GA-based Rule Enhancement in Concept Learning*

Jukka Hekanaho

Turku Center For Computer Science and Åbo Akademi University
Lemminkäisenkatu 14 A
SF-20520 Turku, Finland
hekanaho@abo.fi

Abstract

We apply DOGMA, a GA-based theory revision system, to MDL-based rule enhancement in supervised concept learning. The system takes as input classification data and a rule-based classification theory, produced by some rule-based learner, and builds a second model of the data. The search for the new model is guided by a MDL-based complexity measure. The proposed methodology offers a partial solution both to the local minima trap of fast greedy learners, and to the time complexity problem of GA-based learners. As an example we show how the system improves rules produced by C4.5.

Introduction

GAs have been used in both propositional and relational learning, e.g. (DeJong, Spears, & Gordon 1993; Grefenstette, Ramsey, & Schultz 1990; Giordana & Neri 1996; Augier, Venturini, & Kodratoff 1995; Hekanaho 1995). GAs are, in general, successful in avoiding local minima and produce often near optimal solution. However, this is done under a heavy computational burden. In the other extreme of computational complexity we have fast, more or less greedy, concept learners, like (Quinlan 1993; Clark & Niblett 1989; Cohen 1995). These systems learn fairly good classification theories with only a fraction of the time required by GAs. However, a fast greedy learner might get trapped in local minima, which might sometimes be quite far from the global minimum.

In this paper we apply the GA-based theory revision system DOGMA (Domain Oriented Genetic MAchine) (Hekanaho 1996) to MDL-based theory revision of decision rules. To do so we first produce a classification theory using a rule-based learner. In the second step we use the first theory as a background theory in DOGMA. In this case we have selected C4.5Rules (Quinlan 1993) as the initial learner. DOGMA takes as input preclassified data and a rule-based theory of the data, and tries to build a new, enhanced theory, that more accurately models the same data. Unlike most theory revision systems, e.g. (Donoho & Rendell 1995; Ourston & Mooney 1990), we don't focus on the refinement of the theory, rather we build a completely new theory, using substructures from the initial theory.

The Rule Representation Language

The rule language $\mathcal{L}_1$ of DOGMA is a first-order language similar to the one used in the REGAL system (Giordana & Neri 1996) and to the VL_2 language of Michalski (Michalski 1983). A rule is expressed without its head as a conjunction of predicates like $P(x_1, \ldots, x_n, [v_1, \ldots, v_m])$, where x_i's are variables and $[v_1, \ldots, v_m]$ is a disjunction of constants v_i. The symbol $*$ may be used to collapse a set of values. For example, if a relation of distance rarely takes other values than 0, 1 and 2, we can define a predicate *dist* for values $[0, 1, 2, *]$.

The full language $\mathcal{L}_1$ can be restricted to *k-CNF rules.* A k-CNF rule allows maximally k values in the internal disjunctions. Thus formula (1) below is a k-CNF rule, for $k \geq 2$, but is not a 1-CNF rule.

The Language Template

The formulae in the language $\mathcal{L}_1$ are mapped into bitstrings using a *language template* Λ. Λ is the maximal conjunctive formula representable by the GA. Every predicate in Λ contains all possible values $[v_1, \ldots, v_k]$ (possibly with $*$). As an example consider the language template Λ in Figure 1. All concept description are obtained by deleting some part of Λ. For example

$$color(x, [r, w]) \wedge shape(x, [tr, *]) \qquad (1)$$

is obtained by deleting the size predicate and some internal disjunctions from the template. The formulae are mapped to bitstrings by setting each bit to 1 iff the corresponding value is present in the predicate in Λ. Thus a 0 in the bitstring means that the corresponding value isn't in the internal disjunction. Consequently, a substring corresponding to all the values of a predicate consisting of only 0's is illegal and is automatically rewritten as a substring of 1's. For example mapping formula (1) to a bitstring using the language template in Figure 1 produces 1010111101.

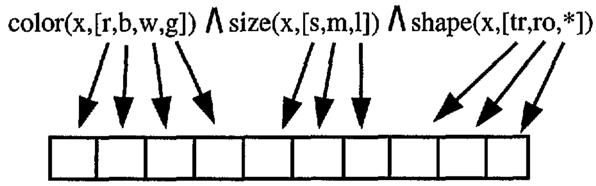

Figure 1: A language template and a bitstring

Background Knowledge

The background rules have the following structure:

$$c : h(x) \leftarrow p_1(x, [v_{11}, \ldots, v_{1i}]), \ldots, p_n(x, [v_{n1}, \ldots, v_{nj}])$$

where c indicates the concept class in which the rule may be used.

