TRIPPER: Rule Learning Using Taxonomies

Flavian Vasile, Adrian Silvescu, Dae-Ki Kang, and Vasant Honavar

Artificial Intelligence Research Laboratory,
Department of Computer Science, Iowa State University,
Ames, IA 50011, USA
{flavian, silvescu, dkkang, honavar}@cs.iastate.edu

Abstract. In many application domains, there is a need for learning algorithms that generate accurate as well as comprehensible classifiers. In this paper, we present TRIPPER - a rule induction algorithm that extends RIPPER, a widely used rule-learning algorithm. TRIPPER exploits knowledge in the form of taxonomies over the values of features used to describe data. We compare the performance of TRIPPER with that of RIPPER on benchmark datasets from the Reuters 21578 corpus using WordNet (a human-generated taxonomy) to guide rule induction by TRIPPER. Our experiments show that the rules generated by TRIPPER are generally more comprehensible and compact and in the large majority of cases at least as accurate as those generated by RIPPER.

1 Introduction

Knowledge discovery aims at constructing predictive models from data that are both accurate and comprehensible. Use of prior knowledge in the form of taxonomies over attribute values offers an attractive approach to this problem.

Several authors have explored the use of taxonomies defined over attribute values to guide learning. Zhang and Honavar developed a Decision Tree [8] and a Naive Bayes [9] learning algorithm that exploit user-supplied feature value taxonomies. Kang et al [2] introduced WTL, Word Taxonomy Learner for automatically deriving taxonomies from data and a Word Taxonomy-guided Naive Bayes (WTNBL-MN) algorithm for document classification. Michalski [7] has proposed a general framework of attributional calculus that can be seen as an alternative way of representing rules containing abstractions. Additional references to related work can be found in [9,11]. Against this background, we present a rule induction method that exploits user-supplied knowledge in the form of attribute value taxonomies to generate rules at higher levels of abstraction, named TRIPPER (Taxonomical RIPPER). We report results of experiments that demonstrate the promise of the proposed approach on a widely used benchmark data set (the Reuters text classification data set [10]).

2 Method

RIPPER (*Repeated Incremental Pruning to Produce Error Reduction)*, was proposed by Cohen [1]. It consists of two main stages: the first stage constructs an initial ruleset

W.K. Ng, M. Kitsuregawa, and J. Li (Eds.): PAKDD 2006, LNAI 3918, pp. 55 – 59, 2006.

using a rule induction algorithm called IREP* [4]; the second stage further optimizes the ruleset initially obtained. These stages are repeated for k times. IREP*[1] is called inside RIPPER-k for k times, and at each iteration, the current dataset is randomly partitioned in two subsets: a growing set, that usually consists of 2/3 of the examples and a pruning set, consisting in the remaining 1/3. These subsets are used for two different purposes: the growing set is used for the initial rule construction (the rule growth phase) and the pruning set is used for the pruning (the rule pruning phase). IREP* uses MDL[5] as a criterion for stopping the process.

The rule growth phase: The initial form of a rule is just a head (the class value) and an empty antecedent. At each step, the best condition based on its information gain is added to the antecedent. The stopping criterion for adding conditions is either obtaining an empty set of positive instances that are not covered or not being able to improve the information gain score.

The rule pruning phase: Pruning is an attempt to prevent the rules from being too specific. Pruning is done accordingly to a scoring metric denoted by v^*.

IREP* chooses the candidate literals for pruning based on a score v^* which is applied to all the prefixes of the antecedent of the rule on the pruning data:

$$v^*(rule, prunepos, prunenef) = \frac{p - n}{p + n} \tag{1}$$

where p / n denote the total number of positive / negative instances covered by the rule. The prefix with the highest v^* score becomes the antecedent of the final rule.

