Boosting Lite – Handling Larger Datasets and Slower Base Classifiers

Lawrence O. Hall[1], Robert E. Banfield[1], Kevin W. Bowyer[2], and W. Philip Kegelmeyer[3]

[1] Department of Computer Science & Engineering
University of South Florida
Tampa, Florida 33620-5399
{hall,rbanfiel}@csee.usf.edu
[2] Computer Science & Engineering
384 Fitzpatrick Hall, Notre Dame, IN 46556
[3] Sandia National Labs
Computational Sciences and Math Research Department
PO Box 969, MS 9159
wpk@sandia.gov

Abstract. In this paper, we examine ensemble algorithms (Boosting Lite and Ivoting) that provide accuracy approximating a single classifier, but which require significantly fewer training examples. Such algorithms allow ensemble methods to operate on very large data sets or use very slow learning algorithms. Boosting Lite is compared with Ivoting, standard boosting, and building a single classifier. Comparisons are done on 11 data sets to which other approaches have been applied. We find that ensembles of support vector machines can attain higher accuracy with less data than ensembles of decision trees. We find that Ivoting may result in higher accuracy ensembles on some data sets, however Boosting Lite is generally able to indicate when boosting will increase overall accuracy.

Keywords: classifier ensembles, boosting, support vector machines, decision trees.

1 Introduction

A boosted ensemble of classifiers is typically created to get higher accuracy for a particular type of base classifier on a particular data set. Here, we are going to instead examine how boosting may be used to deal with data sets that have a large number of examples and/or learning algorithms that are very time-consuming. In [1], Pavlov et. al. examined several methods for speeding up the learning process for support vector machines. One of those methods used boosting on a small subset of the data and reported good accuracies and significantly decreased learning times.

In this paper, we present a learning algorithm along the same lines as in [1], that we call Boosting Lite. The idea is to build the classifiers in the ensemble with small subsets of the available training data. The goal is to obtain approximately the same accuracy that you would get with a boosted ensemble or from a single classifier. Obviously, building each classifier with a small subset of the data will be much faster than using all the data. Hence, we look for a time savings and therefore the ability to better scale to large data sets.

In [2] Ivoting was shown to provide boosting-like results using a subset of the overall training data. This idea was further explored in a distributed system context in [3] where the accuracy was shown to be comparable to that of boosting. The Boosting Lite approach is much closer to standard boosting than is Ivoting. We will compare it to Ivoting results using eleven data sets. Also, for three data sets we compare Boosting Lite with SVMs as the base classifier.

2 Related Work

This work is motivated in part by the Boost-SMO algorithm introduced in [4] and again briefly discussed in [1]. Using linear support vector machines (SVM), between 1% and 90% of the training data was used in the creation of a boosted ensemble. The authors do not indicate how large any individual ensemble was, but indicate it was typically from 10 to 15 classifiers. They do indicate that they stopped when a specified maximum ensemble size was reached or the error of the current SVM was within ϵ of 0.5. Their version of a boosting algorithm was modified from standard boosting described in [5]. It was unclear whether a subset was initially selected or new subsets were selected from all the data for each classifier. From a personal communication [6], we found that a new subset of size x% was selected from all the re-weighted training data for each new classifier. On four data sets, they were able to show that it was possible to obtain the same accuracy as using an SVM-SMO model built on all the data by using as little as 2-5% of the data to train each classifier. The training time was from 3 to 400 times faster. However, the factor of 400 time result used 1% Boost-SMO which did not produce an ensemble classifier that was as accurate as using all the data (it was 0.6% less accurate in that case).

The Boost-SMO algorithm was not adapted for use with any other base classifier. The utility of an adaptation to decision trees is explored here. Work on Ivoting [2,3], which is a boosting-like approach that has been applied to decision trees, suggests that the adaptation may be useful. Ivoting works as follows. An example is randomly selected from the full data set with all examples having the same probability of being selected. If it is incorrectly classified by the classifiers in the existing ensemble that do not have it in its training set, then it goes into the training set for the next classifier. Otherwise, it goes into the training set with probability $\frac{e(k)}{1-e(k)}$, where e(k) is the error estimate at stage k.

The sizes of the training sets in the original work [2] were one of 100, 200, 400 or 800 examples. Experiments used five data sets that ranged from 2000 examples to 43,500 examples. So, relatively small subsets, from roughly 1.8% to 40% of the data, were used for each training set. Boost-SMO used a percentage of the total training data size. At 5% of the data, which resulted in good accuracy, training sets would range from 100 to 2175 examples which is slightly larger than used in [2].

