Creating Ensembles of Classifiers

Nitesh Chawla, Steven Eschrich, and Lawrence O. Hall
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave. Tampa, Fl 33620
chawla, eschrich, hall@csee.usf.edu
FAX 813 974 5456

June 14, 2001

Abstract

Ensembles of classifiers offer promise in increasing overall classification accuracy. Combining multiple classifiers, bagging, and boosting have been utilized to improve the accuracy of classification over a single classifier learned on the entire data. The availability of extremely large datasets has opened avenues for application of distributed and/or parallel learning to efficiently handle very large datasets. In this paper, we look at distributed learning by training classifiers on either random disjoint subsets or subsets created by a clustering algorithm. We examine a random partitioning method to create disjoint subsets and propose a more intelligent way of partitioning into disjoint subsets using clustering. We observe that the intelligent method of partioning generally performs better than random paritioning for our datasets. The reduced complexity associated with the technique of producing random disjoint partitions is attractive for creating classifiers on extremely large datasets. A significant gain in accuracy may be obtained by applying bagging to each of the disjoint subsets, creating multiple diverse

classifiers. The significance of our finding is that a partition strategy on small/moderate sized datasets when combined with bagging can yield better performance than applying a single learner using entire dataset. Similar results may be expected for larger data sets, which suggests a parallel, practical, minimal time approach to learning a classifier from large training datasets.

1 Introduction

Dataset sizes are continually increasing as more and more information is stored electronically. Machine learning techniques are being utilized to learn models over increasingly large feature and example spaces. Efficiently learning from these large datasets is difficult, as datasets can not always be completely loaded into a computer's memory. Reducing training set sizes to the size of available memory or less is a practical approach in machine learning. An attractive option for learning from large datasets is *distributed learning*: data and learning are distributed across different processors (and computers). The approach discussed here is to learn an ensemble of individual classifiers, with each learner creating it's own classifier from a subset of the total dataset.

Disjoint partitioning of a training set is a simple, potentially useful approach in distributed learning. Bagging [5] is a technique that uses repeated random samples of the dataset, thus the sum of cardinalities of subsets is greater than the total size of the initial dataset. Disjoint partitioning can potentially require less processing, since the total amount of training data is not increased. A key issue in partitioning the dataset is the computational effort required. We examine both a simple random partition of the dataset and a more intelligent, and more computationally demanding, clustering method. In either case, disjoint partitioning should lend itself well to distributed learning. Every example from the original dataset will be represented in some subset. In addition, individual learners will be presented different views of the dataset, a feature considered useful in combining of classifiers [11, 14, 6, 4, 10, 15].

Distributed learning involves such issues as data presentation to individual learners and the combination of the classification output of all members of the ensemble. In this paper, we examine the ensemble of classifiers approach in learning classifiers from datasets. Specifically, our proposed method of learning involves partitioning a dataset into disjoint subsets. A classifier is then learned or built for each subset. We examine both a random partitioning method and a more intelligent partitioning method using clustering. With the addition of the bagging technique [5] applied to subsets contained in partitions, we show that disjoint dataset partitioning can actually yield better classifier performance than learning one model over the entire dataset.

2 Background

Machine learning approaches have been generally studied in the context of modest-sized datasets. Larger-sized datasets can show new weaknesses or peculiarities of the learner. In addition, machine learning on large datasets is approaching the stumbling block inherent to most artificial intelligence: scalability. By partitioning the training data and allowing multiple classifiers to be learned independently (and possibly in parallel) we retain some of the characteristics of earlier machine learning work. The added difficulties in distributed learning, partitioning the data (e.g. disjoint or sampled) and a method of combining the knowledge learned in each individual classifier (e.g. majority voting or weighted voting) must be addressed.