Before introducing **TRIPPER**, it is helpful to formally define a taxonomy:

Taxonomy: *Let S = {v1, v2, ... vn} be a set of feature values. Let T be a directed tree where children(i) denotes the set of nodes that have incoming arrows to the node i. A node i is called leaf if it has no children. A taxonomy Tax(T,S) is a mapping which assigns to a node i of the tree T a subset S' of S with the following properties:*

$$Tax(T,S)(i) = \bigcup_{j \in children(i)} Tax(T,S)(j) \tag{2}$$

$$Leaves(T) = S \tag{3}$$

1. TRIPPER(G) - improvement at rule growth phase: Introducing the taxonomical knowledge at the rule-growth phase is a straightforward process we call **feature space augmentation.** The augmentation process takes all the interior nodes of the attribute value taxonomy and adds them to the set of candidate literals used for the growth phase.

2. TRIPPER(G+P) - improvement at rule pruning phase: A more general version of feature selection than pruning is abstraction: in the case of abstraction, instead of casting the problem as a matter of preserving or discarding a feature, we are able to choose from a whole range of levels of specificity for the feature under consideration.

The effect on the resulting rule can be observed in the following example:

[original rule] - *(rate = t) and (bank = t) and* ***(dollar = t)*** *=> is_interest*
[pruned rule] - *(rate = t) and (bank =t) and* ***(any_concept = t)*** *=> is_interest*
[abstracted rule] - *(rate = t) and (bank = t) and* ***(monetary_unit= t)*** *=> is_interest*

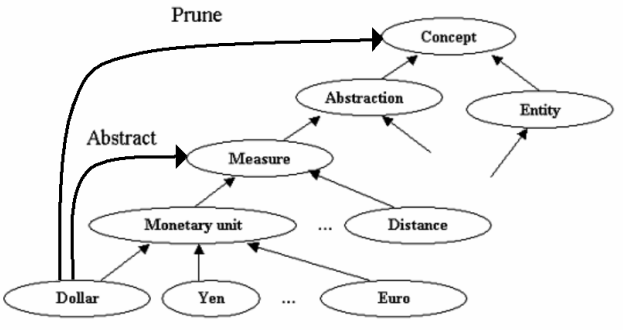

Fig. 1. Taxonomy over a set of nouns. Pruning and abstraction on a taxonomy.

Example 1: Variants of a classification rule for the class "interest"

The algorithm **Prune_by_abstraction** (fig.2.) uses exactly this idea to incrementally search for useful abstractions for the literals in the suffix to be pruned according to the v^* score of the rule prefixes.

```
Prune-by-abstraction(Rule,PruneData)

PrunedRule=PruneRule(Rule,PruneData)
Score=v*(PrunedRule,PruneData)
PrunePos=GePrunePos(PrunedRule), Level=0
While(improvement)
          Improvement=false, Increase(Level)
          For j:=PrunePos to size(Rule)
            AbstrRule=PrunedRule
            For i:=j to size(Rule)
               Literal=Rule(i)
               AbstrRule:=AbstrRule^Abstract(Literal,
               Level)
            If(v*(AbstrRule, PruneData)>Score)
               Update(Score)
               WinRule=AbstrRule, Improvement=true
Return WinRule
```

Fig. 2. Prune by Abstraction pseudocode

3 Experiments

Experimental setup: Experiments were performed on the benchmark dataset Reuters 21578 using the *ModApte* split [10] of training and testing data. Following the experimental setup used in [6], only the ten biggest classes in the dataset were used. As in [6], only the 300 best features were used as inputs to the classifier. The experiments compare RIPPER with TRIPPER (G+P) . The text-specific taxonomies used for our experiments on the Reuters dataset comes from WordNet[3], using only the hypernimy relation that stands for "isa" relation between concepts.

Results: Our experiments show that: (a) TRIPPER (G+P) outperforms, or matches RIPPER in terms of *break-even point* on the Reuters dataset (Table 3-1) in a majority (8 out of 10) of classes; (b) TRIPPER generates more abstract (and often more comprehensible) rules than RIPPER: Table 3-2 shows some of the abstract literals discovered to be important for 3 of the 10 classes. Furthermore, the rules generated by TRIPPER(G+P) are often more concise than those generated by RIPPER (results not shown) [11].

