An Extended Genetic Rule Induction Algorithm

Juliet Juan Liu
Department of Computer Science
Wuhan University
Wuhan, Hubei
China
liujuan@whu.edu.cn

James Tin-Yau Kwok
Department of Computer Science
Hong Kong Baptist University
Hong Kong
China
jamesk@comp.hkbu.edu.hk

Abstract- This paper describes an extension of a GA-based, separate-and-conquer propositional rule induction algorithm called SIA [24]. While the original algorithm is computationally attractive and is also able to handle both nominal and continuous attributes efficiently, our algorithm further improves it by taking into account of the recent advances in the rule induction and evolutionary computation communities. The refined system has been compared to other GA-based and non GA-based rule learning algorithms on a number of benchmark datasets from the UCI machine learning repository. Results show that the proposed system can achieve higher performance while still produces a smaller number of rules.

1 Introduction

The increasingly widespread use of information system technologies and the internet has resulted in an explosive growth of many business, government and scientific databases. As these terabyte-size databases become prevalent, the traditional approach of using human experts to sift through the data has become infeasible. Rather, the automatic discovery of a set of if-then rules to represent the underlying concept is now increasingly important. Rule induction has been studied by researchers in various fields of machine learning, and a diversity of learning algorithms have emerged. In this paper, we concentrate on the use of genetic algorithms (GA). As demonstrated in various application domains, GA has proved to be an appealing alternative to classical search algorithms for exploring a large search space. In particular, they are robust and less likely to get stuck in local optima. Moreover, they tend to cope better with attribute interaction. Besides, they are highly parallel in nature and therefore attractive to parallel and distributed implementations.

There are two main traditional methods to represent the concepts in GA. In the Michigan approach [13, 25], each individual is represented by a fixed-length string and corresponds to a partial concept description. The target concept is represented by the whole set of individuals in the population. Whereas in the Pittsburgh approach [4, 15, 22, 23], each individual is represented by a variable-length string and corresponds to a whole target concept. The Michigan approach has the advantage that traditional genetic operators can be used without modification. However, various strategies have to be adopted in order to extract a non-redundant concept description from the population. On the other hand, in the Pittsburgh approach, a simple GA can be used as each single individual can already represent the whole multi-modal concept. However, more complex genetic operations and chromosome representations have to be introduced.

To alleviate these problems, some hybrid approaches have been proposed [10, 11, 12, 21]. However, a simpler strategy, which will be used in this paper, is to learn just one disjunct at a time. This *separate-and-conquer* approach has been commonly used in many classical (non GA-based) rule learning algorithms [9], and was applied to the GA setting in SIA [24]. It can effectively restrict the size of the search space. Moreover, it obviates the need of speciation, which often involves various special techniques (such as the universal suffrage operator in REGAL [10] and the coverage-based filter in COGIN [11]).

Another advantage of SIA over many other methods is its ability to deal with continuous attributes. Systems like REGAL use a binary representation, which may become very long for continuous attributes and thus significantly slow down the GA process. On the contrary, SIA uses a high-level representation and hence allows high-level operators to efficiently manipulate the continuous attributes.

In this paper we describe an extension[1] of SIA, with improvements in the initialization, design of the genetic operators, rule filtering and classification procedure. SIA has also been modified for the induction of first order logic (FOL) rules [1]. However, we will only consider propositional logic in the paper, and extension to FOL will be discussed elsewhere. The rest of the paper is organized as follows. Section 2 describes the extended version of SIA in detail. Evaluation on a number of benchmark datasets from the UCI machine learning repository [2] is then presented in Section 3, and the last section gives some concluding remarks.

