Classifying Uncovered Examples by Rule Stretching

Martin Eineborg[1] and Henrik Boström[1,2]

[1] Machine Learning Group, Department of Computer and Systems Sciences,
Stockholm University/Royal Institute of Technology,
Electrum 230, Stockholm, Sweden,
{eineborg,henke}@dsv.su.se
[2] Virtual Genetics Laboratory,
171 77 Stockholm, Sweden
Henrik.Bostrom@vglab.com

Abstract. This paper is concerned with how to classify examples that are not covered by any rule in an unordered hypothesis. Instead of assigning the majority class to the uncovered examples, which is the standard method, a novel method is presented that minimally generalises the rules to include the uncovered examples. The new method, called Rule Stretching, has been evaluated on several domains (using the inductive logic programming system Virtual Predict for induction of the base hypothesis). The results show a significant improvement over the standard method.

1 Introduction

One major distinction between methods for induction of classification rules is whether they treat the hypothesis as an ordered or unordered set of rules. In the former case, there is no need for resolving classification conflicts among the rules, since the first applicable rule is used (such an hypothesis is commonly referred to as a decision list [14]). Furthermore, a decision list always includes a default rule at the end, which means that any example that may have passed through the previous rules without being covered will still be assigned a class. It should be noted that the standard inductive logic programming setting with two classes (positive and negative examples) and where a hypothesis is searched for that covers all positive examples but none of the negative, in fact is a special case of the former case, since it implicitly assumes that any example that is not covered by the rules should be classified as negative. In the case with unordered hypotheses, rules need to be generated for all classes and some strategy has to be adopted for resolving conflicts among the rules (e.g., [4]). Furthermore, it may very well happen that none of the rules is applicable when trying to classify new examples. A common strategy for handling such examples is to classify them as belonging to the majority class (e.g., [7]).

In this paper we present a new method for classifying examples that are not covered by any of the rules in an (unordered) hypothesis. The method, Rule

Stretching, is applied after the hypothesis has been induced, during the classification phase. Rule Stretching works by generalising the rules in a hypothesis to cover the previously uncovered examples. The method is not targeted at any special inductive logic programming system but is a general method for assigning classes to uncovered examples.

The paper is organised as follows. In the next section, we present a general algorithm for Rule Stretching. This algorithm is specialised in Section 3 with respect to a specific learning paradigm and a system called Virtual Predict, which is used for induction of base hypotheses. The setup of the experiments and the results are presented in Section 4. The work presented in this paper has some ideas in common with Analogical Prediction [10], which are discussed in Section 5. The paper ends with concluding remarks in Section 6. The reader is assumed to be familiar with basic concepts of logic programming [8].

2 Rule Stretching

Examples that are not covered by an unordered hypothesis are usually classified as belonging to a default class (usually the majority class). The work in this paper is instead based on the idea that unordered rules of an induced hypothesis can be 'stretched out' to cover previously uncovered examples. Rule Stretching is used in the following way:

1. Induce an unordered set of rules using an inductive logic programming system
2. Classify new examples using the induced rules
3. Examples that are not covered by any of the rules are given to the Rule Stretching system for classification

A new, more general, hypothesis that is ensured to cover a previously uncovered example, can be formed by computing the minimal generalisation of the example and each rule in the hypothesis. The rules of the new hypothesis has to be evaluated since, by generalising the rules to cover the example, the accuracy of the rules may have been changed (important conditions could have been removed by the generalisation making it possible for a rule to cover more examples of other classes than it did before the generalisation).

An example on how Rule Stretching works is illustrated in Figure 1 where there are two classes a and b, two rules, $R1$ and $R2$, and an uncovered example denoted '?'. In the picture to the left, the two rules and their coverage can be seen as well as the uncovered example. In the picture to the right the two rules have been generalised so that they cover the unclassified example and the class of the example can be determined by, for example, selecting the most probable class for the most accurate rule, which in this case means class a. Note that when generalising rule $R2$ it covers not only examples of class b but also two examples of class a (thus decreasing its accuracy).

A general algorithm for Rule Stretching is presented in Figure 2. The algorithm takes a hypothesis H, background knowledge B, examples E, an uncovered

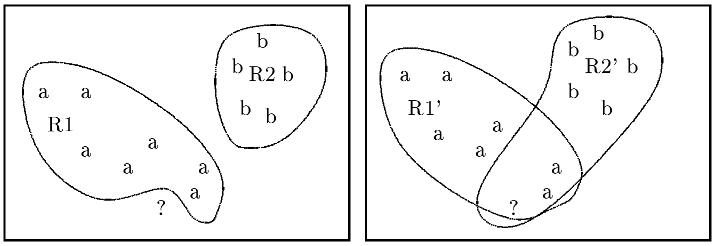

Fig. 1. Rule Stretching

example $e \in E$ such that $H \wedge B \not\models e$, and returns a class label. Three functions are used by the algorithm: the *minimal_generalisation* function, which returns the minimal generalisation of a rule and an example, the *coverage* function, which takes a rule and a set of examples and returns the number of examples of the different classes it covers, and the *classify* function, which returns a class label given a set of pairs of generalised rules and their coverage.

