Racing Committees for Large Datasets

Eibe Frank, Geoffrey Holmes, Richard Kirkby, and Mark Hall

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{eibe, geoff, rkirkby, mhall}@cs.waikato.ac.nz

Abstract. This paper proposes a method for generating classifiers from large datasets by building a committee of simple base classifiers using a standard boosting algorithm. It permits the processing of large datasets even if the underlying base learning algorithm cannot efficiently do so. The basic idea is to split incoming data into chunks and build a committee based on classifiers built from these individual chunks. Our method extends earlier work by introducing a method for adaptively pruning the committee. This is essential when applying the algorithm in practice because it dramatically reduces the algorithm's running time and memory consumption. It also makes it possible to efficiently "race" committees corresponding to different chunk sizes. This is important because our empirical results show that the accuracy of the resulting committee can vary significantly with the chunk size. They also show that pruning is indeed crucial to make the method practical for large datasets in terms of running time and memory requirements. Surprisingly, the results demonstrate that pruning can also improve accuracy.

1 Introduction

The ability to process large datasets becomes more and more important as institutions automatically collect data for the purpose of data mining. This paper addresses the problem of generating classification models from large datasets, where the task is to predict the value of a nominal class given a set of attributes. Many popular learning algorithms for classification models are not directly applicable to large datasets because they are too slow and/or require too much memory. Apart from specialized algorithms for particular classification models, several generic remedies for the above problems have been proposed in the literature. They can be broadly classified into subsampling strategies [8, 13] and learning using committee machines [11, 3, 4, 12, 14]. Of these two strategies, the latter one appears to be particularly promising because (a) it does not require any data to be discarded when building the classifier, and (b) it allows for incremental learning because the model can be updated when a new chunk of data arrives.

The basic idea of committee-based learning for large datasets is to build a committee by splitting the data into chunks, learning a model from each chunk,

and combining the predictions of the different models to form an overall prediction. If the maximum chunk size is kept small, polynomial time algorithms can be applied to induce the individual models in a reasonable amount of time. Working with chunks also makes the process memory-efficient because a chunk can be discarded once it has been processed by the learning scheme.

In this paper we focus on using a boosting algorithm for building the committee machines. Boosting has the advantage that it can combine "weak" classifiers into a committee that is significantly more powerful than each individual classifier [5]. This is particularly advantageous in our application because the individual classifiers are built from relatively small samples of data and are therefore necessarily "weak."

The idea of using boosting for large datasets is not new, and appears to have been proposed first by Breiman [3]. The main contribution of this paper is a method for adaptively and efficiently pruning the incrementally built committee of classifiers, which makes the process computationally feasible for large datasets. It also makes it possible to choose an appropriate chunk size among several candidates based on "racing" the candidate solutions. This is important because the correct chunk size cannot be determined *a priori*. Apart from making the method practical, pruning also has the (desirable) side-effect that the resulting predictions can become more accurate.

This paper is structured as follows. In Section 2 we present our method for constructing committees on large datasets. We start with a naive method that does not perform any pruning and then move on to a more practical method that incorporates a pruning strategy. Finally, we discuss how the resulting committees are raced. Section 3 contains experimental results on a collection of benchmark datasets, demonstrating the importance of choosing an appropriate chunk size and using a pruning strategy. Section 4 discusses related work on combining classifiers built from chunks of data. Section 5 summarizes the contributions made in this paper.

2 The Algorithm

We first describe the basic algorithm—called "incremental boosting"—that generates the committee from incoming chunks of data. Then we explain how incremental boosting can be modified to incorporate pruning. Finally we present the racing strategy for pruned committees built from different chunk sizes.

2.1 Incremental Boosting

Standard boosting algorithms implement the following basic strategy. In the first step a prediction model is built from the training data using the underlying "weak" learning algorithm and added to the (initially empty) committee. In the second step the weight associated with each training instance is modified. This two-step process is repeated for a given number of iterations. The resulting

committee is used for prediction by combining the predictions of the individual models.

Boosting works well because the individual models complement each other. After a model has been added to the committee, the instances' weights are changed so that instances that the committee finds "difficult" to classify correctly get a high weight, and those that are "easy" to classify get a low weight. The next model that is built will then focus on the difficult parts of the instance space instead of the easy ones (where difficulty is measured according to the committee built so far). This strategy generates a diverse committee of models, and this diversity appears to be the main reason why boosting works so well in practice.