2 Extended SIA

Extended SIA (ESIA) learns one disjunct at a time, and then all the discovered disjuncts together form the target concept description. It follows the standard strategy (also called *covering* strategy) in separate-and-conquer rule learning algorithms [9]: learn a rule that covers part of the training set,

[1] An earlier extension is reported in [19].

remove the covered examples from the training set and then recursively learn the remaining examples until all are covered. An outline of ESIA is given in Fig 1. The representation scheme of ESIA follows that of SIA, and a brief discussion is included in Section 2.1 for completeness.

```
1.  While some examples are still uncovered
2.  begin
3.    Select an uncovered example ex;
4.    Rinit=GenerateRule(ex);
5.    t=0;
6.    P(t)=GeneratePopulation(Rinit);
7.    While (stopping criteria not met)
8.    begin
9.       Evaluate P(t);
10.      t=t+1;
11.      Select P(t) from P(t-1);
12.      Crossover();
13.      Mutate();
14.      Generalize();
15.      Specialize();
16.      DropCondition();
17.   end
18.   Add the optimal rule(s) to rule set;
19. end
20. FilterRuleSet();
```

Figure 1: Pseudo-code for ESIA.

ESIA, however, differs from SIA in various ways such as in the initialization process (Section 2.2) and also in the design of genetic operators. While SIA uses only the crossover and generalization operators[2] with fixed probabilities, we have enriched the set of genetic operators in ESIA by

1. adding the mutation and specialization operators (Section 2.3), which have been commonly used in other GA-based rule induction systems;
2. introducing a new task-dependent operator (Section 2.4); and
3. dynamically adapting the probabilities of selecting genetic operators (Sections 2.5).

ESIA also differs from SIA in the other aspects as the definition of the fitness function (Section 2.6), rule filtering procedures (Section 2.7) and classification procedure (Section 2.8).

2.1 Representations for Rules and Examples

Given a problem domain with m (nominal or continuous) attributes $\{A_i\}_{i=1}^m$, rules learned by ESIA are of the form:

$$\text{IF } cond_1 \wedge \cdots \wedge cond_m \text{ THEN } class = C_R,$$

and is represented by the corresponding chromosome $\langle cond_1, \cdots, cond_m, C_R \rangle$. Here, C_R is the class[3] output for this rule, and $cond_i$ is a condition on A_i that can either be:

- "true", meaning that the condition is always true, or
- "$A_i = value$" for a nominal attribute A_i, or
- "$B \leq A_i \leq B'$" for a continuous attribute A_i.

Similarly, examples are also represented by tuples of the form $\langle v_1, \cdots, v_m, C_E \rangle$. As in SIA, examples with missing values are acceptable, and these are denoted by the special symbol "?".

2.2 Non-random Initialization

Most systems, like SIA, use random selection of an uncovered example (*seed*) for the initialization of the population, and they may thus be sensitive to the order of examples selected. Experiments have shown that non-random initialization, or *inoculation*, can often improve the average solution quality and improves the runtime [20]. Here, as we are looking for rules with high consistency and coverage, intuitively then the initial seed should lie inside a cluster of examples belonging to the same class. Thus, we implement a non-random seeding method in ESIA by computing, for each example, the ratio of the numbers of same-class / opposite-class examples lying within a user-defined radius ρ. The one with the highest ratio will be selected as the initial seed.

After this, GenerateRule in Figure 1 will generate a most-specific rule Rinit to cover this seed, and then GeneratePopulation will produce an initial population by applying the generalization operator on Rinit. This is followed by the GA process (lines 7-17 in Figure 1) to find the optimal rules.

2.3 Addition of Mutation and Specialization Operators

While SIA uses only the generalization and crossover operators to change the chromosomes, we add in two other commonly-used genetic operators: mutation and specialization. The mutation operator helps in maintaining the diversity within the population and also in preventing premature convergence to local optima. Here, mutation operates by first randomly selecting a condition from the rule. If that involves a nominal attribute, then the value will be randomly changed from one to the other. Otherwise, if the attribute is continuous, mutation will randomly change the condition's interval values (B and B').

The specialization operator specializes a randomly selected condition in the rule. If that condition was previously dropped (because of previous applications of the generalization operator or the drop-condition operator in Section 2.4), specialization restores it back into the rule[4]. Otherwise, if

[2] The creation operator in SIA can be regarded as a type of generalization operator.

