A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification

W. Nick Street
Management Sciences Department
University of Iowa
Iowa City, IA 52242 USA
nick-street@uiowa.edu

YongSeog Kim
Management Sciences Department
University of Iowa
Iowa City, IA 52242 USA
yong-s-kim@uiowa.edu

ABSTRACT

Ensemble methods have recently garnered a great deal of attention in the machine learning community. Techniques such as Boosting and Bagging have proven to be highly effective but require repeated resampling of the training data, making them inappropriate in a data mining context. The methods presented in this paper take advantage of plentiful data, building separate classifiers on sequential chunks of training points. These classifiers are combined into a fixed-size ensemble using a heuristic replacement strategy. The result is a fast algorithm for large-scale or streaming data that classifies as well as a single decision tree built on all the data, requires approximately constant memory, and adjusts quickly to concept drift.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management—*Database Applications*[Data Mining]

General Terms

ensemble classification, streaming data

1. INTRODUCTION

One of the most common and well-studied tasks in predictive data mining and knowledge discovery is that of classification. Over the past 25 years, a great deal of research has been performed on inductive learning methods for classification such as decision trees, artificial neural networks, and support vector machines. All of these techniques have been successfully applied to a great number of real-world problems. However, their standard application requires the availability of all of the training data at once, making their use for large-scale data mining applications problematic.

Many organizations are collecting data at the rate of millions of records per day. Processing this data poses a significant challenge for existing data mining methods; consider, for example, a retail chain predicting the success of a particular marketing campaign, a telephone company detecting fraudulent phone card calls, or an online company determining which Web pages lead customers to place an order. Even in cases where we cannot read all of the training examples into memory at one time, we would like to use all the available information when building our classifier. Further, such problems are subject to gradual or sudden changes in the underlying concept, as business conditions not reflected in the predictive features – the "hidden context" [22] – can change without warning. This concept drift requires an algorithm that can adjust quickly to changing conditions.

One approach to large-scale classification is to improve the storage efficiency of the induction algorithm, allowing its use on much larger problems. This approach is exemplified by the work of Gehrke *et al.* [12] who developed a decision tree algorithm with dramatically improved efficiency. Their BOAT algorithm builds an initial tree on a subset of the data and refines it iteratively as new data are read, concluding with a tree identical to one that would be built by the original algorithm. The VFDT algorithm of Domingos and Hulten [8] also builds a decision tree incrementally, using a small subset of examples to determine the split at a given node. They employ Hoeffding bounds to show that the resulting tree can be made arbitrarily similar to one that would be built from all the data. Other approaches to scaling include those developed for support vector machines by Mangasarian and colleagues, including chunking [16] and instance selection [11]. Finally, the many formulations for feature selection in machine learning could be considered scaling mechanisms, since they result in dimensionality reduction, although most were motivated by overfitting avoidance.

In this study we approach the problem of large-scale or streaming classification by building *committee* or *ensemble* classifiers that combine the results of many classifiers, each constructed on a subset of the available data points. It has long been known that the combined expertise of a committee of experts can outperform an individual expert, particularly if their respective expertise is somehow different. Specifically, several analyses of ensemble methods have shown that the correctness improved if the individual predictors make errors that are independent of each other [14].

In recent years, a great deal of attention in the machine learning community has been directed toward methods such as Bagging and Boosting. Breiman's Bagging algorithm [4] is a variation of the bootstrap method that resamples the data points with replacement to construct a collection of different training sets. The resulting classifiers are combined with voting to find those concepts that are most often reinforced in the samples. Boosting [20], and its variants such as AdaBoost [10] and Arcing [5], uses a weighted resampling technique, creating a series of classifiers in which later individuals focus on classifying the more difficult points. Several recent studies have examined the relative strengths of these techniques; see for instance Bauer and Kohavi [1], and Opitz and Maclin [18]. The current evidence seems to show that Bagging consistently improves on the accuracy of a single classifier, while Boosting is sometimes better (or even much better) than Bagging but sometimes worse than a single classifier, especially in the presence of noise.