DOGMA filters the background rules through the language template Λ to produce *background chromosomes*. This is done by removing the head of each rule, as well as all predicates that are not a part of the Λ. Then it turns the remaining predicates to the same form as the corresponding ones in Λ. Now the rules are formulae in $\mathcal{L}_1$ and can be mapped to bitstrings.

As an example let Λ denote the language template in Figure 1 and consider the following rules:

$$h(x) \leftarrow color(x, [r, g]), size(x, [m]), shape(x, [tr, sq]).$$
$$h(x) \leftarrow color(x, [r, w]), size(x, [m]), position(x, [1, 2]).$$

Filtering these rules through Λ results in the following background chromosomes:

$$color(x, [r, g]), size(x, [m]), shape(x, [tr, *]).$$
$$color(x, [r, w]), size(x, [m]).$$

The Genetic Algorithm

DOGMA combines the hybrid Michigan/Pittsburgh methodology of JGA (Hekanaho 1995) with special stochastic operators, which induce knowledge from a background theory. The system supports two distinct levels with accompanying operators. The lower level has fixed length chromosomes, which encode rules and are manipulated by genetic operators. On the higher level the chromosomes are combined into genetic families, which encode classification theories. Background knowledge is incorporated into the learning process by background seeding, which creates new chromosomes from the background chromosomes.

DOGMA is a parallel program where each class can be run in parallel, and furthermore, each class may have several parallel GA-populations.

Speciation

In order to keep some structure in the learning process DOGMA uses speciation of the chromosomes. The chromosomes are divided into species according to which background chromosomes they may use. A species corresponds to a single background chromosome. The speciation of chromosomes is used in three ways. Firstly, it's used to control mating of chromosomes of different species. Secondly, speciation affects the GA through background seeding; each chromosome may use background seeding only from its own background chromosome. And finally, speciation is also used while merging chromosomes into families. The families are symbiotic, this means that chromosomes of the same species can't be merged into the same family.

Genetic operators

The genetic operators, working with genetic chromosomes, are *mating, mutation, crossover, seeding,* and *background seeding.* The crossover operators are inherited from REGAL. We use four different crossovers, the well-known two-point and universal crossovers as well as a generalizing and a specializing crossover. The last two operators are specifically designed keeping in mind the specific requirements of concept learning.

Given two parent bitstrings s_1 and s_2 the generalizing crossover first selects a set of predicates from Λ and locates the substrings corresponding to the selected predicates in s_1 and s_2. It then creates the offsprings by doing a bitwise or operation on the located bits of s_1 and s_2. The specializing crossover works like the generalizing one, but instead of a bitwise or it performs a bitwise and.

The seed operator generates a bitstring covering at least one random positive training example. Seeding works by generating a random bitstring, which is then changed minimally so that it covers the selected example. Background seeding is identical to seeding, except that it generates the initial bitstring by selecting random bits from a background chromosome.

Family operators

DOGMA follows the metaphor of competing families, thus fitness as well as selection and replacement are lifted to the level of families. In addition we use operators that break or merge families.

The *break operator* splits randomly a family into two separate families. The counterpart of break is *join,* that merges two families of symbiotic chromosomes into a single family. Because of the symbiotic requirements join puts at most one chromosome of each species into the new combined family. The operator is made slightly more sophisticated by adding directed joining and seeding into it. When a family is to be joined, a random uncovered positive example is chosen and join searches for a suitable family that covers this example. A family is suitable if it has a chromosome of a new species that covers the example. If no suitable family is found then a suitable single chromosome family is made through seeding or background seeding.

In addition to join, we use another family building operator called *make-family.* This is a global operator that builds a family by selecting useful chromosomes, of different species, from the population.

We can freely mix the family operators with the genetic ones. Currently, for each generation of the GA population, we apply first the genetic operators and then the family operators.

MDL-based Fitness

The new version of DOGMA is augmented with a MDL-based fitness function. The MDL principle (Rissanen 1989) advocates that the best theory for describing some data is the one that minimizes the description length of both the theory and the data. The description length associated with a theory T consists of a theory cost, i.e. the length of the encoding of T, and of the exception cost, which is the encoding of the data falsely classified by T. According to the MDL principle the best theory T is the one that has the minimal total description length of the theory and the exceptions. The encoding scheme we use is the one used by Quinlan in (Quinlan 1995).