It turns out that boosting can also be viewed as a statistical estimation procedure called "additive logistic regression." LogitBoost [6] is a boosting procedure that is a direct implementation of an additive logistic regression method for maximizing the multinomial likelihood of the data given the committee. In contrast to AdaBoost and related algorithms it has the advantage that it is directly applicable to multiclass problems. It jointly optimizes the class probability estimates for the different classes and appears to be more accurate than other multi-class boosting methods [6]. For this reason we chose it as the underlying boosting mechanism for our incremental boosting strategy.

LogitBoost assumes that the underlying weak learner is a regression algorithm that attempts to minimize the mean-squared error. This can, for example, be a regression tree learner. For the experimental results reported in this paper we used a learning algorithm for "regression stumps." Regression stumps are 1-level regression trees. In our implementation these stumps have ternary splits where one branch handles missing attribute values.

The only difference between standard boosting and incremental boosting is that the latter uses a different dataset in each iteration of the boosting algorithm: the incoming training data is split into mutually exclusive "chunks" of the same size and a model is generated for each of these chunks. When a new chunk of data becomes available the existing committee's predictions for this chunk are used to weight the data and a new model is learned on the weighted chunk and added to the committee. In this fashion a committee of boosted models is incrementally constructed as training data is processed. Figure 1 depicts the basic algorithm for incremental boosting.

This algorithm assumes that new models can be added to the committee until the data is exhausted. This may not be feasible because of memory constraints. In the next section we will discuss a pruning method for reducing the committee's size. Another drawback of the algorithm is its time complexity, which is quadratic in the number of chunks (and therefore quadratic in the number of training instances). In each iteration i, $i-1$ base models are invoked in order to make predictions for chunk K_i (so that its instances can be weighted). Consequently this naive algorithm can only be applied if the chunk size is large relative to the size of the full dataset.

```
START with an empty committee K_0
REPEAT
  FOR next data chunk C_i DO
  BEGIN
    IF (i > 1) THEN
      weight chunk C_i according to the predictions of K_{0..i-1}
    learn model M_i for chunk C_i and add to committee K_{0..i-1}
  END
UNTIL no more chunks
```

Fig. 1. Incremental boosting.

2.2 Incremental Boosting with Pruning

To make the algorithm practical it is necessary to reduce the number of committee members that are generated. Preferably this should be done adaptively so that accuracy on future data is not affected negatively. The first design decision concerns which pruning operations to apply. The second problem is how to decide whether pruning should occur.

Boosting is a sequential process where new models are built based on data weighted according to the predictions of previous models. Hence it may be detrimental to prune models somewhere in the middle of a committee because subsequent models have been generated by taking the predictions of previous models into account. Consequently the only model that we consider for pruning is the last model in the sequence. This makes pruning a straightforward (and computationally very efficient) procedure: the existing committee is compared to the new committee that has an additional member based on the latest chunk of data. If the former is judged more accurate, the last model is discarded and the boosting process continues with the next chunk of data.

Hence the pruning process makes it possible to skip chunks of data that do not contribute positively to the committee's accuracy. As the experimental results presented in Section 3 show, this is especially useful when small chunks are used to build the committee. The experimental results also show that it is not advisable to stop building the committee when a "bad" chunk of data is encountered because later chunks of data may prove useful and lead to models that improve the committee's accuracy.

The second aspect to pruning is the choice of evaluation criterion. The pruned model needs to be compared to the unpruned one. Pruning should occur if it does not negatively affect the committee's generalization performance. Fortunately our target application domains share a common property: they exhibit an abundance of data. This means we can be generous and reserve some of the data for pruning. We call this data "validation data." This data is held completely separate from the data used for training the models. In our implementation the first N instances encountered (where N is the size of the validation dataset) are skipped by the boosting process. Consequently the first chunk of data that generates a potential committee member starts with instance $N + 1$.

```
START with an empty committee K_0 AND validation data V
REPEAT
  FOR next data chunk C_i DO
  BEGIN
    IF (i > 1) THEN
      weight chunk C_i according to the predictions of K_{0..i-1}
    learn model M_i for chunk C_i
    IF (loglikelihood for K_{0..i-1} + M_i on V >
        loglikelihood for K_{0..i-1} on V) THEN
      add M_i to K_{0..i-1}
  END
UNTIL no more chunks
```

Fig. 2. Incremental boosting with pruning.

