A Combination of Bagging, Dagging and Boosting

10-601 Milestone 6 Project Report

Preeti Singh
Lane Centre of Computational Biology
School of Computer Science
Carnegie Mellon University
preetisi@andrew.cmu.edu

ABSTRACT

When a machine learning method is used by people with no special expertise in machine learning, it is important that the method be 'robust', in the sense that reasonable performance is obtained with minimal tuning for the problem at hand. In this paper, we propose a quantifiable measure of 'robustness', and describe a particular learning method that is robust according to this measure. Our method combined the strengths of a variety of classifiers together in an ensemble. This approach was motivated by the different advantages of boosting, dagging, bagging, and naive bayes, which together can make better decisions than any single learner. By forming an ensemble of these learners, each automatically tuned to the given dataset, we found a substantial improvement in robustness in most cases even over a boosted decision tree.

Keywords

Bagging, Dagging, Boosting, robustness

1. INTRODUCTION

With machine learning methods so often being used by individuals without significant ML expertise, it is becoming increasingly important for methods to be robust enough to have reasonable performance with minimal tuning for the problem at hand. The motivation behind this approach is to obtain such a robust method which performs well on a variety different datasets. This paper proposes such a measure of 'robustness', along with a method that is robust according to this metric. Our method combines the strengths of a variety of classifiers together in ensemble. Ensemble methods can be built by using different datasets, different training parameters, or by using a combination of different methods. The main advantage of using ensemble methods is exploiting the strengths of each base learner in a way that enhances the total accuracy and reliability of the method. We built our ensemble method by combining bagging, dagging, boosting, and naive bayes to take advantage of strengths of each of the approach. In our evaluation, we compare the robustness of this classifier against the strongest of its ensemble bases: one of the untuned boosting algorithms which was evaluated to have the highest robustness. By forming an ensemble of these learners, each automatically tuned to the given dataset, we found a substantial improvement in robustness for most datasets even against this boosted decision tree.

2. BACKGROUND

Our proposed method extends the work of Kotsianti et al., who showed that bagging, boosting, and dagging can be effectively combined in a voting ensemble for high accuracy on a variety of datasets[4]. Though the details of our method differ from this approach in a number of different ways, the core idea of combining these classifiers in a voting ensemble remains. The voting method we used was based on WEKA's weka.classifiers.meta.Vote class. Voting-based ensemble methods are a common and simple way to combine the results of different classifiers. A more sophisticated ensemble method wasn't used because each base classifier was itself an expensive ensemble method, and so combiner had to be simple and efficient. Several were considered (vote-counting, max, average), with average being ultimately chosen. The baseline algorithms that we found useful in our method included Bagging, Dagging, AdaBoostM1, MultiBoostAB, and Naive Bayes, described below and implemented in a machine learning toolkit called WEKA [3]. We also found a variety of weak learners to be appropriate for this technique which were not discussed in the original paper.

2.0.1 Bagging

Bagging (known as weka.classifiers.meta.Bagging in weka) is an ensemble technique that provides a high degree of robustness in a noisy setting. In this method, bootstrapping is first done to create variants of the original dataset. The base classifier is then trained on each each of these variants, and voting is done on each trained model for classificationcite [1]. We also used NNge (nearest-neighbor generalized exemplars, or weka.classifiers.rules.NNge within Weka) as the base learner. It is a nearest-neighbor-like algorithm which uses non-nested generalized exemplars, hyperrectangles that can be viewed as if-then rules [5]. A decision stump is another algorithm that was considered (and used in the original paper), which we found inferior to NNge for this classifier.

2.0.2 Dagging

Dagging (known as weka.classifiers.meta.dagging in weka) is an ensemble technique that provides a high degree of robustness in a noisy setting. In this method, the original data is divided into multiple folds, and then a copy of the base learner is built on each each data fold. During classification, a voting meta-classifier is used to combine the results of each sub-classifier to come to a majority-voted prediction. In our case we used J48 (known as weka.classifiers.trees.J48 in weka) as the base classifier for Dagging [6]. J48 is an implementation of a C4.5 decision tree, and it has the pruning confidence factor as a parameter option. The number of folds in Dagging is a classifier parameter option that we tuned.