The MDL-based fitness function

To turn the MDL measure into a fitness function we compare the MDL against the total exception length, i.e. against the MDL of an empty theory.

$$f_{MDL} = 1 - \frac{MDL}{MAX\ LENGTH}$$

However, we cannot use this fitness measure directly, the reason being that the encoding scheme treats all exceptions, i.e. both false positives and false negatives, equally. While this is perfectly reasonable in comparing final theories, it is not necessarily so while learning theories. In our case we need to be able to give a realistic fitness also to theory fragments. A good theory fragment is a partial theory that has a relatively low false positive rate. Such a theory fragment is a good building block, in the sense that it may be merged with other fragments to form a good complete theory.

Our solution to this problem is to use the MDL-based fitness when we are close to complete theories, and to gradually fall off to other fitness measures if the theories are too incomplete or inconsistent.

$$\begin{aligned} f = \quad & if\ (fprate \geq 1 \wedge tprate \geq 1)\ then\ f_{MDL} \\ & else\ if\ (tprate \geq 1)\ then\ f_{fp} \\ & else\ if\ (fprate \geq 1)\ then\ f_{tp} \\ & else\ \ f_{fp} * f_{tp} \end{aligned}$$

$$f_{fp} = \frac{f_{MDL} \times fprate + conf \times (1 - fprate)}{fprate}$$

$$f_{tp} = \frac{f_{MDL} \times tprate + conf \times (1 - tprate)}{tprate}$$

The MDL-based fitness is applied when the rate of false positives is better than in the background theory and the rate of true positives high enough. Otherwise the fitness is reduced by a consistency measure f_{fp} or by a completeness f_{tp}, so that the total fitness drops. The appropriate levels of $tprate$ and $fprate$ are derived directly from the background theory. $conf$ is the inverse of the pessimistic error rate used by Quinlan in (Quinlan 1993) to estimate the error rate of a single C4.5 rule or leaf. However, it could be any measure that gives a realistic fitness to a theory fragment as a building block.

Empirical Evaluation

In order to evaluate DOGMAs abilities to improve on classification rules, we apply DOGMA to rules generated by C4.5Rules in eight UCI problems, measuring the predictive accuracy, number of rules, and description length of the both initial and final classification theories. We measured also the response time T of DOGMA, defined as $T = max(T_{ij})$, where T_{ij} is the evaluation time of process j in class i. The time is expressed in minutes and comes from runs on a Sun Ultra 2 Model 2170. Note that all classes are treated independently and can be learned in parallel. For the reported experiments we have used one process per class, except for letter domain for which we used two processes per class. In all domains except in letter, the rule language of DOGMA was defined to be 1-CNF rules, since this is the default used by C4.5. For the letter domain we used the subset option of C4.5 and the full rule language of DOGMA.

As can be seen from Table 1 the final rules produced by DOGMA have consistently shorter description lengths than the corresponding C4.5 rules[1]. Although the improvement in accuracy is not statistical significant in a single domain, the results are at least indicative since the accuracy is improved in all domains except in Breast Cancer. The number of rules tends to drop, especially when there are many rules, although there is more variation than in the description lengths.

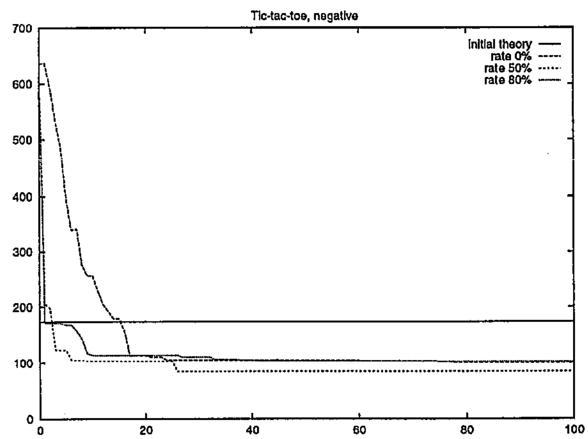

Figure 2: The MDL coding length of the best theory in each generation for different background seed rates in the Tic-tac-toe domain, class negative.

[1]The MDL for C4.5 in Congress voting and Lymphography are quite high, due to the fact the C4.5 doesn't always produce rules for all classes.