In [4, 10], it was demonstrated that datasets too large to be processed on a single processor can be efficiently handled in a distributed fashion. The resulting accuracy with distributed learning was essentially the same as the sequential model on the entire dataset, especially with large datasets [10]. Chan and Stolfo [6, 7] proposed a comparison of arbiter and combiner strategies by applying a learning algorithm to disjoint subsets of data. Their experiments showed that the arbiter strategy sustains the accuracy compared to the classifier learned on the entire data set. The combiner strategy experienced a drop in accuracy with the increase in the number of subsets, which can be attributed to the lack of enough information content from the small subsets, however in a few cases an improvement in the accuracy was seen.

The machine learning work most related to our intelligent approach of partitioning data is the hierarchical mixture of experts approach used in neural networks [8, 13]. In this approach, radial basis functions or neural networks are used to segment the dataset and classifiers are built on individual segments. Classification is accomplished by first routing the test example to the appropriate expert. This expert was trained on similar data and is used to classify the given example. We use a similar approach by clustering, which partitions the data into clusters of related data. Since, we group examples without respect to class using an unsupervised learning method, our "expert" learners are required to discriminate among several classes within their regions of expertise. Additionally, we wish to use a less computationally demanding approach to partition the dataset.

Clustering is a method of grouping like data. Rather than randomly choose partition elements, this approach attempts to choose "similar" elements for a partition. We use a fuzzy c-means (FCM) clustering algorithm for partitioning [3]. A c-means clustering algorithm uses c randomly initialized cluster centroids. When using FCM, each element of the dataset is assigned membership to each cluster, based on the distance between the example and the cluster centroid. Then, the cluster centroids are recomputed as essentially the weighted average of all examples in the dataset, where the weighting is proportional to the membership of the example to the given cluster. The euclidean distance in s-dimensional feature space is often used as the distance metric. Degree of membership in a cluster is represented as a real number between 0 and 1, such that the sum of all membership values for an example sums to 1. For more details on fuzzy clustering, please refer to [3].

Bagging, or bootstrap aggregation, is a method of randomly sampling (with replacement) subsets of the dataset and learning a model/classifier on each subset independently [5].

Therefore, we consider it as another approach to partitioning a dataset. Note, it does not create disjoint partitions. Bagging has been shown to often improve classifier performance vs. learning a classifier over the entire dataset directly [5, 16, 1, 9]. As stated earlier, we wish to use the body of machine learning developed for moderate-sized datasets. Thus, bagging can be employed on each subset in a disjoint partition of the data. We would expect the bagged ensemble of classifiers for a subset to improve classification performance over an individual classifier.

This paper introduces two key points. First, we find that intelligently partitioning the dataset can yield better performance than random partitioning, which is not surprising. We also find that while learning classifiers on disjoint subsets of data is no better than building a single classifier using the entire dataset, the use of bagging on these subsets can actually improve classification accuracy over the use of a single classifier built on all the data. In distributed learning environments, we seek a solution that does not require a classifier to be created from the entire dataset, only from "appropriately" chosen subsets. Thus, we believe this approach is an attractive solution to increasingly large machine learning problems. The next section describes in detail the approach used for both random and intelligent partitioning. Experimental results on a set of common machine learning datasets are then presented to demonstrate the approach.

3 Method

We describe in detail below each of the methods used to partition a dataset into subsets from which an ensemble of classifiers can be built. Two different methods of random partitioning were considered, as was an intelligent approach of partitioning using clustering. Classifier combination is then considered, including simple majority and weighted majority voting. Finally, a method of combining the ensemble of classifiers, each learned using a different subset of a disjoint partition, and bagging at the subset level is described.

In all instances, the ensemble of classifiers are composed of decision trees. C4.5 release 8 [17] is used to learn a decision tree on each partition of training data.

3.1 Random Disjoint Partitions

One of the simplest data partitioning approaches is to separate the dataset into n random subsets. The partitioning is done without respect to the class distribution within the dataset. Each disjoint subset is independently used in the generation of a decision tree classifier and the classifier predictions are combined using a simple majority vote. This approach is well suited to distributed learning, since the entire dataset is never required to be loaded in memory at one time. Examples can be randomly chosen and distributed across a set of processors.