Accuracy on the validation data is the obvious performance measure. However, we found empirically that this is not a good pruning criterion. Preliminary results showed that it results in useful models being skipped because they do not change the accuracy immediately although they do improve accuracy in conjunction with models that are built later in the process. Logistic regression attempts to maximize the likelihood of the data given the model. An alternative candidate for measuring performance is therefore the loglikelihood on the validation data. This measures the accuracy of the class probability estimates that are generated by the committee. It turns out that using the loglikelihood avoids overpruning because it is more sensitive to whether a potential committee member manages to extract useful additional information. The resulting pruning algorithm based on the loglikelihood is depicted in Figure 2.

Pruning reduces the size of the committee according to the properties of the data. Ideally no further models are added to the committee when the information in the data is exhausted. If this is the case there exists an upper bound on the number of models that are generated and the time complexity becomes linear in the number of training instances, allowing very large datasets to be processed effectively. Of course, apart from affecting running time, pruning also reduces the amount of memory that is needed to store the committee.

2.3 Racing Committees

Experimental results show that the performance of the committee varies, sometimes dramatically, with the chunk size. The chunk size should be large enough for each individual committee member to become a reliable predictor. However, as the chunk size increases, returns for each individual committee member diminish. At some point it becomes more productive to increase the diversity of the committee by starting with a new chunk. The best chunk size depends on the properties of the particular dataset and the weak learner used in the boosting process.

Given these observations it appears impossible to determine an appropriate chunk size *a priori*. Consequently the only sensible strategy is to decide on a range of chunk sizes and to run the different committees corresponding to these different chunk sizes in parallel—i.e. to "race" them off against each other. Then we can keep track of which committee performs best and use the best-performing committee for prediction. Typically the best-performing chunk size changes as more data becomes available. The validation data, which is also used for pruning, can be used to compare the performance of the committees. However, in contrast to pruning, where the loglikelihood is employed to measure performance, here it is more appropriate to use percent correct because we want to use the committee that maximizes percent correct for future data.[1]

The question remains as to how many committees to run in parallel and which set of chunk sizes to use. Ultimately this depends on the computing resources available. If the number of committees is constant then the time and space complexity of racing them is the same as the corresponding complexities for its "worst-case" member. Consequently, assuming that pruning works and after a certain number of iterations no further models are added to the committee, the overall time-complexity is linear in the number of instances, and the space complexity is constant.

In our experiments we used the following five chunk sizes: 500, 1,000, 2,000, 4,000, and 8,000. We kept the maximum chunk size relatively small because decision stumps are particularly weak classifiers and the returns on adding more data diminish quickly. Doubling the chunk size from one candidate to the next has the advantage that whenever the committee corresponding to the largest chunk size may have changed this is also true for all the smaller ones, and a comparison on the validation data at this point is fair because all the committees have "seen" the same amount of training data.

3 Experimental Results

To evaluate the performance of racing unpruned and pruned committees we performed experiments on six datasets ranging in size from approximately 30,000 to roughly 500,000 instances. The properties of these datasets are shown in Table 1.[2] We obtained them from the UCI repositories [1, 7]. The kdd cup '99 data is a reduced version of the full dataset (reduced so that incremental boosting without pruning could be applied to this data). The "Train" column shows the amount of data that is used for training the committee (excluding the validation data). We attempted to set a sufficient amount of data aside for validation and testing to obtain accurate performance estimates. In our experiments the validation set size was set to half the size of the test set. Before we split the data

[1] Of course, if the application domain requires accurate probability estimates, it is more appropriate to use the loglikelihood for choosing the best committee.

[2] The first three datasets are rather small but we chose to include them in our comparison because of the lack of publicly available large datasets.

