Boosted Decision Graphs for NLP Learning Tasks

Jon D. Patrick, Ishaan Goyal

Basser Department of Computer Science, University of Sydney, NSW, 2006, Australia

{jonpat, ishaan}@cs.usyd.edu.au, Fax +61-2-9351 3838

Abstract

This paper reports the implementation of DRAPH-GP an extension of the decision graph algorithm DGRAPH-OW using the AdaBoost algorithm. This algorithm, which we call 1-Stage Boosting, is shown to improve the accuracy of decision graphs, along with another technique which we combine with AdaBoost and call 2-Stage Boosting which shows greater improvement. Empirical tests demonstrate that both 1-Stage and 2-Stage Boosting techniques perform better than the boosted C4.5 algorithm (C5.0). The boosting has shown itself competitive for NLP tasks with a high disjunction of attribute space against memory based methods, and potentially better if part of an Hierarchical Multi-Method Classifier. An explanation for the effectiveness of boosting due to a poor choice of prior probabilities is presented.

Keywords: Decision Graph, Natural Language Processing, Boosting

1. INTRODUCTION

In a wide variety of classification problems, boosting techniques have proven to be an effective method to significantly reduce the error of weak learning algorithms. While the AdaBoost algorithm (Freund & Schapire, 1995) has been used to improve the accuracy of a decision tree algorithm (Quinlan & 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, DGRAPH-OW (Oliver & Wallace, 1991, Wallace & Patrick, 1993), which infers classification graphs from data by combining the Minimum Message Length Principle (MML) (Wallace & Boulton, 1968) with the recursive partitioning algorithm. In this paper we present two variants of the boosted decision graphs, DGRAPH-GP, which we denote 1-Stage Boosting and 2-stage Boosting.

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. They optimize their 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, and hence may prove more useful for NLP tasks give the distribution of classes across a very wide range of attribute values.

To discover classification graphs of short message length, the DGRAPH program of Oliver and Wallace (1991), hereafter DGRAPH-OW to distinguish it from the boosted DGRAPH algorithm we have developed, hereafter called DGRAPH-GP, 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 describing the tree. 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 p_j the *probability of join*, and the second is *alpha*, the *leaf node class purity.*

2.1. Probability of a Join

The probability of a join, p_j effects p_i the probability of leaf nodes, and, p_d the probability of decision nodes in the decision graph. A change in the value of the 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 & 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-GP algorithm in 2-Stage Boosting.

2.3. 1-Stage Boosting

In 1-Stage Boosting, 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-OW, 1-Stage Boosting, 2-Stage Boosting, C4.5 and boosted C4.5. All these five algorithms were evaluated on a representative collection of datasets from the UCI Machine Learning Repository.

The comparative results are given for the seven UCI data sets in Table 1. Ten-fold cross-validations were carried out with each data set. The boosting parameter T governing the number of classifiers generated was set at 10 for all these experiments (as is conventional). All C4.5 parameters had their default values and we used pruned trees. For DGRAPH-OW, 1-Stage and 2-Stage Boosting algorithms, we set the lookahead to 2 for all experiments and the probability of a join node is set to 0.2 for DGRAPH and 1-Stage Boosting algorithms. As our 2-Stage Boosting algorithm uses the data itself to estimate the probability of a join node, we do not need to specify the value of the probability of a join node. The first column of Table 1 below shows the name of the data set used, and the remaining columns show the average percentage accuracies of each data set against each of the five algorithms. All the datasets show considerable diversity in size, number of classes, and number and type of attributes. 2-stage Boosted DGRAPH-GP proved to be better than boosted C4.5 in 6 out of 7 tests by 2 to 15% and equal on the seventh. 1-stage Boosted DGRAPH-GP was better in 5 out of 7 of the tests.

