessary to see the benefits of generalization.

```
If (R3 = R5)∧(R5 = R4)∧(R5 != R1)∧(R2 != R4) then (SCORE = 9)
If (R2 != R1)∧(R2 = R5)∧(R2 != R3)∧(R2 = R4) then (SCORE = 9)
```

A simple examination of these rules reveals that they are specific to the positions of certain cards, which means that a large number of rules are required to fully describe this particular score.

When RAGA notices that all of these rules have the same consequent it will automatically attempt to generalize them. This is not done for every rule because the operation is computationally very expensive, however in our case we rely on evolution to choose the subsets that warrant further examination. After testing a number of library functions (eg: transitivity of equality, search for linear relationships, etc) the entire group of rules might be replaced by a single rule as follows:

```
If (NumEqualValues(class = rank) =
3) then (SCORE = 9)
```

Translation: "Within the subset of attributes that are specified to be of class 'rank' by the user, if there are exactly three values that are equivalents then the hand scores as a three-of-a-kind."

When this technique is applied then a subset of the rules will be compressed into a single, more general and more comprehensible rule. Apart from being more comprehensible, this extra layer of abstraction can also increase the overall rule coverage. The coverage will not change if the original set of rules was complete and contained all of the permutations required to satisfy the data, however this is not normally the case. In addition to the original rules being replaced there may be a number of undiscovered rules that are also incorporated automatically. Although there is a risk of over-generalizing, this result will be penalized during the evaluation stage. Similarly, if other rules in the population become redundant because of this new rule then the evolutionary component in RAGA will have them replaced during the next generation.

Another type of abstraction is due to the creation of aggregate functions. An example of this can be seen in the following rules, which were selected as a candidate subset because they all have 100% confidence:

Original rules

```
If (A=5)∧(B>D)∧(C=30) then (Type=2)
If (A=5)∧(C<15)∧(B>D) then (Type=3)
If (A=5)∧(B>D)∧(C>30) then (Type=1)
```

After replacement

```
If (#A)∧(C=30) then (Type=2)
If (#A)∧(C<15) then (Type=3)
If (#A)∧(C>30) then (Type=1)
```

When RAGA examines these rules it finds that the most commonly occurring operations are: **(A = 5)** and **(B > D)**. Because this configuration appears to be a key component to several rules, RAGA creates a function that represents the conjunction of these comparisons. (The new function is denoted as **#A** above).

The technique is similar to *chunking*, which has been used in [3] as the primary mechanism for learning from experience. The aggregate functions become building blocks that extend the representation language. Since they are extensions of the vocabulary, they are protected from disruption by crossover, and only need to be discovered once.

4. Experiments

A number of experiments were performed to test the results generated by See-5 and RAGA. The first step was to generate a set of classifiers (a decision tree for See-5 and a rule hierarchy for RAGA) from a sample set of 10000 data. Afterwards, several tests sets of size 1000 were used to test the classifiers.

When See-5 analyses this data it reports a training accuracy of 64.25%, however after testing the average correctness is only 36.16%. Post analysis of these results seem to indicate that much of the correctness comes from class default values, which are assumptions made based on the more popular scores within the data. The poor quality of the rules it found is further illustrated by the fact that the tree it produced had 3946 nodes of which 3430 were leaves!

When RAGA is run on the same dataset it achieves 90.39% correctness on the training set and an average accuracy of 57.6% over all test sets. Analysis of these results indicates that RAGA does not rely on the default hierarchy to boost the predictive accuracy.

5. Conclusion

We have described a hard dataset that the previous version of RAGA could not handle any more satisfactorily than See-5 (see above paragraph). We now tackle this dataset by expanding the Intergenerational Processing component of RAGA to combine evolutionary and non-evolutionary machine learning methods. The system relies on evolution to generate plausible candidate rules, to point out the rules to which computationally expensive, non-evolutionary generalization techniques are applied, and to weed out the results of eventual over-generalizations. The non-evolutionary techniques, by extending the representation language, enhance rule comprehensibility and rule coverage.

6. References

[1] Cattral, Oppacher, Deugo, A Modified Genetic Algorithm for Supervised and Unsupervised Data Mining, *Proceedings of the IASTED International Symposia APPLIED INFORMATICS*. (ACTA Press, Calgary, AB, Canada, *2001).*

[2] Cattral, Oppacher, Deugo, Rule Acquisition with a Genetic Algorithm, *Proceedings of the Congress on Evolutionary Computation*, 1999 Vol. 1, p. 125-129

[3] Allen Newell, *Unified Theories of Cognition*, (Harvard University Press, 1990).

[4] John Holland, *Adaptation in Natural and Artificial Systems*, (Michigan: University of Michigan Press, 1975).

[5] John R. Koza, *Genetic Programming: On the programming of computers by means of natural selection,* (MIT Press, Cambridge, Mass, 1992).

[6] R.J. Quinlan, (1993). C4.5 and See-5 are licensed products that can be acquired from Morgan Kaufmann Publishers, Ann Arbor, Michigan.