2.0.3 Boosting

MultiBoostAB (known as weka.classifiers.meta.MultiBoostAB) is a combination of Boosting and Wagging. It can reduce the error rate of a base classifier by a significant amount. MultiboostAB [7] has advantage over AdaBoostM1 as it uses the power of boosting to reduce both variance and bias and wagging for reducing the variance significantly [2]. AdaBoostM1(known as weka.classifiers.meta.AdaBoostM1) is another boosting algorithm we used, which enhances the performance of some base learning algorithm and can also significantly reduce its error rate. This learning algorithm is called as 'weak learner', since the best parent classifier performance is typically associated with base learners with low accuracies (close to 0.5). These boosting algorithms run the weak learner on various distributions of the training data and then combine the results into a single composite classifier. We used a tuned J48 (as the weak learner for both).

2.0.4 Parallelization

Finally, we found that the training time for building and tuning all of these classifiers in the ensemble was too high to build a comfortable iteratively design, implement, and evaluate our method. To make the runtime of our classifier more practical for repeated runs (as we tried different tuning approaches), we made multi-threaded versions of the voting and parameter-tuning mechanisms. This substantially decreased the total run time of our classification on a multicore machine. It was also not too difficult, since voting and parameter-tuning use highly independent components which lend themselves well to parallelization. This kinds of algorithms would be a good fit for a distributed computation environment such as EC2, where an extremely high number of sub-classifiers and parameters could be farmed out as tasks to different compute nodes to quickly finish the build and evaluation process.

3. PROPOSED LEARNING METHOD

As previously mentioned, the learning technique proposed here mirrors at a high level the one proposed by Kotsianti et al., which describes combining bagging, dagging, and boosting in a voting ensemble. The paper found that while boosting algorithms typically perform better than bagging and dagging on noise-free data, the latter two are more robust in a noisy context [4]. Combining these techniques in a weighted voter ensemble combines their strengths, and results in higher accuracies across a variety of datasets. Indeed we found that Bagging and Dagging performed well when incorporated into our ensemble method. However, we found that boosting was especially effective on the evaluation datasets, and so incorporated two boosting methods in our ensemble: AdaBoostM1 and MultiBoostAB. Finally, we added the Naive Bayes classifier into the ensemble when we found a few datasets that our method performed badly on compared to this learning method. Incorporating it into the voting group reduced our worst-case and average error.

To do additional tuning on this ensemble mechanism, we individually tuned each sub-ensemble classifier where appropriate. In particular, we found that J48 worked well as a base classifier/weak learner for the relevant classifiers. Thus we used J48 as the base learner for these sub-ensemble classifiers, and internally tuned the J48 learner's pruning confidence factor based on its robustness performance on cross-validation of the parent classifier (for each dataset). This worked well with all base ensemble classifiers except Bagging, where NNge performed even better than a tuned J48. The robustness is defined as an error ratio of the classifier's error rate to that of an untuned naive bayes classifier. This ratio is used in any tuning context to evaluate the performance of a classifier. The tuned parameter (confidence factor) affects tree pruning, which helps the J48 tree by removing branches which are not sufficiently helpful. This can be very effective at preventing overfitting, reducing the total model size, and speeding up evaluation performance. Pruning confidence is used to calculate an upper bound on error rate at leaf/node. In our case, we varied the value of pruning confidence from 0.1 to 0.5 in 0.05-sized intervals. We used 10-fold cross-validation on the training instances in the process of tuning this parameter and selecting the lowest-error classifier. After tuning for MultiBoostAB and AdaBoostM1, we found a substantial performance increase. There was also a benefit for Dagging, but not as substantial. Besides the automatic tuning done on the confidence factor, there were also a few manually tuned parameters. The number of algorithm iterations for Bagging is one parameter we found useful to manually tune. We found this performed a bit better with a high number of iterations (15) than the default 10. AdaBoostM1's and MultiBoostAB performed a bit better with the number of iterations set as high as 25. With Dagging, we increased the number of folds to 15. Our voting method extended WEKA's Vote class and combined votes via an average of confidences. This is done by having each classifier in the ensemble generate a distribution of confidence values for each potential class. The confidence values for each class are summed up (i.e. averaged), which weights more highly the votes of classifiers which have a higher confidence in their predictions.

4. EXPERIMENTAL RESULTS

For our experimental setup, we compared the performance of our tuned ensemble classifier with a baseline. The baseline was an untuned version of AdaBoostM1 with J48 set as its base classifier. AdaBoostM1 had the highest measured performance of the classifiers within the ensemble (bagging, dagging, boosting, naive bayes), which made it an appropriate choice for a comparison with our proposed method. We used the M5a datasets, described below, to incrementally design our robust classifier, whereas M5b data was purely used for the evaluation of our classifier.