Input: hypothesis H, background knowledge B, examples E, an uncovered example e
Output: a class label c

1. $H' = \{r' \mid r \in H \wedge r' = minimal\_generalisation(r, e)\}$
2. $V = \{(r, v) \mid r \in H' \wedge v = coverage(r, E)\}$
3. $c = classify(V)$

Fig. 2. Rule Stretching Algorithm (general version)

3 Rule Stretching using Least General Generalisations

The general version of the Rule Stretching algorithm should be specialised with respect to the theoretical foundation of the inductive logic programming system that is used. In this study we consider the framework of the system Virtual Predict, which is described in Section 3.1. A special version of the Rule Stretching algorithm that takes advantage of the properties of this particular system is given in Section 3.2.

3.1 Virtual Predict

Virtual Predict [1] is an inductive logic programming system that is a successor of Spectre 3.0 [2]. The system can be viewed as an upgrade of standard decision tree and rule induction systems in that it allows for more expressive hypotheses to be generated and more expressive background knowledge (i.e., logic programs) to be incorporated in the induction process. The major design goal has been to achieve this upgrade in a way so that it should still be possible to emulate the standard techniques with lower expressiveness (but also lower computational cost) within the system if desired. As a side effect, this has allowed the incorporation of several recent methods, such as bagging, boosting and randomisation, that have been developed for standard machine learning techniques into the more powerful framework of Virtual Predict.

Like its predecessor, Virtual Predict uses resolution as a specialisation operator [3]. This means that each rule generated by the system is the result of repeatedly applying resolution to some overly general clause. For reasons of efficiency, the system internally represents rules on the same format as it represents proofs of examples, namely as *derivation terms*.

A derivation term is a term on the form $c_i(t_1, \ldots, t_n)$, where c_i is an identifier of some input clause in the derivation of the rule (or the proof of the example), and $t_1, \ldots, t_n$ are derivation terms corresponding to the sub-derivations (or sub-proofs) for the n literals in the body of the clause c_i.

For example, given the following overly general theory:

```
(c1) target(Size,Shape,Weight):-
        size(Size), shape(Shape), weight(Weight).
(c2) size(A):- A = small.
(c3) size(A):- A = medium.
(c4) size(A):- A = large.
(c5) shape(A):- regular(A).
(c6) shape(A):- irregular(A).
(c7) regular(A):- A = circular.
...
(c15) weight(A):- A = low.
```

the proof of the example `target(small,circular,low)` would be represented by the derivation term $c1(c2, c5(c7), c15)$. The derived rule

```
target(Size,Shape,Weight):-
        Size = small, regular(Shape), weight(Weight).
```

is represented by the derivation term $c1(c2, c5(\_), \_)$.

It should be noted that a derivation term for a derived rule typically is non-ground, while a derivation term for a proof of an example always is ground[1]. By finding the proofs of all examples in advance of the induction process and by representing the proofs together with derived rules as derivation terms, the coverage check of a derived rule and an example is reduced to unification, i.e. no theorem proving is needed. This has led to an order of magnitude speedup in Virtual Predict compared to its predecessor.

[1] It should also be noted that some built-in predicates (such as arithmetic predicates) need special treatment, but this falls outside the scope of the paper.

3.2 A Specialised Rule Stretching Method

Since Virtual Predict represents the rules of a hypothesis as terms it is possible to compute the minimal generalisation of a rule and the proof of an example by computing the least general generalisation [12].

Definition 1. *An atom c is a generalisation of atoms a and b if there exists substitutions θ_1 and θ_2 such that $c\theta_1 = a$ and $c\theta_2 = b$.*

Definition 2. *A generalisation c for two atoms a and b is a least general generalisation (lgg) if for each other generalisation c_i of a and b there exists a substitution θ_i such that $c = c_i\theta_i$.*

A new more specific version of the general Rule Stretching algorithm, was formed by replacing the *minimal_generalisation* function with a function, *lgg*, that computes the least general generalisation of a rule and the proof of an example. Furthermore, the *classify* function was replaced with a function, *use_best_rule*, that given a set of evaluated rules returns the class of the rule that has the highest accuracy (with Laplace-correction). The lgg version of the Rule Stretching algorithm can be seen in Figure 3.