The results from “Sick” data set seem to be anomalous, where boosting had no effect on the accuracy of C4.5, and DGRAPH-GP marginally decreases in accuracy when compared to DGRAPH-

Data Set	DGRAPH-OW	1 Stage boosted DGRAPH-GP	2 Stage boosted DGRAPH-GP	C4.5	Boosted C4.5
Tic-Tac_toe	91.30	95.00	97.10	86.03	86.40
Credit	86.45	85.81	87.10	85.30	85.33
Car	93.71	95.43	97.72	93.10	92.60
Vote	95.56	97.78	97.80	96.80	97.80
Nursey	96.46	97.23	99.92	96.20	96.90
Sonar	70.00	80.00	85.00	80.00	70.00
Sick	97.81	97.19	97.19	98.10	98.10

Table 1: Comparison of accuracy of all five algorithm on the UCI datasets.

OW. A look at the trees produced for the “Sick” data set suggest that the data contains a large number of insignificant attributes having no effect on classification and the trees are constructed using just three of the available twenty five attributes for both C4.5 and DGRAPH-OW. Also the distribution of the “Sick” data set is very uneven with almost 88% of training data containing items with one particular class. These two factors seem to effect boosting, which produces results with no or marginal change. This also suggests that implicit priors of C4.5 favour skewed class distributions or class distributions from either skewed or small attribute sets, whereas priors in DGRAPH favour uniform class distribution in data, which emphasises the importance of explicitly identifying priors. It may also be that DGRAPH-OW and DGRAPH-GP are disadvantaged using MML criteria that has to pay a much higher overhead cost than C4.5 for a large number of attributes that make no contribution to the final graph.

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

3. ANALYSIS OF NLP TASKS

DGRAPH-GP has been evaluated against a number of other classifiers on a common NLP task at the Computational Natural Language Learning Workshop (CoNLL, 2001) held at the 39th Annual Meeting of the Association for Computational Linguisitcs, in 2001. At that time Patrick and Goyal (2001) reported results on Clause Identification data using a window size of only zero and the results were not encouraging proving to be one of the poorest results amongst the six competing entries. Since that time we have conducted an extensive set of experiments on the shared task data which have significantly improved the results. However relative to the other methods our results will still be inferior because decision graph classifiers are unable to do any processing at the individual word level. Hence our results (Tables 2-6) can be considered as a minimum performance standard. A study of these results show significantly better results than the baseline tests with an improvement in F-value of 14 to 36 for the development test set, and 16 to 24 for the Final test set. The experiments were conducted incrementally, firstly the results were calculated using attributes of POS (p), clause boundaries (c) and the combined labels (pc) for word windows of 0, 1, 2 and 3. These results were scrutinized for zones where the best results occurred. This was invariably for the pc attribute set in windows of size 2 and 3. Further analyses showed these regions to perform better with 2 stage boosting yielding the best regions to direct further experiments where the Look-Ahead distance and the join probablity of the decision graph were varied. Invariably the optimal join probability was around 0.2.

Task	Precision	Recall	F	Baseline
Part1	94.84	87.33	90.93	54.58
Part2	80.12	83.03	81.55	67.42
Part3	78.19	67.63	72.53	52.17

Table 2: Results with Development Test Data

Task	Precision	Recall	F	Baseline
Part1	89.79	84.88	87.27	53.34
Part2	80.11	83.47	81.76	65.34
Part3	73.75	60.00	66.17	47.71

Table 3: Results with Final Test Data

Of the six systems that participated in the shared task for CoNLL-2001, two used boosting (Patrick and Goyal, 2001; Carreras and Marquez, 2001), Hammerton (2001) used a feed-forward neural network for predicting embedded clause structures, Tjong Kim Sang (2001) evaluated a memory-based learner while using different combinations of features describing items which needed to be classified, Molina and Pla (2001) applied a specialized Hidden Markov Model (HMM) to the shared task and Dejean (2001) used a symbolic learner ALLiS (Architecture for Learning Linguistic Structure) for predicting clause boundaries, and Hammerton only participated in Task 3.

.