A distributed version of Ivoting called DIvoting has also been introduced [3]. Accuracies were found to approximate Adaboost accuracies in many cases. That is, they were even higher than for a single classifier in many cases. Further, DIvoting also showed significant reductions in the time required to build an Adaboost-equivalent ensemble. Still, DIvoting was not faster than building a single decision tree. Decision trees typically require $O(fn\log n)$ time to build a single tree, where n is the number of training samples and f is the number of features. So the time to build an ensemble of size e, each using only $1/k$th of the data, is $O(ef\frac{n}{k}\log(\frac{n}{k}))$. So it scales up linearly with e, but scales down faster than linearly with k. As a result, it is *possible* to build ensemble of decision trees on $1/k$th of the data more quickly than a single tree on all the data, but it is unlikely unless there is a very large amount of data.

In this paper, we focus on what can be accomplished on a single processor. The previous work raises the question of whether an adaptation of Adaboost.M1W [7], which has been shown to be a highly accurate boosting algorithm for more than two classes, can be effectively applied to small subsets of data for other types of classifiers. In particular, we will look at decision trees. We will compare with Ivoting results in terms of accuracy and whether speedups are possible for decision trees. We will compare with SVMs on 3 data sets, also. The other question that we are investigating is whether Ivoting might be effective in providing fast training for support vector machines.

We compare Boosting Lite experimentally on 11 datasets to (a) regular Adaboost and to (b) a single classifier trained on all the training data, using decision trees. We compare Boosting Lite using decision trees to Boost-SMO using SVMs [1,4] on three data sets by using new experimental numbers for Boosting Lite generated for this paper and published numbers for Boost-SMO on three datasets. We compare Boosting Lite using decision trees to Ivoting [2,3] using decision trees on 7 datasets using new experimental numbers for Boosting Lite generated for this paper and published numbers or approximate numbers from a graph for Ivoting [3].

3 Boosting Lite

The algorithm we are naming Boosting Lite is based on Adaboost.M1W [7]. The differences in implementation are as follows. While all the training data is weighted, only a specified x% of it is chosen for the k^{th} training set. The examples are chosen probabilistically based on their weights, with replacement. So, it is possible for an example to appear in the training set more than one time. The algorithm is shown in Figure 1.

We grow the ensemble until a specified number of classifiers have been added or until the next classifier does not meet the standard boosting criterion for being added to the ensemble. Each of the classifiers in the ensemble is built on

```
Let I("statement") = 1 iff "statement" is true, and 0 otherwise.
Let $x_i$ be the $i^{th}$ example and $c_i$ denote the class of that example.

Input: Let $L = \{(x_1, c_1), \ldots, (x_n, c_n) : x_i \in X$ and $c_i \in C\}$ with $2 \leq |C| \leq n$ and $|X| = n$.
Let $h$ be a classifier that takes in an $x_i$ and generates a class in $C$.
Let $T$ be the number of boosting rounds

Initialize: $D_i(i) = \frac{1}{n}$.
For $t = 1, \ldots, T$ :
  – Train a classifier $h_t$ with a subset of size $S << n$ randomly sampled according to
    the weighted distribution, $D_t$, where $h_t$ should minimize the weighted error rate:
    $\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq c_i)$.
  – Set $\alpha_t = \ln(\frac{(|C|-1)(1-\epsilon_t)}{\epsilon_t})$.
  – Update D: $D_{t+1}(i) = D_t(i) e^{-\alpha_t I(h_t(x_i)=c_i)} / Z_t$
    where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ is a distribution)

Output: Set the final classifier $H(x)$: $H(x) = \arg\max_{c \in C} f(x, c) = \arg\max_{c \in C}(\sum_{t=1}^{T} \alpha_t I(h_t(x) = c))$
```

Fig. 1. Boosting Lite algorithm (a modified version of AdaBoost.M1W)

a rather small subset of the original data set. This enables each classifier to be built more quickly and creates the potential to scale to very large data sets.

The algorithm was applied with the OpenDT [8] decision tree learning algorithm, which is essentially a public domain re-implementation of C4.5 release 8 [9] without pruning, with the RainForest algorithm for evaluating attributes [10] and using the median method for handling missing values. The single tree results come from C4.5 with default pruning (CF=0.25).