While all of these methods result in highly accurate predictors, the necessity of resampling or re-weighting the data points makes them inappropriate for large-scale applications. Our motivation is the idea that even simple ensembles can produce results that are comparable to individual classifiers. Our solution to the problem is designed to meet the following data mining desiderata, a superset of those outlined by Fayyad *et al.* [9] for large-scale predictive methods:

1. **Iterative:** The algorithm must operate in an iterative fashion, reading blocks of the data at a time, rather than requiring all of it at the beginning.
2. **Single pass:** The algorithm should make only one pass through the data.
3. **Limited memory:** Data structures used by the algorithm should require an approximately constant amount of memory, and should not have sizes that depend on the size of the data.
4. **Any-time learning:** The algorithm should provide a best answer if it is stopped before its conclusion.

The remainder of this paper is organized as follows. In Section 2, we motivate and describe our framework for ensemble construction. Section 3 describes the data sets used in our experiments and shows the computational results. Section 4 discusses the conclusions we reached based on these experiments and outlines directions for future research.

2. METHODS

Our design, based on simple combinations of classifiers is outlined as follows and represented as pseudocode in Figure 1. Individual classifiers are built from relatively small subsets of the data, read sequentially in blocks. Component classifiers are combined into an ensemble with a fixed size. Once the ensemble is full, new classifiers are added only if they satisfy some quality criterion, based on their estimated ability to improve the ensemble's performance. In this case, one of the existing classifiers must be removed, maintaining the ensemble's constant size. Performance estimates are done by testing the new tree (and the existing ensemble) on the next chunk of data points. The building blocks of all of our ensembles are decision trees constructed using Quinlan's C4.5 [19]. The only operational parameter of C4.5 examined here is whether or not to prune the trees; when they are pruned, the default setting is used.

We performed some preliminary experiments on this basic framework, varying a number of operational parameters. A few early results included:

```
while more data points are available
  read d points, creating training set D
  build classifier C_i using D
  evaluate classifier C_{i-1} on D
  evaluate all classifiers in ensemble E on D
  if E not full
    insert C_{i-1}
  else if Quality(C_{i-1}) > Quality(E_j) for some j
    replace E_j with C_{i-1}
  end
end
```

Figure 1: Pseudocode for streaming ensembles. E_j represents the jth tree in ensemble E.

- Increasing the size of the ensemble resulted in better generalization, up to around 20 or 25 classifiers. Obviously there is a trade-off between the number of classifiers and the number of points per classifier with data sets of limited size, unless resampling is performed.
- Pruning the individual trees resulted in *decreased* ensemble accuracy, even though the accuracy of the trees themselves was increased. This is consistent with the findings of [21] who concluded that overtraining the individual classifiers was a useful ensemble strategy.
- Simple variations on majority voting had little or no effect on ensemble accuracy. Weighting the votes based on classifier accuracy, or on the confidence of each classification, resulted in small and inconsistent improvements.
- We also experimented with a "gated" voting procedure, in which a separate classifier G_i was trained to predict whether its corresponding classifier member T_i would correctly classify any given point. Ensemble members would then vote only on cases they were predicted to get right. This method also showed no consistent improvement over simple voting.

These qualitative results are included here as motivation for some of our design decisions. We set the number of ensemble elements to 25, and combined their predictions with majority voting (ties broken randomly). No gating or filtering of the training examples was performed. Component trees were not pruned.

The key to the performance of this algorithm is the method used to determine whether a new tree should be added to the ensemble, and which existing tree should be removed. Metrics like accuracy can be accumulated based on many individual estimates while the algorithm is running. Our experiments show that simple accuracy is not the best metric; as discussed earlier, classification diversity plays an important role in the ensemble's performance. This suggests that diversity should be built directly into the ensemble objective, as was done in [17]. However, in addition to the problem of determining the right trade-off between accuracy and diversity, it turns out that diversity is much harder to measure in our framework. Estimates based on a single test set are noisy, and accumulating the measurements over time is problematic, since the ensemble mix is constantly changing. Another possibility is to favor classifiers that do well on points misclassified by most of the ensemble, as in Boosting, but this leaves the method susceptible to noisy data.