Name of ms5A Datasets	train size	test size	# features
anneal	602	296	39
audiology	75	151	70
Autos	137	68	26
Balance-scale	419	206	5
Breast-Cancer	192	94	10
Horse-Colic-test	247	121	23
Credit-rating	462	228	16
Diabetes	515	253	9
Glass	143	71	10
Heart	203	100	14
Hepatitis	104	51	20
Hypothyroid	2527	1245	28
Ionosphere	235	116	35
Labor	38	19	17
Lymph	99	49	19
Mushroom	5443	2681	23
Segment	1548	762	20
Sonar	139	69	61
Soybean	458	225	36
Splice	2137	1053	62
Vehicle	567	279	19
Vote	291	144	17
Vowel	663	327	14
Zoo	68	33	18

Table 1: Detailed information about the Datasets used in Development.

4.1 Datasets

Tables 1 and 2 show details about the datasets used for development and evaluation, respectively.

4.2 Results and Discussion

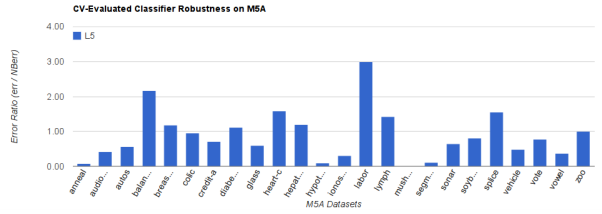

Figure 1: CV-Evaluated Clasifier Robustness on m5a

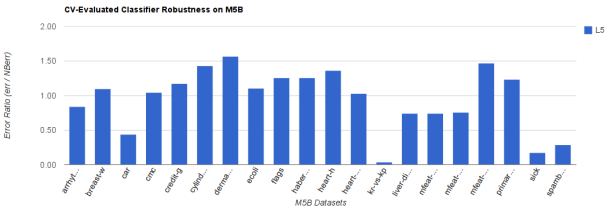

Figure 2: CV-Evaluated Classifier Robustness on m5b

After running L5 over M5b datasets, our mean error was 0.88 whereas the untuned AdaBoostM1 gave really high error

Name of ms5B Datasets	train size	test size	# features
arrhythmial	303	149	280
breast-w	468	231	10
cars	1158	570	7
cmc	987	486	10
credit-g	670	330	21
Cylinder-bands	362	178	40
dermatology	245	121	35
ecoli	225	111	8
flag	130	64	30
haberman	205	101	4
heart-h	197	97	14
Heart-statlog	181	89	14
kr-vs-kp	2141	1055	37
liver-disorders	231	114	7
mfeat-factors	1340	660	217
mfeat-fourier	1340	660	77
mfeat-karhunen	1340	660	65
primary-tumor	227	112	18
sick	2527	1245	30
Spambase	3083	1518	58

Table 2: Detailed information about the Datasets used in Evaluation.

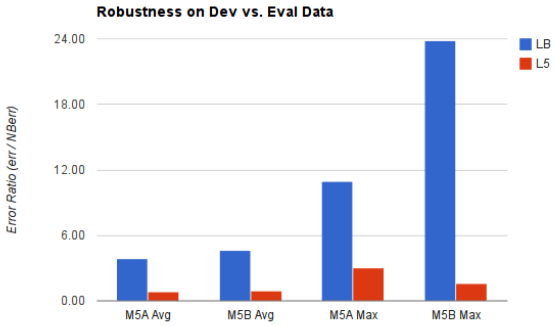

Figure 3: Comparision of Robustness between development and Evaluation dataset

ratio (1.57) as compared to that of L5. So we can see from the results that there is significant decrease in the mean error ratio after using L5 instead of LB(untuned AdaBoostM1).

Initially we utilized the Weka Experimenter, a tool used to run and evaluate classifiers on datasets, to identify promising classifiers to use within an ensemble. Our primary means of incrementally improving our classifier was by then running it locally on the old (first 12 of M5a) dataset and evaluating its accuracy via cross-validation. We would experiment with different methods and incorporate the ones that improved the results. Later in the process we also experimented on the new dataset (second 12 of M5a) on autolab, and incorporated a few enhancement (e.g. using Naive Bayes as one of the sub-ensembles in the voter).