4 Data sets

For comparison purposes we have used all but one of the data sets on which Ivote and SMO-Boost were evaluated in [1,4]. The Reuters data set was not used because we could not re-create the train/test data partitions. The data sets were used with the training and testing sub-divisions from [1,2,3] for comparison. We also include two other data sets which have a good number of examples, **pendigits** and **krk**. Table 1 shows the names of the data sets and their characteristics. They come primarily from the UCI repository [11] and statlog project [12].

The training set sizes range from a modest 4335 examples to 209,529 examples in 315 dimensions. For the **krk** data set that was not previously partitioned into a training and test set, we did a tenfold cross validation to get an average accuracy and time.

Table 1. Description of data sets attributes and size

Data Set	# attributes	# Train ex.	# Test ex.	# classes
adult	14	30162	1560	2
digit	256	7291	2007	10
dna	60	2000	1186	3
forest cover	54	98884	396257	2
Jones	315	209529	17731	3
krk	6	28056	0	18
letter	16	15000	5000	26
pendigits	16	7494	3498	10
shuttle	9	43500	14500	7
satimage	36	4335	2000	6
web	294	31932	4886	2

5 Experimental Results

There were three data sets, **adult**, **forest cover** and **web**, used in both the SVM experiments and our experiments. The support vector machine was used with sequential minimal optimization training (SMO) and 1% of the data. The first column of Table 2 show the accuracy from a single tree, and the next three show the accuracy of ensembles of 100 boosted trees using 1%, 5%, and 100% of the data. We calculate the difference in accuracy between a single support vector machine classifier and the boosted SVM ensemble, as well as the single decision tree accuracy and the 1% Boosting Lite ensemble. The gap between the two differences is given in the fifth column (BL - SVM). A positive number means the SVM ensemble is closer to the accuracy of a single SVM than the decision tree ensemble is to a single tree. For both types of classifier, the ensembles are less accurate than the single classifier. For each dataset, the difference in accuracy between an ensemble of decision trees and a single tree is more than for the ensemble of support vector machines and a single support vector machine. This suggests that support vector machines may be used to create an accurate ensemble classifier from less examples than are needed for a decision tree. This result is particularly striking for the **forest cover** data set where there is over an 11% difference between a single tree and an ensemble of boosted decision trees built with 1% of the data. For support vector machines, the difference between the ensemble and a single support vector machine was only 1%.

However, there are two caveats that arise when examining results from these three data sets. As can be seen in Table 2, boosting does not increase the accuracy for two of the three data sets when using all the data, at least for decision tree classifiers. This suggests that more data sets and more experiments are necessary. The data sets need to be evaluated to determine if they do in fact benefit from boosting.

It has been shown, for a relatively small number of data sets, that Ivoting can result in accuracies that are very close to the accuracy of a boosted ensemble for CART trees [2] and C4.5 trees [3]. Seven of those data sets are included in this

Table 2. Boosting Lite (BL) on decision trees compared with SVM's. The last column compares the gap in ensemble accuracy and single classifier accuracy with a positive number meaning the gap was lower for the SVM's.

Data Set	Single tree	BL-1%	BL-5%	Boosting	BL - SVM
adult	85.97	83.71	84.26	84.56	1.36
forest cover	76.91	65.58	75.06	81.58	10.16
web	76.16	75.1126	75.5014	74.5395	0.41

study. The best accuracy with Ivoting (as reported in the literature) came with a bite-size (training set size) of 800 examples, [2]. In Table 3, we show the results for a single tree and ensembles of size 100 created with 1, 5, and 100% of the data, as well as the accuracy for an Ivoted ensemble and the most comparable accuracy from Boosting Lite. The Ivote results are all from the CART decision tree classifier using bites of 800 examples with three exceptions. The **forest cover**, **pendigits**, and **Jones** results come from a C4.5 classifier and the **Jones** bite size is 818 [3]. We do not show the Ivote results for **krk** because one cannot obtain good accuracy when subsets are used, as discussed below.