Domain	IA	FA	IR	FR	IL	FL	T	M
Breast cancer	70.3 (64.4, 76.2)	68.2 (58.4, 78.0)	10.7	10.4	434.6	432.6	7.0	CV
Chess	98.9 (98.1, 99.4)	99.4 (98.7, 99.7)	23.0	21.0	452.9	378.0	76.7	TT
Congress voting	95.8 (94.3, 97.3)	96.3 (95.4, 97.2)	6.0	7.9	431.7	220.5	1.9	CV
Letter	76.2 (75.2, 77.2)	77.0 (76.0, 78.0)	248.0	214.0	36561.9	36051.6	317.7	TT
Lymphography	77.7 (72.1, 83.3)	80.4 (75.6, 85.3)	9.7	12.2	226.1	187.3	2.1	CV
Splice junction	92.3 (90.5, 93.8)	94.5 (92.9, 95.8)	89.0	71.0	1847.7	1830.2	321.6	TT
Splice 12 positions	93.1 (91.4, 94.5)	94.6 (93.9, 95.8)	90.0	70.0	1605.6	1592.3	116.7	TT
Tic-tac-toe	98.5 (97.1, 99.9)	99.5 (99.0, 100.0)	22.0	20.0	288.9	175.5	10.3	CV

Table 1: Results from eight UCI domains. IA and FA stand for the test set accuracy percentages of the initial C4.5 theory and of the final theory. IR, FR, IL, and FL give the number of rules and description lengths of the initial vs. final theory. T column gives the response time of DOGMA. M is the evaluation method, where TT is a single train and test set and CV a 10-fold crossvalidation. For the accuracies we give also the 95% confidence intervals.

A background seed rate controls the average amount of bits that are copied from the background chromosomes. To show an example of the effect of different background seed rates we run DOGMA for 100 generations, with different background seed rates, in the Tic-tac-toe domain. The initial C4.5 rules are quite easily improved, see Figure 2, even when no background seeding is used. The best result is achieved with a 50 % background seeding rate. The 80 % rate does slightly worse, which suggests that the initial rules become too dominant in this case and hinder the search for new shorter theories.

Conclusions

We have described how to use the GA-based theory revision system DOGMA to enhance rule-based classification theories. In our methodology we first use a fast concept learner to produce a first approximation of the data, and then improve the result in DOGMA using a MDL-based measure, to gain a second approximation of the data. The methodology itself is of course not limited to C4.5 as a preprocessor or to MDL as a model selection method. We could in principal have used other rule learners and other fitness measures.

Acknowledgments

Thanks to the Ljubljana Oncology Institute, Slovenia, for the Breast cancer and Lymphography domains, and to Christopher Merz and Patrick Murphy for maintaining the UCI repository of machine learning databases.

References

Augier, S., Venturini, G., and Kodratoff, Y. 1995. Learning first order logic rules with a genetic algorithm. In *1st International Conference on Knowledge Discovery and Data Mining*, 21–26.

Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3:261–283.

Cohen, W. W. 1995. Fast effective rule induction. In *12th International Conference on Machine Learning*, 115–123.

DeJong, K. A., Spears, W. M., and Gordon, F. D. 1993. Using genetic algorithms for concept learning. *Machine Learning* 13:161–188.

Donoho, S. K., and Rendell, L. A. 1995. Representing and restructuring domain theories: a constructive induction approach. *Journal of Artificial Inteligence Research* 2:411–446.

Giordana, A., and Neri, F. 1996. Search-intensive concept induction. *Evolutionary Computation Journal* 3/4.

Grefenstette, J. J., Ramsey, C. L., and Schultz, A. C. 1990. Learning sequential decision rules using simulation models and competition. *Machine Learning* 5:355–381.

Hekanaho, J. 1995. Symbiosis in multimodal concept learning. In *12th International Conference on Machine Learning*, 278–285.

Hekanaho, J. 1996. Background knowledge in GA-based concept learning. In *13th International Conference on Machine Learning*, 234–242.

Michalski, R. S. 1983. A theory and methodology of inductive learning. In *R. S. Michalski and J. Carbonell and T. Mitchell (Eds.), Machine Learning: An AI Approach*, volume Vol. I, 83–134. Los Altos, CA: Morgan Kaufmann.

Ourston, D., and Mooney, R. 1990. Changing the rules: A comprehensive approach to theory refinement. In *1990 National Conference on Artificial Intelligence*.

Quinlan, J. R. 1993. *C4.5 Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. 1995. MDL and categorical theories (continued). In *12th International Conference on Machine Learning*, 464–470.

Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. River Edge, NJ: World Scientific.