Instead, we favor classifiers that correctly classify points on which the ensemble is nearly undecided. We give each classifier (the new candidate and the ensemble members) a quality score based on its ability to classify the points in the current test set (i.e., the next set of points). If the tree in question gets a point right (wrong), it's quality goes up (down) in proportion to how close the ensembles voting was to 0.5 (for two-class problems). In this way, little credit (or blame) is given to a case in which the ensemble was homogeneous in its predictions. If a point is easy, or impossible, to classify, the effect of a new classifier will be minimal; however, the classification of a point on which the ensemble is evenly split could be decided by one vote either way. This method may help overcome one problem of Boosting, which performs poorly on data sets with classification noise because of its emphasis on "hard" points.

More formally, we define the following percentages based on the number of votes received by the various classes for a given training point:

P_1 = percentage of top vote-getter
P_2 = percentage of second-highest vote-getter
P_C = percentage for the correct class
P_T = percentage for the prediction of the new tree T

If both ensemble E and new tree T are correct, the quality measure for T is increased by 1 - $|P_1 - P_2|$. That is, if the vote was close, the new tree gets a high quality score, since it could directly affect future votes. If T is correct but E was incorrect, quality is increased by 1 - $|P_1 - P_C|$. Finally, if T's prediction was incorrect (regardless of the outcome of E), its quality is decreased by 1 - $|P_C - P_T|$. Note that these quantities are defined for problems with an arbitrary number of classes, although our experiments to date have focused on two-class problems.

3. EXPERIMENTAL RESULTS

3.1 Accuracy

The following real-world data sets were used to evaluate the effectiveness of the ensemble construction method.

- Adult: Data from the U.S. Census Bureau was originally used by Kohavi [15] to compare different classification methods. The classification problem is to predict whether a person makes more or less than \$50,000 per year based on 14 demographic features such as age, education level, marital status, occupation, and gender. The data set contains 44,848 instances of which 29.3% are in the "over 50k" class.

- SEER breast cancer: The breast cancer data set from the Surveillance, Epidemiology, and End Results (SEER) program [6] of the National Institutes of Health contains follow-up data on over 44,000 breast cancer patients. The cases were filtered to create a classification problem. Class 1 contains those patients who died of breast cancer within five years of surgery, and class 2 contains those with at least five years survival. This filtering results in a data set of 37,715 cases, 25.7% of which belong to class 1. Predictive features include nuclear grade, tumor extent, tumor size, lymph node status, and number of lymph nodes examined.

- Anonymous Web browsing: This data set records browsing patterns for 32,710 anonymous visitors to the Microsoft Web site. We created a classification problem in a manner similar to Breese, *et al.* [3] by choosing to predict whether a visitor browses the "Free downloads" page, based on the other pages the user visits. Filtered in this way, the data set contains 10,835 (33.1%) positive examples, and 296 binary features.

The first and third data sets are publicly available from the UCI machine learning repository [2]. In all cases, the ensemble methods were compared to the results of constructing single decision trees on the entire training set. Data sets were therefore chosen to be large enough to reliably test the ensemble techniques but small enough to permit one-shot training. The plots in Figures 2 through 4 represent the results of a five-fold cross-validation run for both ensemble and single-classifier solutions, as more data points become available. Both pruned and unpruned single trees were evaluated in the single-classifier tests. Plots are shown at various scales to make algorithm differences more clear.

In all cases, the accuracy of the ensemble method is comparable to that obtained with a single decision tree. Typically, the ensemble performance is between that of the pruned and unpruned single trees, although in the case of the SEER data, the ensemble did slightly better than the pruned tree. Only on the Adult data are there points at which the classification rate of a full ensemble is statistically significantly different ($\alpha = 0.05$) from a single pruned tree trained on the same number of points. Such a difference occurs on 27% of the test increments for the Adult 1000 runs, and 83% for the Adult 500 test. The performance of the ensemble on this data, which is known to have classification noise, may indicate that our approach still has a certain susceptibility to noise.