Input: hypothesis H, background knowledge B, examples E, an uncovered example e
Output: a class label c

1. $H' = \{r' \mid r \in H \wedge r' = lgg(r, e)\}$
2. $V = \{(r, v) \mid r \in H' \wedge v = coverage(r, E)\}$
3. $c = use\_best\_rule(V)$

Fig. 3. Rule Stretching Algorithm (lgg version)

4 Empirical Evaluation

A number of experiments were conducted in order to find out whether the Rule Stretching method performs better than choosing the majority class for uncovered examples. In all of the experiments the base hypotheses were induced by Virtual Predict. In Section 4.1, we describe how Virtual Predict was configured and the domains used. The experimental results are given in Section 4.2.

4.1 Experimental Setting

There are a number of parameters that can be set when defining learning methods in Virtual Predict, allowing a very wide range of methods to be defined, including the emulation of standard techniques, such as decision tree induction and naive Bayes classification. The parameters include the search strategy to use (separate-and-conquer or divide-and-conquer), optimisation criterion (e.g. information gain), probability estimate (e.g. m estimate), whether an ordered or unordered hypothesis should be induced, inference method (how to apply the resulting hypothesis), post-pruning (using e.g. an MDL criterion or a prune set) as well as ensemble learning methods (bagging, boosting, and randomisation). There are also a number of parameters that have to be set when defining experiments in Virtual Predict, such as what experimental methodology to use (e.g. n-fold cross validation).

The parameters and their values were in this study set according to Table 1. A covering (separate and conquer) approach to rule induction was used together with incremental reduced error pruning [5], by which a generated rule is immediately pruned back to some ancestor in the derivation sequence (the pruning criterion was in this experiment set to accuracy on the entire training set, but other options in Virtual Predict include accuracy on a validation set and most compressive ancestor).

In case an example was covered by more than one rule this conflict was resolved by computing the most probable class using naive Bayes, by maximising the following expression:

$$P'(C|R_1 \wedge \ldots \wedge R_n) = P(C)P(R_1|C) \ldots P(R_n|C) \tag{1}$$

where C is a class and $R_1 \ldots R_n$ are the rules that cover a particular example. It should be noted that in case a particular example is not covered by any rule, maximising the above expression leads to assigning the example the most probable class *a priori*, which is the standard method for classifying uncovered examples.

Table 1. Virtual Predict settings

PARAMETER	VALUE
STRATEGY	Separate and Conquer
OPTIMISATION CRITERION	Information Gain
PROBABILITY ESTIMATE	M Estimate, with M=2
MEASURE	Information Gain
INCREMENTAL REDUCED ERROR PRUNING	Most Accurate on Training Set
INFERENCE METHOD	Naive Bayes
EXPERIMENT TYPE	10-Fold Cross Validation

Rule Stretching was tested on the seven problems that can be seen in Table 2 along with some statistics about the number of classes, the distribution of the classes, the number of examples that were not covered by the base hypothesis, and the total number of examples in the domain. Four of the domains were collected from the UCI Machine Learning Repository: the Balance Scale Database, the Car Evaluation Database, the Congressional Voting Records Database, and the Student Loan Relational Database. The data for the problem of recognising illegal positions on a chess endgame with a two kings and a rook, KRKI, and the problem of predicting the secondary structure of proteins (described in [11]) were available from the web page of the Machine Learning Group at the University of York. The Alzheimers toxicity domain was available from Oxford University Computing Laboratory. The domain was described in [6].

In the secondary protein structure domain the hypothesis was restricted to looking at properties for only three positions at a time (i.e., the predicate `alpha_triplet/3` was used).

Table 2. Domain statistics

Domain	Classes	Class Distribution (%)	Uncovered Examples (%)	Total Examples
Balance	3	7.84; 46.08; 46.08	19.04	625
Car	4	3.76; 3.99 22.22; 70.02	3.36	1728
House Votes	2	38.62; 61.38	2.53	435
KRKI	2	34.2; 65.8	2.6	1000
Alzheimers tox.	2	50; 50	6.54	886
Secondary Protein Structure	2	43.4; 56.6	27.61	1014
Student Loan	2	35.7; 64.3	5.4	1000

4.2 Experimental Results

Two methods for classifying uncovered examples were compared in the seven domains: selecting the majority class and Rule Stretching. The same base hypotheses were used in conjunction with the two methods, and these were produced by Virtual Predict using the settings shown in the previous section. In all but one of the domains, 10-fold cross validation was employed. Due to long computation time for the secondary protein structure domain only a single run was made, using a single training and test set.