Development - Task 1	Precision	Recall	F-value
Carreras & Mar.	95.77%	92.08%	93.89
Patrick & Goyal	94.84%	87.33%	90.93
Tjong Kim Sang	92.94%	86.87%	89.80
Molina & Pla	90.11%	88.80%	89.45
Dejean	94.08%	84.59%	89.08
Baseline	96.32%	38.08%	54.58%

Final Test - Task 1	Precision	Recall	F-value
Carreras & Mar.	93.96%	89.59%	91.72
Tjong Kim Sang	92.91%	85.08%	88.82
Molina & Pla.	89.54%	86.01%	87.74
Dejean	93.76%	81.90%	87.43
Patrick & Goyal	89.79%	84.88%	87.27
Baseline	98.44%	36.58%	53.34

Table 4: The performance of five systems while processing the development data and test data Task 1 of the shared task: identifying clause starts.

The three tasks consisted of determing the clause starts, clause ends and the combination of the two in the one sentence, or whole clauses. While Carreras and Marquez (2001), Tjong Kim Sang (2001) and Dejean (2001) concentrated on optimising data attribute features i.e. linguistic features, Patrick and Goyal (2001), Molina and Pla (2001) and Hammerton (2001) concentrated on Machine Learning methods. So, Patrick and Goyal (2001) give the best of the Machine Learning solutions.

Development - Task 2	Precision	Recall	F-value
Carreras & Mar.	91.27%	89.00%	90.12
Tjong Kim Sang	83.80%	80.44%	82.09
Patrick & Goyal	80.12%	83.03&	81.55
Molina & Pla.	78.65%	78.97%	78.81
Dejean	99.28%	51.73%	68.02
Baseline	96.32%	51.86%	67.42

Test - Task 2	Precision	Recall	F-value
Carreras & Mar.	90.04%	88.41	89.22
Tjong Kim Sang	84.72%	79.96%	82.28
Patrick & Goyal	80.11%	83.47%	81.76
Molina & Pla.	79.57%	77.68%	78.61
Dejean	99.28%	48.90%	65.47
Baseline	98.44%	48.90%	65.34

Table 5: The performance of five systems while processing the development data and the test data for Task 2 of the shared task: identifying clause ends.

Development - Task 3	Precision	Recall	F-value
Carreras & Mar.	87.18%	82.48%	84.77
Patrick & Goyal	78.19%	67.63%	72.53
Molina & Pla.	70.98%	72.31%	71.64
Tjong Kim Sang	76.54%	67.20%	71.57
Dejean	73.93%	62.44%	67.70
Hammerton	59.85%	55.56%	57.62
Baseline	96.32%	35.77%	52.17

Test - Task 3	Precision	Recall	F-value
Carreras & Mar.	84.82%	73.28%	78.63
Molina & Pla.	70.89%	65.57%	68.12
Tjong Kim Sang	76.91%	60.61%	67.79
Patrick & Goyal	73.75%	60.00%	66.17
Dejean	72.56%	54.55%	62.77
Hammerton	55.81%	45.99%	50.42
Baseline	98.44%	31.48%	47.71

Table 6: The performance of the six systems while processing the development data and the test data for Task 3 of the shared task: recognizing complete clauses

The results from all six systems given in tables 4, 5 and 6 (Tjong Kim Sang & Dejean, 2001) show that Carreras and Marquez (2001) clearly outperform the other five systems on all parts of the shared task. They converted the clause task into a set of binary decisions which they modeled with decision trees which are combined by AdaBoost. The system uses features which in some cases

contain relevant information about a complete sentence. It produces a list of clauses from which the ones with the highest confidence scores will be presented as output. They were the only ones to use input features that contained information of a complete sentence and the results tell that it was a good choice.

Carreras and Marquez (2001) were the most successful because they used an extensive variety of pre and post processing methods based on general linguistic knowledge, while their classifier was quite simple. This is contrasted with our method which used a relatively simple set of linguistic attributes but a sophisticated classifier. The comparison of the results points to the fact that capturing the linguistic features and constraints makes a greater contribution to accuracy than the differences between the classifiers. This result has motivated the two groups (Patrick and Goyal, 2001; Carreras and Marquez, 2001) to agree to collaborate on a reanalysis of the data combining both their methods.