Evaluation on M5a, the development set, drove a number of design decisions in developing our classifier. Thus, even though we want to design a classifier that is robust against

any dataset, the fact that we developed it based on its performance against a particular set means we somewhat overfit to that data. This is reflected in our results, as we have a somewhat worse robustness when running against our evaluation datasets (average .95 with M5b) than we do with our development datasets (average .88 with M5a)"

To evaluate robustness we tried to reduce our error rate as compared to the error rate of the Naive Bayes classifier. We define L5's robustness as this error ratio. So error ratio of L5 should be less than that of error ratio of NB classifier, in order to get robustness less than 1. So to make our method more robust as compared to Naive Bayes classifier we made a combination of Bagging, Dagging and Boosting, and it significantly decreased the error ratio also, hence increased the robustness of our method.

5. CONCLUSIONS

This paper described the important concept of 'robustness', a measure introduced to compare the performance of classifiers under minimal tuning for a variety of datasets. Robust methods in machine learning are especially relevant for use by individuals without special expertise in machine learning techniques and knowledge of appropriate tuning methodologies. This context is increasingly relevant in both industry and academic contexts, where machine learning can be applied to a wide variety of fields. The measures introduced here can evaluate methods on their robustness and help introduce new algorithms for robust classification. One such algorithm is proposed, which uses an ensemble of autotuned bagging, boosting, dagging, and naive bayes learners in a average-weighted voting combination. By tuning and evaluating the learners on their robustness, a final ensemble learner was designed and implemented which had significantly higher performance than the baseline of the best untuned classifier in the group. Such solid results are likely a result of effectively combining the strengths of very different classifiers; different classifiers (e.g. boosting) perform better than others (e.g. bagging/dagging) in different contexts, and so good results could be obtained by taking confidence-weighted combinations of these classifiers within an ensemble. By manually and automatically tuning the base/subbase classifiers on their robustness, we further managed to improved these results, with cross-validation helping avoid overfitting. Further improvements might be made by investigating several cases of bad performance and potentially incorporating a larger variety of base classifiers into the ensemble. This may result in better overall robustness for these contexts by reducing the average and max error ratios.

6. REFERENCES

[1] W. Cohen. Ensemble methods 1 lecture notes. Number 3. Machine Learning department,Carnegie Mellon University.

[2] Y. F. et al. Experiments with a new boosting algorithm. (7).

[3] G. Holmes, A. Donkin, and I. Witten. Weka: A machine learning workbench. *Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia.*, (2), June 2007.

[4] S. Kotsiantis and D. Kanellopoulos. Combining bagging, boosting and dagging for classification problems. *Knowledge-Based Intelligent Information and Engineering Systems*, (1).

[5] B. Martin. Instance-based learning: Nearest neighbor with generalization. (4), 1995.

[6] K. M. Ting and W. I. H. Stacking bagged and dagged models. in: Fourteenth international conference on machine learning, san francisco, ca. (5).

[7] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. 40.

7. APPENDIX

Project responsibility breakdown

Preeti Singh

- Background research on classifiers and ensemble method
- Background research on classifiers and ensemble methods
- Contributions to reports
- Some evaluation, tuning-related, and other misc. code implementations
- WEKA experimentation to find promising classifiers and methods
- Running evaluation and gathering data

7.0.1 What we learned

More details and familiarity with variety of ensemble methods. More details and familiarity with a variety of classification methods and their tunable options. Using a toolset like WEKA to experiment with and evaluate different classifiers and methods. Importance of the concept of robustness in evaluating learning methods. Various practical aspects of designing classification methods: Using cross-validation and/or test amd held-out sets. Experimenting with different tuning methods for best results. Considering the runtime of different methods and using techniques such as filters and parallelization to make them practical.

7.0.2 What we'd change

Overall a decent project. There were of course some data, grading etc. issues that could be improved upon in future semesters. Greater clarity and more sensible grading could also be given on milestone 5. In particular, don't encourage people to do the same kind of tuning they've done before. It should just be a verification step for report data. Greater care could also be taken to make improvement goals for earlier milestones more fair across the class. For instance, make an absolute baseline to pass when you are free to pick/tune/ensemble any choice of classifiers. It seems completely arbitrary and unfair to require a relative improvement to the previously assigned classifier when the two are completely unrelated and groups have very different performance on them.