The null hypothesis was that Rule Stretching is not more accurate than selecting the majority class. The results of using the two methods is shown in Table 3. One can see that in seven out of seven cases, Rule Stretching results in more accurate classifications than when assigning uncovered examples the majority class. The one sided binomial tail probability of this number of successes, given that the probability of success is 0.5, is approximately 0.0078, which allows for a rejection of the null hypothesis at the 1% level.

5 Related Work

The idea of including the example to be classified in the formation of the hypothesis is shared with Analogical Prediction [10]. Analogical Prediction uses background knowledge, training examples, and the example to be classified to

Table 3. Results

Domain	Majority Class	Rule Stretching
Balance	76.32%	84.64%
Car	93.98%	94.16%
House Votes	94.25%	95.4%
KRKI	97.1%	99.4%
Alzheimers Tox.	88.26%	90.18%
Secondary Protein Structure	57.99%	61.14%
Student Loan	92.2%	92.7%

form a hypothesis which is used to classify the example. For every new example that is to be classified, a new hypothesis is formed. This leads to a different behavior when classifying examples than the normal setting of inductive logic programming does.

The main difference between Rule Stretching and Analogical Prediction is that Analogical Prediction forms a new hypothesis for every example that is to be classified whereas in Rule Stretching a previously induced hypothesis is used to classify examples that it covers, and rules of the hypothesis are minimally generalised to cover the remaining examples. This results in that Rule Stretching only evaluates as many candidate rules as there are rules in the original hypothesis, while Analogical Prediction performs a costly search for each example to be classified, typically evaluating a large number of candidate rules.

6 Concluding Remarks

A novel method for classifying examples that are not covered by an unordered hypothesis has been presented. The method, Rule Stretching, is based on the assumption that a more accurate classification can be made by generalising rules of a base hypothesis to cover the uncovered examples than using the standard method of assigning the examples to the majority class. The experiments, in which the inductive logic programming system Virtual Predict was used for the induction of a base hypothesis, showed that Rule Stretching performs significantly better than the standard method.

There are several directions for future research. One is to alter the least general generalisation version of the Rule Stretching algorithm, by replacing the *use_best_rule* function with some other, more elaborate, function. For example, all of the generalised rules could contribute to the decision of the correct class label to return, by using naive Bayes to find the most probable class given all of the generalised rules and their coverage. Another alternative is to replace the *classify* function with a CN2 type of function [4].

Another interesting direction for future research would be to formulate a version of Rule Stretching for some other system, such as Progol [9]. Since rules and examples are not represented as terms in Progol, it is not possible to compute the minimal generalisation by computing the least general generalisation.

Instead, rules in a hypothesis could be stretched out to include uncovered examples by using relative least general generalisation [13]. However, one major drawback of using relative least general generalisation (compared to computing the least general generalisation of a pair of atoms as done in this study) is that the computational cost is significantly higher.

Yet another direction for future research would be to relax the condition in Rule Stretching that each rule is generalised minimally. One possibility is to allow the system to search in the lattice formed by the rule to be stretched and the most general rule. This would however be significantly more costly than the current approach. Another possibility would be to let the generalisation process continue after having classified an uncovered example by computing the minimal generalisations of the generalised hypothesis and a new uncovered example and stop this process only when the accuracy of the rules in the hypothesis significantly decreases.

References

1. H. Boström. Virtual Predict User Manual. Virtual Genetics Laboratory, 2001.
2. H. Boström and L. Asker. Combining divide-and-conquer and separate-and-conquer for efficient and effective rule induction. In *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634, pages 33–43. Springer-Verlag, 1999.
3. H. Boström and P. Idestam-Almquist. Induction of logic programs by example-guided unfolding. *Journal of Logic Programming*, 40(2–3):159–183, 1999.
4. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, pages 151–163, Berlin, 1991. Springer Verlag.
5. W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann, 1995.
6. R.D. King, A.Srinivasan, and M.J.E. Sternberg. Relating chemical activity to structure: an examination of ilp successes. *New Generation Computing*, 13(3–4):411–433, 1995.
7. W. Van Laer, L. De Raedt, and S. Dzeroski. On multi-class problems and discretization in inductive logic programming. In *Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems*. Springer-Verlag, 1997.
8. J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.
9. S. Muggleton. Inverse entailment and Progol. *New Generation Computing Journal*, 13:245–286, 1995.
10. S. Muggleton and M. Bain. Analogical prediction. In *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634, pages 234–244. Springer-Verlag, 1999.
11. S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5:647–657, 1992.
12. G. D. Plotkin. A note on inductive generalisation. *Machine Intelligence 5*, pages 153–163, 1970.
13. G.D. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press, 1971.
14. R.L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.