Boosted Decision Graphs for NLP Learning Tasks

Jon D. Patrick and **Ishaan Goyal**
Basser Department of Computer Science
University of Sydney, NSW, Australia
{*jonpat,ishaan*}*@cs.usyd.edu.au*

Abstract

This paper reports the implementation of the AdaBoost algorithm on decision graphs, optimized using the Minimum Message Length Principle. The AdaBoost algorithm, which we call 1-Stage Boosting, is shown to improve the accuracy of decision graphs, along with we another technique which we combine with AdaBoost and call 2-Stage Boosting. which shows the greater improvement. Empirical tests demonstrate that both 1-Stage and 2-Stage Boosting techniques perform better than the boosted C4.5 algorithm. However the boosting has not shown a significant improvement for NLP tasks with a high disjunction of attribute space.

1 INTRODUCTION

In a wide variety of classification problems, boosting techniques have proven to be an effective method to significantly reduce the error of any weak learning algorithm. While the AdaBoost algorithm (Freund and Schapire, 1995) has been used to improve the accuracy of a decision tree algorithm (Quinlan and Rivest, 1989), which uses the Minimum Description Length Principle (MDL), little is known about it's effectiveness on the decision graphs.

This paper examines the application of the AdaBoost technique to the decision graph algorithm (Oliver and Wallace, 1991; Wallace and Patrick, 1993), which infers classification graphs from data by combining the Minimum Message Length Principle (MML) (Wallace and Boulton, 1968) with the recursive partitioning algorithm. In this paper we present two variants of the boosted decision graphs, which we call as 1-Stage Boosting and 2-stage Boosting respectively.

1.1 Decision Graphs

There have been a number of attempts to extend the representational power of the decision trees by allowing a node to have more than one parent. Oliver and Wallace (1991), introduced decision graphs which are generalizations of decision trees, having decision nodes and leaves. They optimize the decision graphs based on the MML Principle. The feature that distinguishes decision graphs from decision trees is that the former may also contain joins. A join is represented by two nodes having a common child, and this specifies that two subsets have some common properties, and hence can be considered as one subset. The manner in which the objects are allocated to leaf nodes in decision graphs is the same as decision trees. The decision graph offers an elegant solution to the replication and fragmentation problems faced by decision trees.

To discover classification graphs of short message length, the DGRAPH program of Oliver and Wallace (1991) starts with a single leaf at the root and grows a graph by repeatedly applying the modification that results in the greatest savings in the message length. Modifications considered either replace a leaf with a decision node or join two leaves together. The join probability, p_j is taken as an argument to the algorithm. The procedure halts when the best modification fails to reduce the message length of the resulting graph. To assist in preventing the greedy algorithm from becoming trapped in local message length minima, a lookahead parameter, l, is used to calculate message length savings over the next lower l levels of the tree under each possible modification.

2 BOOSTING DECISION GRAPHS - DGRAPH Parameters

The coding scheme for decision graphs has two important parameters, one is the p_j *probability of join*, and the second is *leaf node class purity, alpha*.

2.1 Probability of a Join

The probability of a join, p_j determines p_i the probability of leaf nodes, and, p_d the probability of decision nodes in the decision graph. The change in the value of probability of joins determines the way in which the decision graph is constructed. A low value for the probability of joins will encourage graphs with few join nodes, while a high value will encourage graphs with many join nodes.

As it is difficult to propose any fixed value for a probability of joins that will be suitable for a wide range of applications, we propose to use the data itself to estimate the value of p_j. Given a data set, we grow decision graphs for a range of values of p_j, say from 0.0, ..., 0.5. We select the graph with the smallest message length as being the best graph, and hence derive the estimate of p_j.

2.2 Prior Probability of Leaf Node Class Purity

The second parameter that effects the coding of decision graphs is the leaf node purity prior, *alpha*. The value of *alpha* determines how pure the leaf nodes will be. A heterogeneous class distribution in a leaf node has a uniform distribution where the value of the prior is one, however values of prior less than one places greater weight on more pure (homogeneous) distributions. As stated in Wallace and Patrick (1993), the data itself is used to estimate the leaf node purity prior. Having grown the full tree with say, $alpha = 1$, we find the best pruned form and message length for various values of *alpha*, and hence the best value of *alpha* is the one that generates the shortest message length.