Tests were also performed using fewer (100 and 200) points for each tree. Results were consistently inferior to the 500 and 1000 points/tree cases shown in the plots. This indicates that the quality of the component classifiers – and possibly their ability to "overfit" their respective chunks of data – still has a significant effect on the ensemble performance.

These runs were also used to evaluate the effectiveness of our tree-replacement strategy by counting the number of times the ensemble's accuracy improved after the initial 25 trees were constructed. We observed improvement in 90% of the runs on the Adult data, and 84% on the SEER data, and 58% on the Anonymous data. Using ensembles with fewer elements improved these numbers, but had little or no effect on overall accuracy. Our replacement strategy is appears to be generally effective.

3.2 Concept drift

In order to test our algorithm on data displaying concept drifts over time, we generated an artificial data with two classes as follows. We first generated 60,000 random points in a three-dimensional feature space. All three features have values between 0 and 10 but only the first two features are relevant. We then divided those points into four blocks with different concepts. In each block, a data point belongs to class 1 if $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two features and θ is a threshold value between the two classes. We used threshold values 8, 9, 7, and 9.5 for the four data blocks. We inserted about 10% class noise into each block of data. Finally, we reserved 2,500 records from each block as test sets for the different concepts. We summarize the class distributions of our training and testing sets in Table 1.