Table 3-1. Comparison of performance (break even point) of TRIPPER and RIPPER using WN

Class	Acq	Corn	Crud	Earn	Grn.	Inter	Mon	Ship	Trd.	Wht.
Trip.	**86.3**	**85.7**	**82.5**	**95.1**	87.9	**71.5**	**70.4**	**80.9**	58.9	**84.5**
Ripp.	85.3	83.9	79.3	94	**90.6**	58.7	65.3	73	**68.3**	83

Table 3-2. Abstract literals from WordNet

Class subject	**Abstract literals**
Crude Oil	assets, chemical_phenomenon, chemical_element, financial_gain, macromolecule, magnitude_relation, process, worker
Money, Foreign Exchange	artifact, assets, businessperson, document, institution, location, medium_of_exchange, measure, organization, signal, social_ event, solid
Trade	assembly, assets, calendar_month, change_of_magnitude, mass_unit, outgo, signal

The usefulness of abstraction is confirmed by the prevalence of abstract literals in almost all the rules of every ruleset. Both of the phases (growth and pruning) generated improvements (results not shown) [11], lending empirical support for the idea that both of the extensions are useful.

4 Conclusions

TRIPPER is a taxonomy-based extension of the popular rule-induction algorithm RIPPER [1]. The key ingredients of TRIPPER are: the use of an augmented set of features based on taxonomies defined over values of the original features (WordNet in the case of text classification) in the growth phase and the replacement of pruning, as an overfitting avoidance method, with the more general method of abstraction guided by a taxonomy over the features. The experiments briefly summarized in this paper show that TRIPPER generally outperforms RIPPER on the Reuters text classification task in terms of break-even points, while generating potentially more comprehensible rule sets than RIPPER. It is worth noting that on the Reuters dataset, TRIPPER slightly outperforms WTNBL [2] in terms of break-even points on 7 out of 10 classes.

The additional computation cost of TRIPPER is small when compared with RIPPER, consisting in an additional multiplicative factor that represents the height of

the largest taxonomy, which in the average case scales logarithmically with the number of feature values.

References

1. Cohen, W. W.: Fast effective rule induction. *Proceedings of International Conference on Machine Learning,* Lake Tahoe, CA. (1995)
2. Kang, D.-K., Silvescu, A., Zhang, J., Honavar, V.: Generation of Attribute Value Taxonomies from Data for Data-Driven Construction of Accurate and Compact Classifiers, *Proceedings of the 4th IEEE International Conference on Data Mining,* Brighton, UK. (2004)
3. Fellbaum, C: WordNet, An Electronic Lexical Database. *The MIT Press.* (1998)
4. Fürnkranz, J., Widmer, G: Incremental reduced error pruning. *Proceedings of International Conference on Machine Learning*. New Brunswick, NJ. (1994)
5. Quinlan, J. R.: MDL and categorical theories. *Proceedings of International Conference on Machine Learning,* Lake Tahoe, CA. (1995)
6. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: *AAAI-98 Workshop on Learning for Text Categorization*. (1998) 3-5.
7. Michalski, R. S.: Attributional Calculus: A Logic and Representation Language for Natural Induction, *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA. (2004)
8. Zhang, J., Honavar, V.: Learning decision tree classifiers from attribute value taxonomies and partially specified data. *Proceedings of International Conference on Machine Learning*, Washington, DC. (2003)
9. Zhang, J., Honavar, V.: AVT-NBL 2004: An algorithm for learning compact and accurate naive bayes classifiers from feature value taxonomies and data, *Proceedings of the Fourth IEEE International Conference on Data Mining,* Brighton, UK. (2004)
10. Apte, C., Damerau, F., Weiss Sholom, .M.: Towards language independent automated learning of text categorization models. *SIGIR '94*, Springer-Verlag New York, Inc. (1994) 23-30.
11. Vasile, F, Silvescu, A, Kang, D.-K., Honavar V.: TRIPPER: Rule learning using taxonomies, Tehnical Report ISU-CS-TR, Department of Computer Science, Iowa State University, Jan.2006. (Publicly available at http://www.cs.iastate.edu/~flavian/tripper_long.pdf)