Table 1. Datasets and their characteristics

Dataset	Train	Validation	Test	Numeric	Nominal	Classes
anonymous	30211	2500	5000	0	293	2
adult	33842	5000	10000	6	8	2
shuttle	43000	5000	10000	9	0	7
census income	224285	25000	50000	8	33	2
kdd cup '99	475000	25000	50000	34	7	22
covertype	506012	25000	50000	10	44	7

into training, validation, and test data we randomized it to obtain independent and identically distributed samples.

The first row of Figure 3 shows the results for the anonymous data. The leftmost graph shows percent incorrect on the test set for the unpruned committees as an increasing amount of training data is processed. Points mark the graph corresponding to the committee that performs best on the validation data (i.e. the committee that would be used for prediction at that point under the racing scheme). The middle graph shows the same for the pruned committees, and the rightmost graph shows the committee sizes for the pruned committees.[3]

The worst-performing chunk size is 8,000 because there is insufficient data to build a large enough committee. The final ranking of committees is the same with and without pruning (and the resulting error rates are comparable). Pruning appears to smooth fluctuations in error on the test data. Pruning also substantially reduces the committee size for small chunk sizes. After 30,000 training instances, 60 models are built without pruning for chunk size 500, with pruning there are only 16 (and it appears that the number of models has reached a plateau). No pruning is done for chunk sizes 4,000 and 8,000. It appears as if pruning should have occurred for chunk size 8,000. However, the loglikelihood on the validation data does increase after models are added and consequently no pruning occurs.

The second row of Figure 3 shows the results for the adult data. Substantial pruning occurs for chunk sizes 500 and 1,000. In both cases it smoothes fluctuations in the performance and results in improved final error. It is interesting to see that the final size of the pruned committee for chunk size 1,000 is larger than the size of the committee for chunk size 500. Pruning appears to behave correctly because the final error is lower for the former.

The shuttle data in the third row of Figure 3 is different from the previous two datasets in that very high accuracy scores can be achieved. The results show that pruning produces substantially smaller committees for chunk sizes 500, 1,000, and 2,000. However, for chunk sizes 500 and 1,000 it also results in fractionally lower accuracy. Choosing the best-performing committee on the validation data under the racing scheme results in approximately the same final error rate both with and without pruning.

[3] The size of the unpruned committees is not shown because it increases linearly with the amount of training data.

Fig. 3. Error On Test Data: No Pruning (left), Loglikelihood Pruning (center). Committee Sizes: Loglikelihood Pruning (right)