Table 3. Boosting Lite (BL) on decision trees compared with Ivoting

Data Set	Single tree	BL-1%	BL-5%	Boosting	BL Comp	Ivote (800)
digit	86.85	89.44	92.68	94.37	94.22	94.5
dna	92.66	88.11	94.8567	94.941	95.62	96.2
forest cover	76.91	65.58	75.06	81.58	65.57	73.9
Jones	52.78	61.29	62.04	66.97	61.29	64.2
krk	80.73	45.90	69.66	89.17	-	-
letter	81.02	79.96	93.96	97.12	95.88	96.2
pendigits	92.11	94.68	97.03	97.43	96.91	96.97
shuttle	99.95	99.99	99.99	99.99	99.99	99.99
satimage	85.35	82.25	88.05	90.85	90.35	91.3

For Boosting Lite, the results come from a percentage of the training data which is comparable to that of bites with 800 examples and up to 500 trees in the ensemble. In each case, we set the number of trees for Boosting Lite to be equal to the number of iterations Ivoting required to converge to a stable accuracy [2,3]. With the exception of the two data sets with the same type of decision tree classifier, it is not possible to directly compare accuracies. However, we can look at how well the ensemble built with less data per tree did in approximating the accuracy of an ensemble built with all of the data available for each tree. For the **shuttle** data set all types of boosted classifiers are highly accurate. Boosting Lite is within 0.5% in accuracy for the **satellite** and **digit** data sets and exceeds the accuracy of the full ensemble for the **dna** data set.

For the **forest cover**, **Jones**, and **letter** data sets, a Boosting Lite ensemble results in a classifier that is at least 1% less accurate and as much as 16%

less accurate than Adaboost.M1W on all the data. Ivote (100 trees) is also less accurate than a fully boosted classifier on each of these data sets. However, it is almost equivalent to the results from the boosted CART ensemble for the **letter** data set. It is only 3% less accurate for **forest cover** and 2.7% less accurate for the **Jones** data set. Ivoting is using just over 0.5% of the data for the **forest cover** results and about 0.4% of the data for the **Jones** results. For both **forest cover** and **Jones**, Ivoting results in a higher accuracy ensemble classifier than Boosting Lite even though it uses less than half as much data.

The results with the **pendigits** data set do show a 3% accuracy difference between boosting at 1% and boosting with the full data set. Using 5% of the data results in an ensemble within about 0.4% of the boosted ensemble with all the data. **krk** involves a chess endgame and suffers significant accuracy degradations in a tenfold cross validation with less than 10% of the training data available and up to 500 trees. With 10% of the data available for each tree in the ensemble, it reaches 80.9% accuracy with 200 trees. The nature of this domain likely causes the requirement for more data.

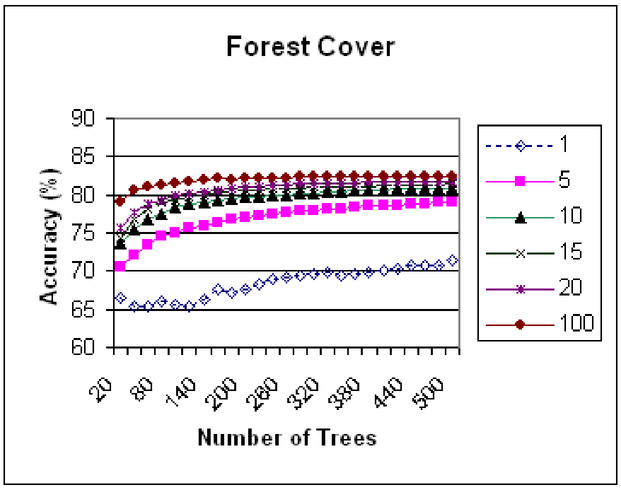

Fig. 2. Boosting Lite and Boosting accuracies on the **forest cover** data

To get an idea of the differences in classifier accuracy for different percentages of the training data and as more trees are added to a Boosting Lite ensemble, we show results for the **forest cover** data set in Figure 2. This data set shows greater differences for different sized training sets than most. The shape of the curves for 5% and greater is typical. It is also the largest data set with most of the data reserved for testing. You can see that with 1% of the data the ensemble never reaches the accuracy of boosting with all the data. With 5% the accuracy gets close and for 15% and 20% the accuracies are very close.