[3] In the sequel, we will use the terms "class" and "target concept" interchangeably.

[4] This is implemented in ESIA by associating a special delete flag with each condition. Conditions are marked as "deleted" when they are dropped, and reset when they are restored.

the condition involves a continuous attribute, specialization shrinks the interval $B \leq A_i \leq B'$. However, if the condition involves a nominal attribute, then specialization will have no effect.

2.4 Relevance-Based Drop-Condition Operator

The generalization operator in SIA can randomly drop a condition from a particular rule. However, in the presence of a lot of irrelevant attributes, as is common in typical real-world problems, this may be a slow and inefficient process. Here, we introduce an additional drop-condition operator that is particularly directed at conditions involving these irrelevant attributes. We use RELIEFF [17], which is an extension of RELIEF [16] for noisy, incomplete and multi-class problems. Its key idea is to estimate attributes according to how well their values distinguish among examples that are near each other. A relevant attribute should have the same value for neighbors from the same class, while different for neighbors from the other classes. RELIEF has been shown to outperform other measures like information gain, J-measure and gini-index [18].

In the following, let the relevance for the ith attribute computed by RELIEFF be r_i. The relevance-based drop-condition operator then selects a condition (with its corresponding attribute) with probability

$$\frac{1/r_i}{\sum_{i=1}^{m} 1/r_i}.$$

Hence, conditions involving highly irrelevant attributes will more likely be dropped.

2.5 Adapting the Operator Probabilities

While SIA selects the genetic operators with fixed probabilities, it is usually more helpful to dynamically adjust these probabilities based on the fitness of the individual chromosomes [15]. Take the mutation / crossover operators as an example. While it is usually a good idea is to give the less-fit individuals a higher chance to mutate / crossover, it may not be the case for those high-fitness individuals. Hence, in ESIA, we adapt the probability p_m for selecting the mutation operator as:

$$p'_m = p_m + \alpha_m\left(\frac{f_{max} - f}{f_{max} - f_{avg}}\right),$$

where f is the fitness of the individual, f_{avg} and f_{max} are the average and maximal fitness of the population respectively, and α_m is an user-defined parameter. Similarly, the probability p_c for selecting the crossover operator is adapted as:

$$p'_c = p_c + \alpha_c\left(\frac{f_{max} - \bar{f}}{f_{max} - f_{avg}}\right).$$

Here $\bar{f}$ is the average fitness of the two mates, and α_c is another user-defined parameter.

The probabilities for the generalization and specialization operators are also dynamically adapted. In general, we give fitter individuals a higher chance to generalize and specialize so as to allow them for further refinement. Moreover, when an individual covers many examples, the probability of specialization is increased so as to give it a higher chance of creating a more consistent offspring. Otherwise, generalization will be given a higher probability. To sum these up, we adapt the probabilities for generalization and specialization (p_g and p_s respectively) as:

$$\begin{aligned} p'_g &= p_g + \alpha_g f(1-g), \\ p'_s &= p_s + \alpha_s f g. \end{aligned}$$

Here, α_g, α_s are user-defined parameters, and $g = (n^+ + n^-)/N^2$, where n^+ is the number of examples that the rule matches correctly, n^- is the number of examples that the rule match incorrectly, and N is the size of the training set.

2.6 Fitness Function

For a particular rule R, the fitness function is dependent on the three aspects of

- consistency: Here, we use the Laplace estimate $cons(R) = \frac{n^+ + 1}{n^+ + n^- + \#classes}$, which helps to penalize rules with low coverage [9];
- completeness: $compl(R) = n^+/N^+$, where N^+ is the number of examples in the training set belonging to the same target class C_R; and
- rule generality: $gen(R) = 1 - \frac{length(R)}{m}$, where $length(R)$ be the number of conditions in R.