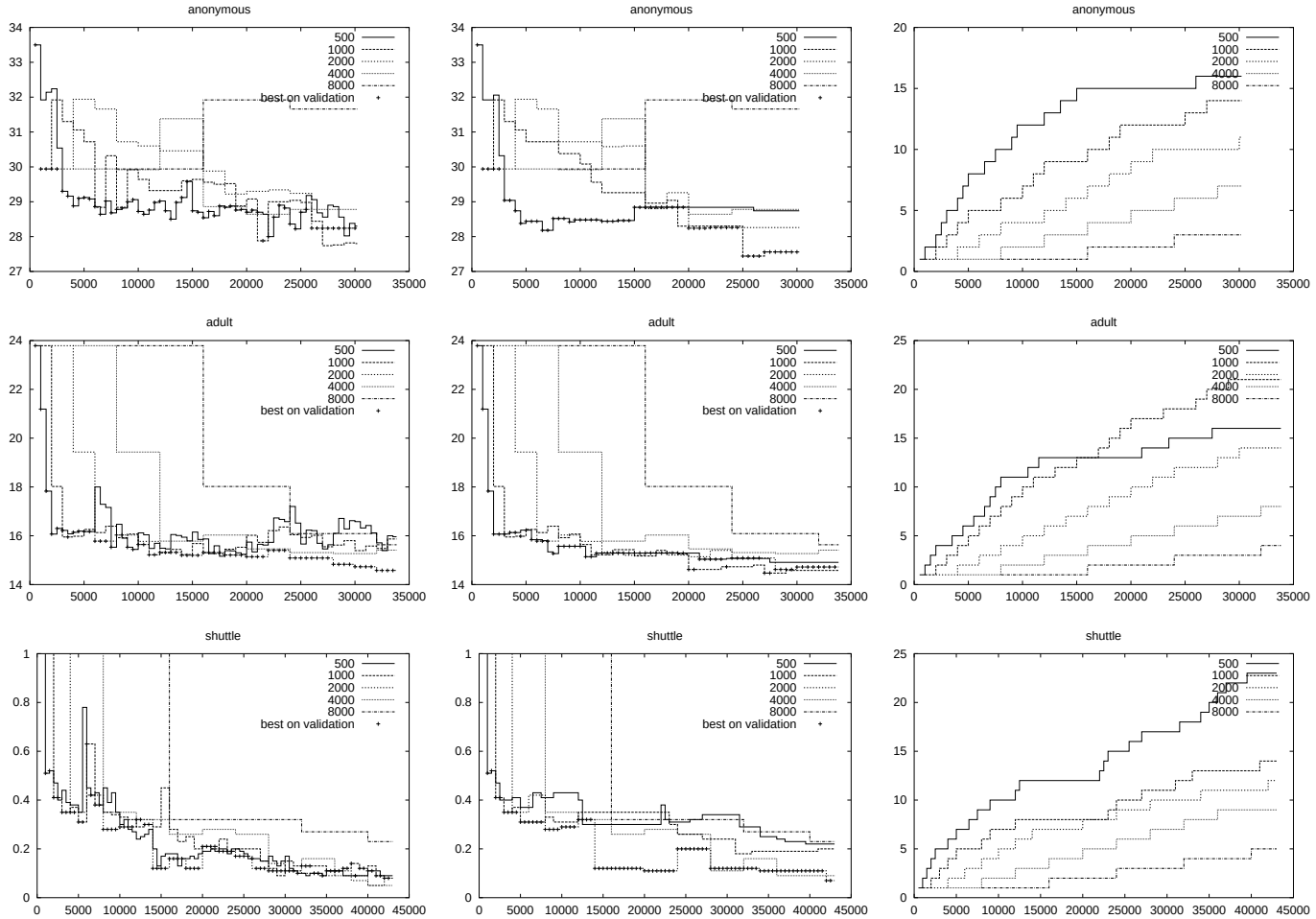

Fig. 4. Error On Test Data: No Pruning (left), Loglikelihood Pruning (center). Committee Sizes: Loglikelihood Pruning (right)