The Patrick and Goyal (2001) results are most consistent with that of Tjong Kim Sang (2001) who used a memory based learner. This has advantage over the DGRAPH-GP method because it can use word level differentiation. We did a post processing of the graph from DGRAPH-OW, where the training items within each leaf node were grouped by orthography and the leaf nodes were further divided based on this orthography. The test files were then parsed on the resultant decision graph and items re-classified based on orthography, thereby giving it word level processing, de facto. In these experiments the F-values improved on average by about 0.2%. A better result may be obtainable if the word level analysis is integrated as a decision list procedure into the primary stages of decision graph selection and not treated as a post-processing stage. Such a combined Decision Graph and Decision List we would call an Hierarchical Multi Model Classifier (HMMC). We speculate from observations made from the erroneous classifications, that DGRAPH-GP when used in an HMMC will decrease its error rate by 25-40% and hence show significant superiority to these results.

Our results go some way to challenge 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. Decision Graphs are intended to overcome the disjunction problem and there is an indication in these results that it is being achieved. When DGRAPH-GP is adapted to be a HMMC and to allow word level processing we will be able to make a true comparison between the methods although there is every indication that decision graphs will at least equal the performance of memory based learners.

CONCLUSIONS

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 DGRAPH-GP, DGRAPH-OW or C4.5 in their current form, and hence they have performed better with the addition of boosting. Unfortunately, usually no detailed determination of distribution characteristics is established in published experimental results for NLP tasks. We suggest that DGRAPH-GP and C4.5 will both perform better on NLP tasks when they allow variation of the distribution models of the classes and we have better probability distribution knowledge for this NLP data.

REFERENCES

Carreras, X. and Marquez, L. (2001). Boosting Trees for Clause Splitting. In Proceedings of CoNLL-2001, pp 73-75. Toulouse, France.

Dejean, H. (2001). Using ALLiS for Clausing. Proceedings of CoNLL-2001, pp 64-66. Toulouse, France.

Daelemans, W., Van Den Bosch, A. and Zavrel, J. (1999). Forgetting Exceptions is Harmful in Language Learning. Machine Learning, 34, pp. 11-43, 1999.

Freund, Y & Schapire (1995). A decision theoretic generalization of on-line learning and an application to boosting. Proceedings of the Second European Conference on Computational Learning.

Hammerton, J. (2001). Clause Identification with Long Short-Term Memory. Proceedings of CoNLL-2001, pp 61-63. Toulouse, France.

Molina, A. and Pla, F. (2001). Clause Detecting using HMM. Proceedings of CoNLL-2001, pp70-72. Toulouse, France.

Oliver, J.J. & Wallace, C.S. (1991). Inferring decision graphs. Proceedings of Workshop 8 – Evaluating and Changing Representation in Machine Learning IJCAI-91.

Oliver, J.J. (1993). Decision graphs – an extension of decision trees. Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics, pages 343-350.

Patrick, J.D. and Goyal, I. (2001). Boosted Decision Graphs for NLP Learning Tasks. Proceedings of CoNLL-2001. pp 58-60. Toulouse, France.

Quinlan, J.R. & Rivest, R.L. (1989). Inferring decision trees using minimum description length principle. Information & Computation, 80, 227-248.

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

Tjong Kim Sang, E.F. (2001). Memory-Based Clause Identification. Proceedings of CoNLL-2001, pp67-69. Toulouse, France.

Tjong Kim Sang, E.F. & Dejan, H. (2001). Introduction to the CoNLL-2001 Shared Task: Clause Identification. Proceedings of CoNLL-2001. pp 53-57. Toulouse, France.

Wallace, C.S. & Boulton, D.M. (1968). An information measure for classification. Computer Journal, 11, 185-195.

Wallace, C.S. & Patrick, J.D. (1993). Coding Decision Trees. Machine Learning 11, pp 7-22.