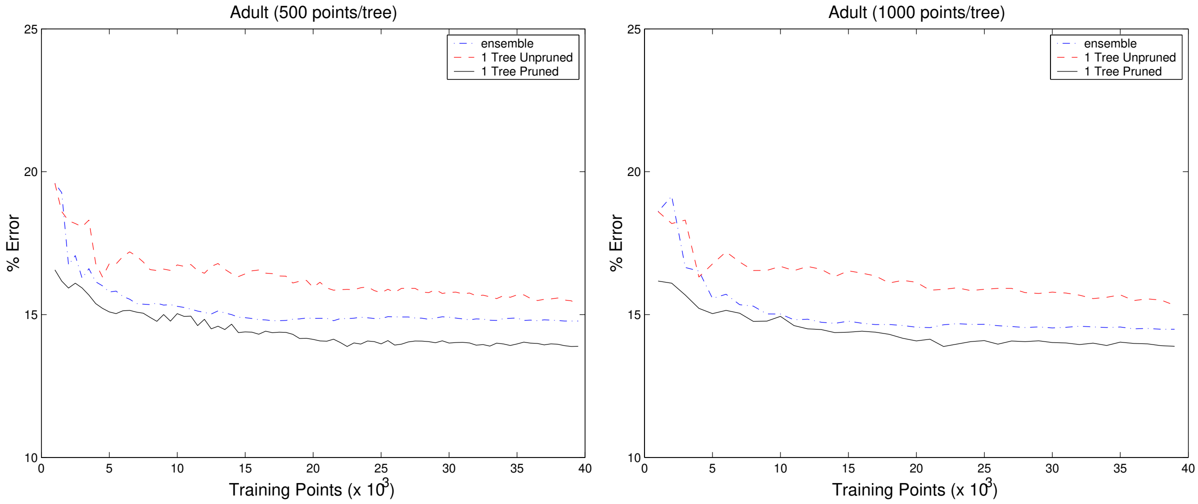

Figure 2: Error on Adult data

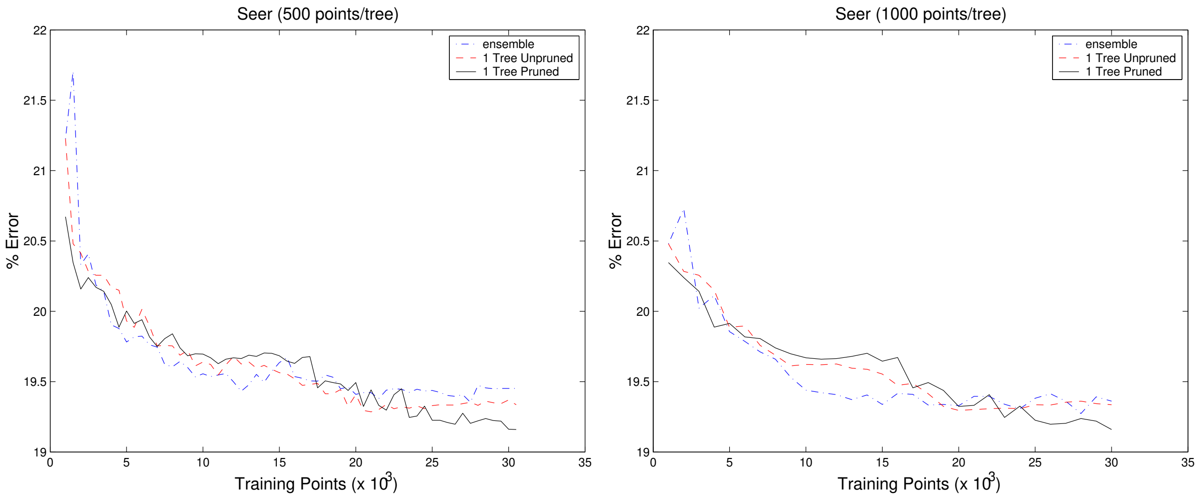

Figure 3: Error on SEER data

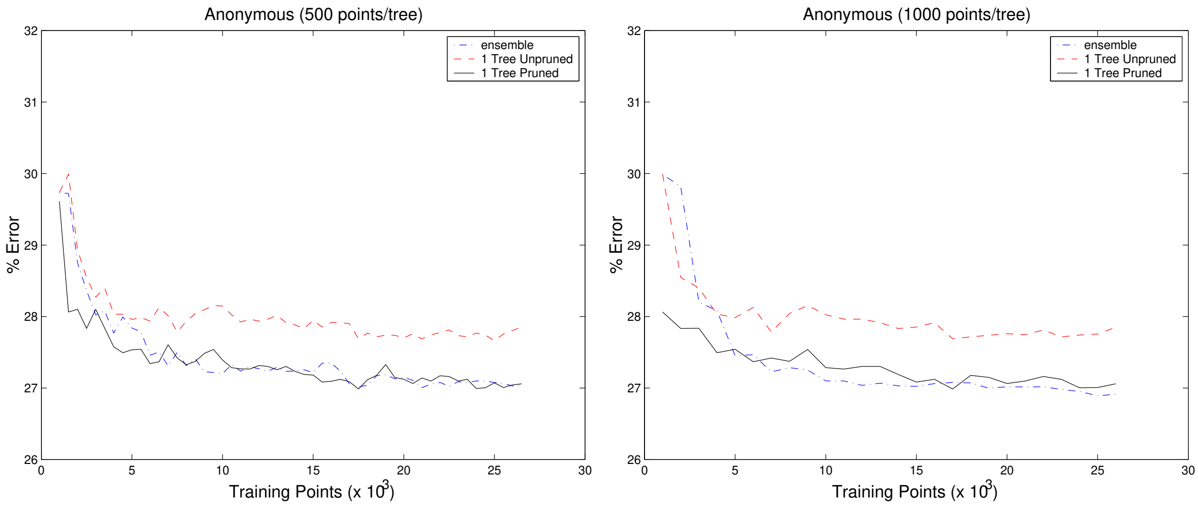

Figure 4: Error on web-browsing data

Block	Train set				Test set			
	Class 1		Class 2		Class 1		Class 2	
	points	%	points	%	points	%	points	%
Block 1	3,602	28.8	8,898	71.2	761	30.4	1,739	69.6
Block 2	4,473	35.8	8,027	64.2	891	35.6	1,609	64.4
Block 3	2,758	22.1	9,742	77.9	553	22.1	1,947	77.9
Block 4	4,995	40.0	7,505	60.0	1,031	41.2	1,469	58.8
Total	15,828	31.7	34,172	68.3	3,236	32.4	6,764	67.6