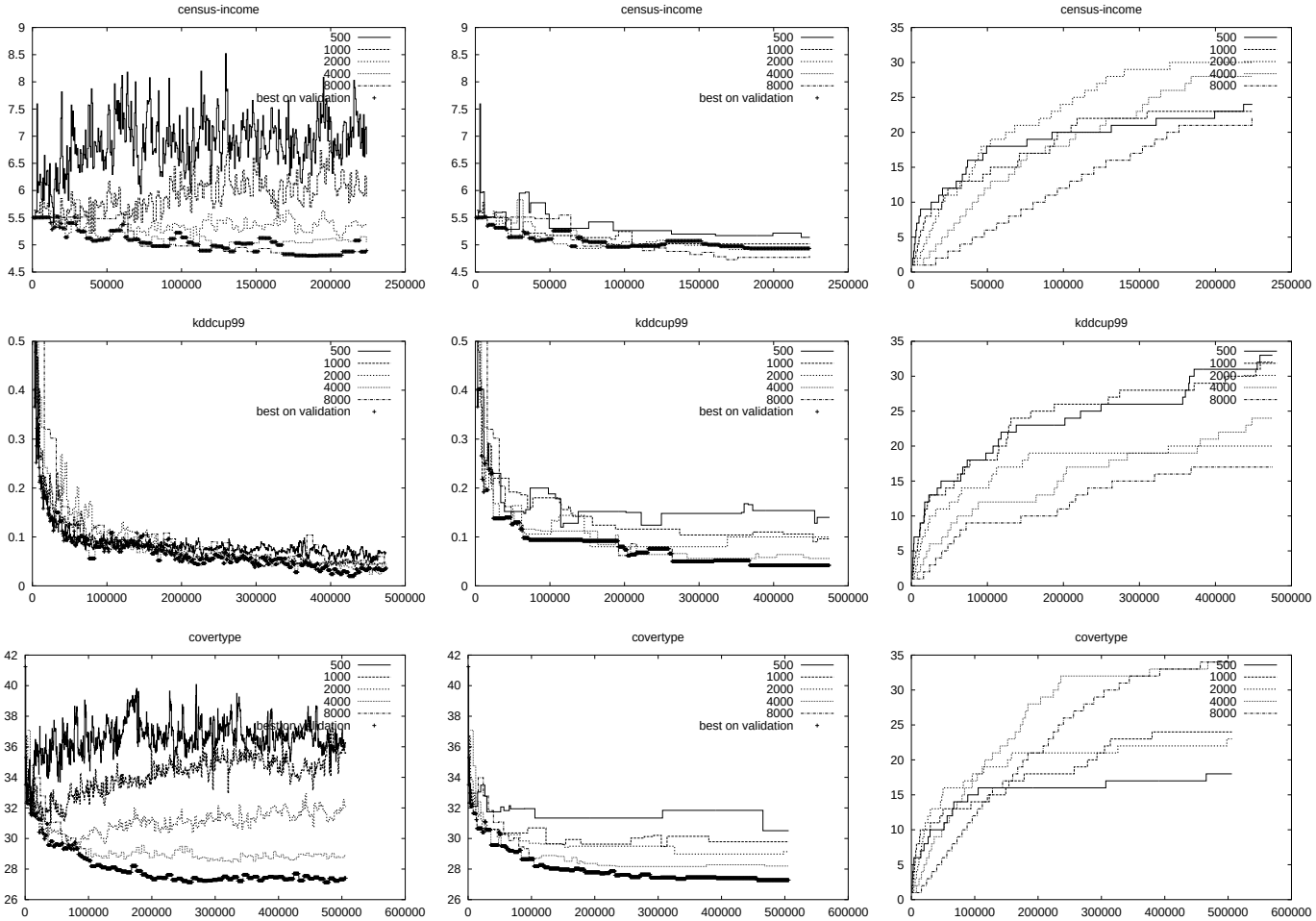

Table 2. Percent incorrect for standard LogitBoost compared to our racing scheme

Dataset	LogitBoost	#Iterations	Racing w/o pruning	Racing w pruning
anonymous	27.00%	60	28.24%	27.56%
adult	13.51%	67	14.58%	14.72%
shuttle	0.01%	86	0.08%	0.07%
census-income	4.43%	448	4.90%	4.93%

The next dataset we consider is census-income. The first row of Figure 4 shows the results. The most striking aspect is the effect of pruning with small chunk sizes. In this domain the fluctuation in error is extreme without pruning. With pruning this erratic behavior disappears and error rates decrease dramatically. Pruning also results in a marginally lower final error rate for the largest chunk sizes. The results also show that the size of the pruned committees starts to level out after a certain amount of training data has been seen. Note that even though chunk size 8,000 results in the most accurate pruned committee on the test data, chunk size 4,000 is chosen for prediction based on superior performance on the validation set.

The kdd cup '99 domain is similar to the shuttle domain in that very high accuracy can be achieved. As in the shuttle domain overpruning occurs for small chunk sizes (however, note that the degradation in performance corresponds to fractions of a percent). Although this is difficult to see on the graphs, pruning marginally improves performance for the largest chunk size (8,000). Under the racing scheme the final performance is approximately the same both with and without pruning. Because the dataset is so large, pruning results in substantial savings in both memory and runtime.

The behavior on the largest dataset (third row of Figure 4) is similar to that seen on the census-income dataset. The only difference is that the pruned version chooses chunk size 4,000 on census-income, whereas 8,000 is chosen for covertype. Pruning substantially increases accuracy for chunk sizes 500, 1,000, and 2,000 eliminating the erratic behavior of the unpruned committees. The best-performing committee (both pruned and unpruned) is based on a chunk size of 8,000. Pruning does not improve the accuracy of the final predictor under the racing scheme. However, it does lead to substantial savings in both memory and runtime. The final committee for chunk size 8,000 is less than half the size of the unpruned version.

Table 2 compares the final error under the racing scheme to standard LogitBoost (i.e. where the weak learner is applied to the *full* training set in each boosting iteration). We set the number of iterations for standard LogitBoost to be the same as the number of committee members in the largest unpruned committee (i.e. the one built from chunk size 500). The table does not include results for the two largest datasets because processing them with standard LogitBoost was beyond our computing resources. As might be expected, standard LogitBoost is slightly more accurate on all four test sets. However, the results are very close.

