Random Relational Rules

Grant Anderson and Bernhard Pfahringer

Department of Computer Science, University of Waikato, Hamilton, New Zealand

1 Introduction

Exhaustive search in relational learning is generally infeasible, therefore some form of heuristic search is usually employed, such as in FOIL[1]. On the other hand, so-called stochastic discrimination provides a framework for combining arbitrary numbers of weak classifiers (in this case randomly generated relational rules) in a way where accuracy improves with additional rules, even after maximal accuracy on the training data has been reached.[2] The weak classifiers must have a slightly higher probability of covering instances of their target class than of other classes. As the rules are also independent and identically distributed, the Central Limit theorem applies and as the number of weak classifiers/rules grows, coverages for different classes resemble well-separated normal distributions. Stochastic discrimination is closely related to other ensemble methods like Bagging, Boosting, or Random forests, all of which have been tried in relational learning[3, 4, 5].

2 Method

The Sparrow (Stochastic Production and Aggregation of Relational Rules) algorithm operates on two-class problems, and produces one set of first-order rules for each class. Unlike bagging or boosting, it neither resamples nor reweights the training set. Rules are generated fully randomly by adding literals to a partial clause in a manner similar to Foil, but without the immediate coverage computation. To determine which of the randomly generated rules should be added to a ruleset, Kleinberg's criteria of "Enrichment" and "Uniformity" are employed. Enrichment is a rule-level quality: a rule is enriched for a particular class, if it covers a greater proportion of the instances of that class than it does of the instances of the other class.

$$\text{A rule is enriched if } \frac{\#covered_{target}}{\#total_{target}} > \frac{\#covered_{other}}{\#total_{other}} \tag{1}$$

Uniformity is a ruleset-level quality - a uniform ruleset covers the training instances as evenly as possible. To obtain uniformity, enriched rules are generated in small batches, then the most uniformity-preserving non-zero subset of that batch of rules is added to the respective class' rule-set.

Sparrow produces two sets of rules, one for each class. Because the rulesets are constructed independently, there is no guarantee that the coverage distribution of one ruleset will be mirrored in the other. This means that the raw

Algorithm 1 Pseudocode for the Sparrow algorithm

```
for Each class do
  while Number of rules is less than the minimum do
    while Number of rules in batch is less than the minimum do
      Generate a rule
      if Rule is enriched then
        Add rule to rule batch
      end if
    end while
    Calculate the most uniform subset of rules in the current rule batch
    Add those rules to the ruleset
  end while
end for
```

proportions of rules in each ruleset that cover a test instance cannot be directly compared to determine a prediction. Therefore a transformation must be applied to produce compatible predictors for both rulesets. Two ratios which make the proportions comparable have shown good performance – each ruleset's average coverage across all training instances (AC) and the per-ruleset mean of each class's average coverage on the training instances (MAC).

$$AC_{ruleset}(Instance) = \frac{proportion\ of\ rules\ that\ cover\ Instance}{mean\ coverage\ of\ training\ instances} \tag{2}$$

$$MAC_{ruleset}(Instance) = \frac{proportion\ of\ rules\ that\ cover\ Instance}{mean\_coverage_{class\ A} + mean\_coverage_{class\ B}} \tag{3}$$

3 Results

An evaluation of Sparrow on several datasets has been conducted, using multiple ten-fold cross-validations – Mutagenesis (with and without regression-unfriendly instances), Musk1, Cancer (using only the Atom and Bond tables) and Diterpenes. As Diterpenes is a multiclass dataset and Sparrow currently only handles two-class problems, we set up three two-class datasets distinguishing between the three most numerous classes. Sparrow's results are compared to Foil 6.4 on these datasets using the default options. Foil 6.4 fails to produce rules on Musk1, but Ray and Craven[6] report results gained from a modified Foil version run on that dataset, and we compare to their results (marked by *).

4 Evaluation and Future Work

The AC and MAC methods both give reasonable results. MAC is consistently better on Mutagenesis, while the reverse is true for the three Diterpenes datasets.

Table 1. Accuracies and AUC for Sparrow and Foil

	Accuracy			*AUC*		
Dataset	*Sparrow AC*	*Sparrow MAC*	*Foil*	*Sparrow AC*	*Sparrow MAC*	*Foil*
$Muta_{RF}$	73.2 ± 1.2	76.8 ± 1.2	75.3 ± 3.4	0.815 ± 0.015	0.816 ± 0.015	0.791 ± 0.021
$Muta_{All}$	73.7 ± 1.2	75.1 ± 1.2	71.5 ± 0.9	0.772 ± 0.011	0.773 ± 0.012	0.749 ± 0.013
$Musk1$	82.6 ± 2.2	80.1 ± 4.1	$*$	0.896 ± 0.011	0.895 ± 0.011	$0.719*$
$Di_{52,54}$	71.5 ± 4.9	62.7 ± 1.4	45.7 ± 0.4	0.807 ± 0.044	0.688 ± 0.032	0.510 ± 0.007
$Di_{52,3}$	80.0 ± 0.7	66.7 ± 1.2	61.2 ± 1.5	0.899 ± 0.004	0.805 ± 0.004	0.647 ± 0.013
$Di_{54,3}$	79.8 ± 2.7	64.2 ± 3.1	51.4 ± 1.7	0.870 ± 0.030	0.732 ± 0.031	0.514 ± 0.020
Cancer	57.4 ± 0.6	55.3 ± 1.5	47.7 ± 4.5	0.642 ± 0.016	0.636 ± 0.018	0.500 ± 0.043

Foil has previously been reported to find highly specific rules and to fail to cover many examples[7], (and in our experiments on Diterpenes, Foil indeed displayed this behaviour). So not surprisingly Sparrow, which does not require its rules to be perfect, performs better here.

With Sparrow, we have demonstrated that it is possible for ensembles of randomly generated weak rules to be competitive with those produced by Foil's heuristic search. In the future, we plan to investigate the effects of ruleset size and batch size on accuracy, AUC, and runtime. We also want to compare to other relational rule learners like Progol[8], and also investigate randomization of their respective rule generation algorithms.

References

[1] Quinlan, J.R. Learning logical definitions from relations. Machine Learning 5 (1990), 239-266.

[2] Eugene M. Kleinberg. On the Algorithmic Implementation of Stochastic Discrimination. IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 22 No. 5 (2000), 473–490.

[3] Inês de Castro Dutra, David Page, Vítor Santos Costa, Jude Shavlik. An Empirical Evaluation of Bagging in Inductive Logic Programming. 12th Int. Conf. on Inductive Logic Programming (2002).

[4] Susanne Hoche, Stefan Wrobel. A Comparative Evaluation of Feature Set Evolution Strategies for Multirelational Boosting. Proc. 13th Int. Conf. on Inductive Logic Programming (2003).

[5] Celine Vens, Anneleen Van Assche, Hendrik Blockeel, and Sašo Džeroski. First Order Random Forests with Complex Aggregates. Proc. 14th Int. Conf. on Inductive Logic Programming (2004), 323–340.

[6] Soumya Ray and Mark Craven. Supervised versus multiple instance learning: an empirical comparison. Proc. 22nd Int. Conf. on Machine Learning (2005), 697–704.

[7] S. Džeroski, S. Schulze-Kremer, K.R. Heidtke, K. Siems, and D. Wettschereck. Applying ILP to Diterpene Structure Elucidation from ^{13}C NMR Spectra. Proc. 6th Int. Workshop on Inductive Logic Programming (1996), 41–54.

[8] S. Muggleton. Inverse entailment and Progol. New Generation Computing, Special issue on Inductive Logic Programming, 13(3-4):245–286, 1995.