Fitness $f(R)$ is then defined as

$$f(R) = w_1 cons(R) + w_2 compl(R) + w_3 gen(R).$$

Here, we place a higher emphasis on consistency, and w_1 and w_2 are set to be $0.5 + 0.25cons(R)$, and $0.5 - 0.25cons(R)$ respectively.

2.7 Additional Rule Filtering Mechanisms

In SIA, weak rules are removed by checking the "strengths" of the rules alone. Although this can sometimes drastically reduce the number of rules, usually still quite a number of redundant rules are left. Here, we identify three additional kinds of rules that should also be deleted.

1. Noisy rules: If a rule covers more examples from the other classes than from its own, then it is noisy.
2. Redundant rules: For a particular class, if the set of examples covered by a rule is contained in the set of examples covered by another rule, then the former is redundant.
3. Highly incomplete rules: If the number of examples covered by a rule is less than a certain threshold, then the rule is highly incomplete.

2.8 Distance Measure used for Classification

Like SIA, ESIA also classifies a new example by assigning it to the class of the nearest rule in the rule set. To be more specific, the distance between a rule $R = \langle cond_1, \cdots, cond_m, C_R \rangle$ and an example $E = \langle v_1, \cdots, v_m, C_E \rangle$ is measured by:

$$d(R, E) = \sqrt{\sum_{i=1}^{m} \delta^2(cond_i, v_i)}.$$

Considering first the simpler case that there is no missing values, $\delta(cond_i, v_i)$ is defined in SIA as follows:

- if $cond_i = true$, then $\delta(cond_i, v_i) = 0$;
- if $cond_i$ is "$B \leq A_i \leq B'$" with A_i continuous, then

$$\delta(cond_i, v_i) = \begin{cases} 0 & B \leq v_i \leq B', \\ \frac{v_i - B'}{max_i - min_i} & v_i > B', \\ \frac{B - v_i}{max_i - min_i} & v_i < B. \end{cases}$$

  Here, max_i and min_i are the maximum and minimum values respectively of A_i in the training set;
- if $cond_i$ is "$A_i = value$" with A_i nominal, then $\delta(cond_i, v_i) = 0$ if the condition holds, and 1 otherwise.

In ESIA, we adopt the same method in the first two cases, but the simple measure used for nominal attributes is not quite informative and can lead to poor performance [3]. Here, we adopt the simplified value difference metric (SVDM) as defined in [5]:

$$SVDM(cond_i, v_i) = \sum_{j=1}^{\#Class} |P(C_j|cond_i) - P(C_j|A_i = v_i)|.$$

Intuitively, the idea is that two attribute values are considered similar if they make similar predictions (i.e., they correlate similarly with the target concept). Different variants of this metric have been successfully used [3].

Examples with missing values deserve special attention. If the value of a nominal attribute is missing, ESIA, like [5], treats it as another legitimate attribute value and computes $\delta(cond_i, v_i)$ using SVDM. However, if the value of a continuous attribute is missing, [5] simply assumes the distance to be zero. Here, we define instead:

$$\delta(cond_i, v_i) = \begin{cases} 0 & B \leq v_i \leq B', \\ \frac{v_i - B'}{max_i - min_i} & v_i > B', \\ \frac{B - v_i}{max_i - min_i} & v_i < B, \\ \frac{(max_i - min_i) - (B' - B)}{max_i - min_i} & v_i \text{ is missing.} \end{cases}$$

The idea is that if v_i is missing, then it is likely to take values in the observed range $[min_i, max_i]$. Subsequently, the wider is the interval $[B, B']$, the more likely will this condition cover this particular example, and the smaller is the distance $\delta(cond_i, v_i)$.