4 Related Work

Breiman [3] appears to have been the first to apply boosting (or "arcing" [2]) to the problem of processing large datasets by using a different subsample in each iteration of the boosting algorithm. He shows that this produces more accurate predictions than using bagging in the same fashion. He also shows that incremental boosting (if used in conjunction with an appropriate subsample size) produces classifiers that are about as accurate as the ones generated by standard boosting applied to the full dataset. However, his work does not address the problem of how to decide which committee members to discard.

Fan *et al.* [4] propose an incremental version of AdaBoost that works in a similar fashion. Their method retains a fixed-size "window" of weak classifiers that contains the k most recently built classifiers. This makes the method applicable to large datasets in terms of memory and time requirements. However, it remains unclear how an appropriate value for k can be determined.

Street and Kim [14] propose a variant of bagging for incremental learning based on data chunks that maintains a fixed-size committee. In each iteration it attempts to identify a committee member that should be replaced by the model built from the most recent chunk of data. Because the algorithm is based on bagging (i.e. all data points receive equal weight and a simple majority vote is performed to make a prediction), the algorithm has limited potential to boost the performance of the underlying weak classifiers.

Oza and Russell [10] propose incremental versions of bagging and boosting that differ from our work because they require the underlying weak learner to be incremental. The method is of limited use for large datasets if the underlying incremental learning algorithm does not scale linearly in the number of training instances. Unfortunately, the time complexity of most incremental learning algorithms is worse than linear.

Prodromidis and Stolfo [11, 12] present pruning methods for ensemble classifiers built from different (unweighted) subsets of a dataset. These methods require an unpruned ensemble to be built first before pruning can be applied. The pruned ensemble cannot be updated incrementally as new data arrives. Similarly, Margineantu and Dietterich [9] investigate pruning methods for ensembles built by the standard AdaBoost algorithm (i.e. where a weak classifier is built from the entire dataset in each boosting iteration). Again, their method is applied once an unpruned ensemble has been generated.

5 Conclusions

This paper has presented a method for efficiently processing large datasets using standard learning techniques by wrapping them into an incremental boosting algorithm. The main contribution of this paper is a pruning method for making the procedure efficient in terms of memory and runtime requirements. The accuracy of the resulting committee depends on an appropriately chosen chunk

size. Experimental results obtained by racing candidate solutions based on different chunk sizes demonstrate the effectiveness of our method on six real-world datasets.

Although our technique can be used in an online setting, it cannot be applied in domains with concept drift (i.e. where the target concept changes over time) because it assumes that all the incoming data is independent and identically distributed.

Acknowledgments

We would like to thank Bernhard Pfahringer for his valuable comments.

References

1. C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. [www.ics.uci.edu/∼mlearn/MLRepository.html].
2. Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.
3. Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, pages 85–103, 1999.
4. Wei Fan, Salvatore J. Stolfo, and Junxin Zhang. The application of AdaBoost for distributed, scalable and on-line learning. In *5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 362–366, 1999.
5. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th Int. Conf. on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.
6. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistic*, 38(2):337–374, 2000.
7. S. Hettich and S. D. Bay. The UCI KDD archive, 1999. [http://kdd.ics.uci.edu].
8. G. H. John and P. Langley. Static versus dynamic sampling for data mining. In *2nd ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases and Data Mining*, pages 367–370, 1996.
9. D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proc. of the 14th Int. Conf. on Machine Learning*, pages 211–218, 1997.
10. Nikunj Oza and Stuart Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *7th ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases and Data Mining*, pages 359–364, 2001.
11. A. L. Prodromidis, S. J. Stolfo, and P. K. Chan. Pruning classifiers in a distributed meta-learning system. In *Proc. of 1st National Conference on New Information Technologies*, pages 151–160, 1998.
12. Andreas L. Prodromidis and Salvatore J. Stolfo. Cost complexity-based pruning of ensemble classifiers. *Knowledge and Information Systems*, 3(4):449–469, 2001.
13. A. Srinivasan. A study of two sampling methods for analysing large datasets with ILP. *Data Mining and Knowledge Discovery*, 3(1):95–123, 1999.
14. W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *7th ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases and Data Mining*, pages 377–382, 2001.