Table 1: The class distributions of an artificial data demonstrating concept drift

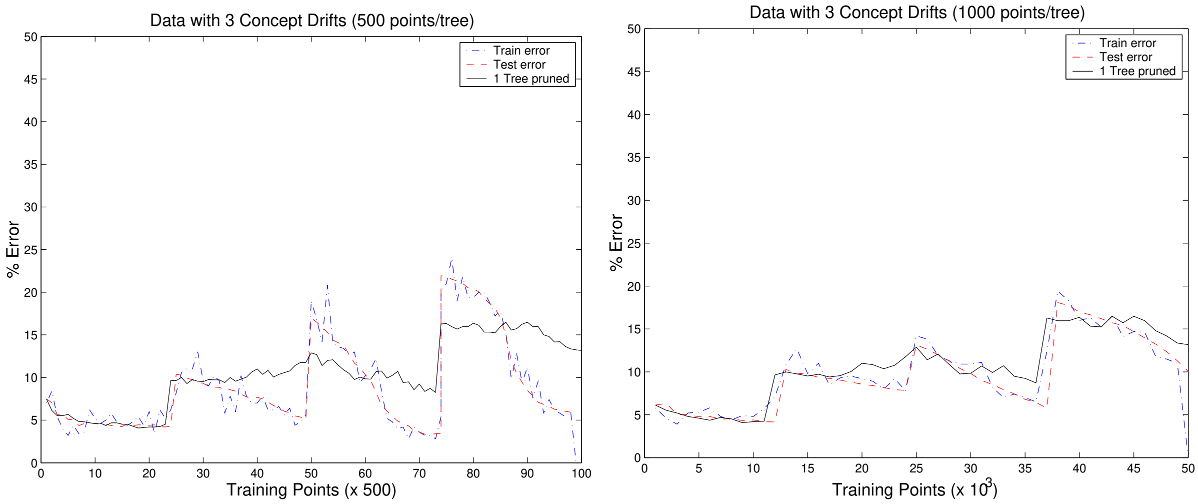

Figure 5: Error on simulated data with changing concept

Figure 5 shows the generalization results for a single tree and for an ensemble ("Test error" in the figure legend). As expected, when the target concept changes suddenly, both techniques have a jump in error; in fact, the ensemble is affected somewhat more dramatically. However, the dynamic nature of our method allows the ensemble accuracy to recover very quickly. The trees that were trained on the outdated concept are replaced, and the error returns very quickly to its original level – in some cases, even lower. The single decision tree, which is still using data points from the old concept, recovers much more slowly, if at all.

The ensembles built with 500 points per tree adapted to the new concept faster than those with 1000 points per tree. While this behavior is a positive trait in this experiment, in other situations the sensitivity to new data may be a drawback. This relationship between the number of points per tree and adaptability warrants further investigation.

The "Train error" curve in these figures is the observed error of the ensemble on the next chunk of data. This error closely tracks the testing error, providing an accurate on-line estimate of generalization error. In an application to streaming data, in which a separate test set is not available, this estimate could be very valuable. To reduce noise in this estimate, the observed error could be averaged over a sliding window of test sets.

4. DISCUSSION AND CONCLUSIONS

We have presented an ensemble-based solution to the problem of large-scale or streaming classification. Our framework facilitates any-time learning on problems of arbitrary size. Our method is efficient, as accurate as a single classifier, and adjusts quickly to changes in the target concept. It provides a natural mechanism for estimating the generalization accuracy of the model at any given stage of its construction. Moreover, the method is easy to implement, and independent of the underlying classifier.