We use both these parameters, that is, the optimal probability of a join, and the estimate of the leaf node purity prior together with the AdaBoost algorithm to improve the performance of the DGRAPH algorithm in our 2-Stage Boosting algorithm.

2.3 1-Stage Boosting

In our 1-Stage Boosting algorithm, we use AdaBoost to improve the prediction accuracy of our Decision Graphs. For all our experiments using 1-Stage Boosting, we use a fixed value of p_j= 0.2. Due to change in the distribution of weights on the training data, at the start of each new boosting step we re-adjust the value of *alpha* to 1 (assuming uniform distribution) and the best value of *alpha* is calculated according to the changed distribution, in the same manner as described in section 2.2.

2.4 2-Stage Boosting

Our 2-Stage Boosting algorithm, in each trial, finds the best probability of join, p_j and the best *alpha* according to the distribution of weights on the training data in that particular trial. In this way it finds the optimal number of join nodes required for data distribution in each particular boosting step and hence constructs the best graph in each trial, using the data itself to estimate the value of p_i and *alpha*.

2.5 Test Results

We have implemented our proposals and compared the results between the five algorithms - DGRAPH, 1-Stage Boosting, 2-Stage Boosting, C4.5 and boosted C4.5. The parameter T governing the number of classifiers generated was set at 10 as is conventional. All C4.5 parameters were set to their default values and we used pruned trees. Ten complete 10-fold cross-validations were carried out with each dataset.

All these five algorithms were evaluated on a representative collection of datasets from the UCI Machine Learning Repository. All the datasets show considerable diversity in size, number of classes, and number and type of attributes. 2-stage Boosted DGRAPH proved to be better than boosted C4.5 in 6 out of 7 tests by 2 to 15the seventh. 1-stage Boosted DGRAPH was better in 5 out of 7 of the tests.

2.6 Conclusions

Our trials over a diverse collection of datasets have confirmed that boosting improves the accuracy of the DGRAPH noticeably and the results also show that 2-Stage boosted DGRAPH is more accurate than the boosted C4.5.

3 RESULTS & CONCLUSIONS OF NLP SHARED TASKS

DGRAPH with 1 and 2-stage boosting and both C4.5 versions produced identical results. It is noticeable that they have all failed to perform better than the baseline for the Shared Task data sets. This is contrary to our experimental work that has shown boosting improves classification performance, for increased errors in data, and for increases in missing data. Likewise boosted DGRAPH has consistently shown better results than boosted C4.5 in our tests. These results confirm the extensive set of results produced by Daelemans, Van den Bosch and Zavrel (1999) that indicate memory based methods produce better results on NLP tasks than decision tree methods due to a high level of disjunctions in the attribute space.

Our experimental work on the nature of boosting indicates that it may be having a successful effect on classification because it compensates for poor selection of the prior probabilities of the data set, no matter whether that prior is implicit or explicit to the method. This suggestion is consistent with the known characteristic of language data that notionally follows Zipf's Law or at least a power series which at the moment are not usable in either DGRAPH or C4.5 in their current form. Unfortunately, usually no detailed determination of distribution characteristics is established in published experimental results for NLP tasks. We suggest that DGRAPH and C4.5 will both perform better on NLP tasks when they allow variation of the distribution characteristics of the classes and we have better probability distribution knowledge for this NLP data.

development	precision	recall	$F_{\beta=1}$
part 1	78.34%	36.82%	50.10
part 2	92.04%	48.57%	63.58
part 3	30.73%	13.57%	18.82

test	precision	recall	$F_{\beta=1}$
part 1	72.81%	38.47%	50.34
part 2	89.61%	45.39%	60.26
part 3	29.54%	13.45%	18.49

Table 1: Results obtained for the development and the test data set for the three parts of the shared task.

References

W. Daelemans, A. van den Bosch, and J. Zavrel. 1999. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, 34(1).

Y. Freund and R. Schapire. 1995. A decision theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning.*

J.J. Oliver and C.S. Wallace. 1991. Inferring decision graphs. In *Proceedings of Workshop 8 - Evaluating and Changing Representation in Machine Learning.* IJCAI-91.

J.R. Quinlan and R.L. Rivest. 1989. Inferring decision trees using minimum description length principle. *Information & Computation*, 80:227–248.

C.S. Wallace and D.M. Boulton. 1968. An information measure for classification. *Computer Journal*, 11:185–195.

C.S. Wallace and J.D. Patrick. 1993. Coding Decision Trees. *Machine Learning*, 11:7–22.