3 Evaluation

In this section, we report results on applying ESIA to a number of benchmark problems in concept learning. Experiments have been performed on 13 datasets from the UCI machine learning repository[5] [2] (Table 1). We run 10-fold cross-validation and compare ESIA with the original SIA and another successful non GA-based rule learning algorithm called RISE [5]. Based on the results in [5], RISE consistently achieves higher accuracies than both its parent approaches (PEBLS and CN2) as well as the decision tree learner C4.5. To provide a baseline reference, we have also included the performance of the majority classifier, which always predicts the most frequent class.

We used the following parameter settings in the experiments:

- RISE: $q = 1$, $s = 2$, and with global stopping [5];
- SIA: $Nb_{max} = 600, T_{str} = 0.4$ and a population size of 50. The remaining parameters are set as suggested in [24];
- ESIA: $\rho = 0.1, w_3 = 0.1, Nb_{max} = 600, p_c = 0.8, p_m = 0.01, p_g = 0.9, p_s = 0.1, \alpha_c = \alpha_g = 0.3, \alpha_m = \alpha_s = 0.4$, and the population size is 10.

Table 2 shows the accuracies of the various methods averaged over the 10 folds. As can be seen, ESIA outperforms SIA on all datasets. It also achieves better performance than RISE on 9 of the 13 datasets. Table 3 shows the number of rules produced. In general, ESIA can also produce a smaller number of rules.

4 Conclusion

This paper describes an extension of SIA, by taking in advantages of the recent advances in the rule induction and evolutionary computation communities, such as techniques on initialization, design of the genetic operators, rule filtering and distance measuring. In general, the refined system is superior to both the original SIA and RISE on a number of benchmark datasets. Results show that the proposed system can achieve higher performance while still produces a smaller number of rules.

Recently, rather than focusing only on discovery accurate rules from the data, the data mining community is also interested in discovery *comprehensible* and *interesting* rules [7]. Hence, users can understand the system's results rather than blindly trusting a "black box". With ESIA and other GA-based rule induction algorithms, this can easily be done by incorporating various "interestingness" measures [8] into

[5] Following [6, 14], the glass2 variant of the glass dataset has classes 1 and 3 combined and classes 4 to 7 deleted, and the horse-colic dataset has attributes 3, 25, 26, 27, 28 deleted and with attribute 24 being used as the class label. We also deleted all identifier attributes from the datasets.

Table 1: Datasets used in the experiments.

Dataset	#examples	#discrete attributes	#continuous attributes	#classes
annealing	898	32	6	5
australian	690	8	6	2
breast	699	0	9	2
cleveland	303	7	6	2
crx	690	9	6	2
pima indians	768	0	8	2
german	1000	0	24	2
glass	214	0	9	6
glass2	163	0	9	2
heart	270	0	13	2
horse-colic	368	15	7	2
iris	150	0	4	3
vehicle	846	0	18	4

Table 2: Average accuracies and standard deviations over the ten folds (Numbers in bold indicate the highest accuracy obtained over the four methods).

Dataset	majority	RISE	SIA	ESIA
annealing	76.17±0.06	90.65±0.02	86.53±0.03	**93.32** ±0.01
australian	55.51±0.04	**85.36**±0.02	72.46±0.19	80.58±0.10
breast	65.52±0.02	91.85±0.07	84.84±0.02	**94.71** ±0.04
cleveland	54.13±0.21	74.92±0.17	67.66±0.23	**77.23** ±0.24
crx	55.51±0.07	**82.32** ±0.06	69.57±0.16	77.39±0.23
pima indians	65.10±0.02	65.63±0.30	69.14±0.27	**70.18**±0.21
german	70.00±0.00	64.40±0.26	69.90±0.27	**70.50** ±0.26
glass	35.51±0.31	70.56±0.24	53.74±0.40	**72.43** ±0.03
glass2	55.21±1.61	69.94±0.28	76.69±0.15	**78.53** ±0.22
heart	55.56±0.00	69.26±0.26	69.63±0.16	**74.44** ±0.26
horse-colic	63.04±0.09	**83.15** ±0.16	71.74±0.14	72.55±0.12
iris	33.33±0.00	92.67±0.06	92.00±0.01	**95.33** ±0.03
vehicle	26.00±0.20	**70.57** ±0.22	60.52±0.25	67.61±0.15

Table 3: Average number of rules produced over the ten folds (Numbers in bold indicate the smallest number of rules obtained over the three rule-based methods).