As part of our evaluation, we return to the list of data mining criteria mentioned in Section 1. Our approach was designed to read blocks of data, rather than the entire data set, at a time. In addition, since the ensembles are built incrementally, we have certainly satisfied the "any-time learning" criterion. The algorithm may be stopped at any point, with the collection of classifiers built up to that point providing the "answer." In fact, since near-maximum accuracy is usually found fairly quickly (less that 10,000 points, in most cases), an intermediate ensemble is likely to perform well. By limiting the number of classifiers in the ensemble, and reading only a few points at a time, the algorithm uses only a nearly-constant amount of memory. This has a negligible effect on accuracy and allows the application of the method to data sets of arbitrary size. Satisfaction of the "single-pass" criterion is somewhat less obvious. Certainly we are reading and processing each block of data only once, but the individual points are re-examined several times in the construction of the tree in the C4.5 algorithm. However, since decision-tree algorithms like C4.5 are extremely fast for moderately-sized data sets, this is not a serious drawback.

As stated previously, many large-scale prediction problems have a temporal component, and are subject to concept drift over time. By quickly replacing trees that were trained on the old concept, our method can recover from changes in the target concept much faster than methods that use all of the training points in a single model.

The next step of this research is to parallelize the algorithm in the straightforward way and compare results, in terms of both speedup and accuracy, to other meta-learning methods [7, 13]. In the long term, we will focus on improving the generalization accuracy of the method. While the accuracy appears to be about the same as a single classifier, our use of ensemble classification suggests that we can do significantly better. One direction is to examine the diversity mechanism. We use different blocks of data for the various classifiers to create classification diversity, which is necessary for effective ensembles. We will experiment with other approaches to promoting diversity, such as the use of different classification methods, limited data resampling, and different methods of combining predictors. Another simple variation is to keep a working set of high-quality classifiers that is slightly larger than the ensemble size, and test all the combinations of these on the evaluation set. This "best-n-of-m" strategy moves us somewhat closer to a globally optimal set, while limiting the additional computation.

In order to properly "optimize" an ensemble, it is important to have a clear understanding of the mechanisms through which ensembles achieve their effectiveness. In our related work in the direct optimization of ensembles via evolutionary search, we are in the early stages of a data mining task over the space of possible ensembles, varying ensemble characteristics and allowing a group of ensembles to compete for limited resources based on accuracy. Resulting insights will be incorporated into the greedy optimization scheme described here.

Acknowledgements

This work was supported in part by NSF grant IIS-9996044. The authors wish to thank Cris Choi and Danni Jiao for their help with the experiments, Alberto Segre for his helpful comments, and Dr. Don Henson of the National Institutes of Health for providing the SEER data.

5. REFERENCES

[1] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–142, July-August 1999.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1998. University of California, Irvine, Department of Information and Computer Sciences.

[3] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26(3):801–849, 1998.

[6] C. L. Carter, C. Allen, and D. E. Henson. Relation of tumor size, lymph node status, and survival in 24,740 breast cancer cases. *Cancer*, 63:181–187, 1989.

[7] P. K. Chan and S. J. Stolfo. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems*, 8:5–28, 1997.

[8] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM Press, 2000.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, 1996.

[10] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[11] G. Fung and O. L. Mangasarian. Data selection for support vector machine classifiers. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 64–70. ACM Press, 2000.

[12] J. Gehrke, V. Ganti, R. Ramakrishnana, and W.-Y. Loh. BOAT - Optimistic decision tree construction. In *Proceedings of the 1999 SIGMOD Conference*, Philadelphia, PA, 1999.

[13] L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, T. E. Moore, and C. Chao. Distributed learning on very large data sets. In *Workshop on Distributed and Parallel Knowledge Discovery (KDD-00)*, pages 79–84, Aug 2000.

[14] R. A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888, 1995.

[15] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[16] O. L. Mangasarian and D. R. Musicant. Massive support vector regression. *Machine Learning*, to appear.

[17] D. Opitz. Feature selection for ensembles. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, pages 379–384, 1999.

[18] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[19] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[20] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[21] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[22] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.