6 Discussion

A question that we were originally interested in was whether there was any possibility of speeding up decision tree construction for large data sets using Boosting Lite. OpenDT uses the RainForest algorithm [10] to avoid sorting data at internal nodes, making it quite a bit faster than older decision tree building algorithms such as C4.5. To look at timing, we took the time required to build a single tree in OpenDT and the time required to prune a C4.5 tree built on the same data and added the two times. This became the time to build a single tree. This can be compared with the time required to build 100 trees using 1% of the data. For more trees and/or more data, there is no time reduction from using the Boosting Lite approach. In fact even with 1% of the data, only for the **web**, **Jones** and **digit** data sets were there any speedups. In these cases, the speedups are 6.7, 1.3 and 1.4 times respectively. However, in general there is a slowdown for the data sets used here under the conditions described. Interestingly, the accuracies were higher than those obtained with a single tree for the **Jones** and **digit** data sets for which speedups were observed. The accuracy on the third (**web**) was higher than boosting with all the data, but 1% lower than a single decision tree.

Boosting Lite on 5% of the data provides accuracies which are greater than those obtained from one tree for all data sets where boosting results in a more accurate ensemble with two exceptions (**krk** and **forest cover**). The **krk** data set would probably not induce a data miner to try using subsets because the search space for chess is known to be large. So, Boosting Lite at 5% could be used to predict whether boosting will help on a data set.

The results indicate that while Boosting Lite is closer to classical boosting in operation than Ivoting, Ivoting seems often more accurate. Ivoting selects examples for the next training set randomly, but primarily admits them to the training set based on error from classifiers that do **not** have them in their training data set (this error rate is often called out of bag error). Ivoting's higher accuracy in some cases seems to indicate that using out of bag accuracy allows for better selection of training examples for the boosted classifiers. Given that Boosting Lite provides significant time savings for support vector machines, this raises the question of whether Ivoting might generally provide more accurate ensembles and maintain the time savings for slower classifiers like support vector machines and potentially neural networks.

7 Summary

In this paper we examined a variant of boosting that was originally introduced for speeding up support vector machine learning and adapted it to decision trees. We found that it appears that, when boosted, support vector machines can learn accurate models from smaller subsamples than decision trees. Boosting with subsampled data using decision trees as the base classifier was faster for 3 of 11 data sets than building a single tree on all the data. In order to have any chance of speeding up the learning on these data sets, the number of trees needed to be limited to 100 and the training set for each tree needed to be limited to 1% of the total training data set size. This is a function of the speed of decision trees and indicates that benefits require larger training sets than used here. Still, Boosting Lite on 5% of the data could be used to predict whether boosting with all the data will help on a data set.

For two of the three data sets used in the support vector machine work [4,1], we found that boosting with decision trees and all of the training data did not result in an increase in accuracy compared to a single decision tree. This result is somewhat anomalous, and suggests that further investigation on a broader range of datasets is needed. Also, we compared with an alternative boosting approach using subsamples, Ivoting. Ivoting was often able to result in a more accurate ensemble. This is likely due to using an error estimate from only trees which were not built with a potential training example in the selection process of examples for succeeding training data sets. Applying Ivoting to SVMs with large data sets has the potential to build accurate classifiers quickly, and it is an area for future research.

Acknowledgments. This research was partially supported by the Department of Energy through the ASCI PPPE Data Discovery Program, Contract number: DE-AC04-76DO00789.

References

1. D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 295–299, 2000.
2. L. Breiman. Pasting bites together for prediction in large data sets. *Machine Learning*, 36(1,2):85–103, 1999.
3. N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5:421–451, 2004.
4. D. Pavlov, J. Mao, and B. Dom. Scaling-up support vector machines using boosting algorithm. In *15th International Conference on Pattern Recognition*, volume 2, pages 219–222, 2000.
5. R. Schapire. A brief introduction to boosting. In *Proc. of the Sixteenth Intl. Joint Conf. on Artificial Intelligence*, 1999.
6. Dmitry Pavlov. Personal communication, 2006.
7. Gunther Eibl and Karl-Peter Pfeiffer. How to make AdaBoost.M1 work for weak classifiers by changing only one line of the code. In *Machine Learning: Proceedings of the Thirteenth European Conference*, pages 109–120, 2002.
8. Robert Banfield. *OpenDT*, 2005. http://opendt.sourceforge.net/.
9. J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
10. J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. *Journal of Data Mining and Knowledge Discovery*, 4(2-3):127–162, 2000.

11. C.J. Merz and P.M. Murphy. *UCI Repository of Machine Learning Databases.* Univ. of CA., Dept. of CIS, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.
12. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. Machine learning, neural and statistical classification, 1994. ftp://ftp.ncc.up.pt/pub/statlog/.