Dataset	RISE	SIA	ESIA
annealing	284.3	125.7	**13.3**
australian	380.7	623.0	**47.9**
breast	32.8	**16.5**	23.9
cleveland	155.6	117.2	**76.6**
crx	412.37	512.8	**159.6**
pima indians	229.1	440.9	**36.3**
german	233.6	297.0	**77.3**
glass	**38.0**	614.2	40.3
glass2	**14.7**	50.4	19.0
heart	**33.6**	66.4	34.2
horse-colic	319.0	74.3	**46.1**
iris	11.9	7.0	**6.4**
vehicle	123.5	347.7	**72.2**

the fitness function, and this will be investigated in the future. Moreover, future directions also include parallelization of ESIA and its extension for FOL induction.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant HKBU2063/98E.

Bibliography

[1] S. Augier, G. Venturini, and Kodratoff Y. Learning first order logic rules with a genetic algorithm. In *Proceedings of the First International Conference on Knowledge Discovery in DataBases*, pages 21–26, Montreal, Canada, 1995.

[2] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Department of Information and Computer Sciences.

[3] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[4] K.A. De Jong, W. Spears, and D.F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13(2-3):155–188, 1993.

[5] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[6] E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 144–151, 1998.

[7] A.A. Freitas. A genetic algorithm for generalized rule induction. In R. Roy, editor, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 340–353. 1999.

[8] A.A. Freitas. On rule interestingness measures. *Knowledge-Based Systems*, 12(5-6):309–315, October 1999.

[9] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[10] A. Giordana and F. Neri. Search-intensive concept learning. *Evolutionary Computation*, 3(4):375–416, 1996.

[11] D.P. Greene and S.F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257, 1993.

[12] J. Hekanaho. DOGMA : A GA-based relational learner. In *Proceedings of the 8th Intentional Conference on Inductive Logic Programming*, pages 205–214. Springer Verlag, 1998.

[13] J.H. Holland. Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An AI Approach*, pages 593–623. Morgan Kaufmann, 1986.

[14] R. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[15] C.Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2-3):180–228, 1993.

[16] K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Conference on Machine Learning*, pages 249–256. Morgan Kaufmann, 1992.

[17] I. Kononenko. Estimating attributes: Analysis and extensions of RELIEF. In L. De Raedt and F. Bergadano, editors, *Proceedings of the European Conference on Machine Learning*. Springer Verlag, 1994.

[18] I. Kononenko. On biases in estimating the multivalued attributes. In *Proceedings of IJCAI*, Montreal, Canada, August 1995.

[19] J. Liu and W. Li. A concept learning method based on a hybrid genetic algorithm. *Science in China (Ser. E)*, 28(4):488–495, 1998.

[20] Surry P.D. and Radcliffe N.J. Inoculation to initialise evolutionary search. In Fogarty T.C., editor, *Evolutionary Computing: AISB Workshop*, pages 268–285. Springer, Brighton, U.K., 1996.

[21] M.A. Potter and K.A. De Jong. The coevolution of antibodies for concept learning. In *Proceedings of the 5th International Conference on Parallel Problem Solving From Nature*, pages 530–539, Amsterdam, 1998. Springer Verlag.

[22] S. Sen, L. Knight, and K. Legg. Prototype based supervised concept learning using genetic algorithms. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 223–239. Springer, 1997.

[23] S. Smith. Flexible learning of problem solving heuristics through adaptive search. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 422–425, Karlsruhe, Germany, 1983.

[24] G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of the European Conference on Machine Learning*, pages 280–296, 1993.

[25] S. Wilson. Classifier systems and the Animat problem. *Machine Learning*, 2:199–228, 1987.