The number of subsets (n) is the only parameter required for this algorithm. Larger numbers of subsets with moderate sized datasets will almost certainly lead to data starvation,

thereby limiting the amount of learning possible from a subset and, also the ensemble. An algorithmic method of selecting n could be considered by identifying the number of processors in hand, the memory limitations of each processor and giving each processor enough data to fit in its memory; however, this is more applicable to extremely large datasets.

3.2 Stratified Random Disjoint Partitions

One obvious difficulty with random disjoint partitions is that the class distributions are not necessarily maintained across subsets. Some subsets may not even contain instances of a particular class, skewing the learned classifiers for that subsets. Therefore, we considered a stratified random disjoint partition approach in which each subset contains examples according to the dataset class distribution. The within-class choices are still random, however we preserve the class characteristics of the original data. For example, if the original dataset had two classes with a 2:1 ratio, then the same ratio of 2:1 is maintained in each of the subsets. Each subset is used in classifier generation and classifier predictions are combined using a simple majority vote.

Since the partitioned subsets are stratified, class distributions must be maintained. Thus, this approach requires an initial pass over the dataset to calculate class frequencies. Although it is a single-pass, sequential operation, for very large datasets the approach may not be suitable.

3.3 Clustering to create Partitions

Fuzzy c-means clustering is used to examine the effects of intelligent partitioning of a dataset. A cluster-splitting FCM algorithm is applied to the dataset in order to create meaningful partitions of the data. Several issues arise from the use of FCM, including the determination of an appropriate number of clusters (subsets) and the computational cost of FCM.

The algorithm begins with $c = 2$ and clusters until the membership values are stable. The validity of the partition is evaluated using the Xie-Beni partition validity metric [18]. Assuming the stopping conditions (described later) are not met, a cluster is selected to be split. The "worst" cluster, as measured by a cluster validity metric, is split into two distinct clusters (see [2] for more details).

Since the number of clusters is not known, values of c from 2 to 25 were tried by using the cluster splitting process above to increase the number of clusters. This process is terminated early if the partition validity after clustering is worse than 5 times the best partition validity. This number was empirically observed to prune the search well, since a bad cluster split could almost never be improved by successive splits. Once the algorithm finishes clustering, the FCM step is repeated a final time with the best c found. FCM produces a membership matrix, representing the membership of each example in each cluster. A maximum membership function is applied to assign each example to a single cluster (i.e. hardening the fuzzy clusters), thereby maintaining the disjoint partition aspect of the algorithm.

Clustering is often a computationally intensive approach to partitioning a dataset. Some approaches to clustering require examining all pair-wise distances between examples. This paper addresses the use of partitioning in large datasets, therefore computational complexity is of crucial concern. Hard c-means clustering[12] requires less memory and processing requirements vs. FCM. Rather than store an entire membership vector for each dataset example, HCM can simply store a pointer to the cluster to which an example currently belongs. Additionally, fuzzy clustering provides the ability for every example to contribute to every cluster centroid (to various degrees). Thus, the amount of processing required per cluster is larger in FCM. Although the partitions created by HCM and FCM are generally different given the same intializations, they are overall similar in results.

3.4 Bagging

Bagging is another method of partitioning a dataset, however the partitions created are not disjoint. In our bagging approach, n bags (approaches) are created by sampling (with replacement) until a bag of P% (P=80 here) of the dataset is obtained. Each bagged train set is independently used to generate a classifier and classifier predictions are combined using a simple majority vote. This approach requires more computational overhead than random partitioning. In bagging, we create 50 different subset samples of the dataset – this requires accessing the entire dataset many more times than a simple one-pass random partition.

3.5 Classifier Combination

Once the dataset is partitioned and individual classifiers are trained on the partitions, we must consider methods of combining the classification output. Given a number of independently learned decision trees, we must combine their knowledge effectively. In the case of random partitions and bagging, a simple majority vote is reasonable [4, 11] and, as we will show, effective. More intelligent methods of classifier combination are possible using clustering: ask-expert voting and fuzzy voting.

The ask-expert method of voting is analagous to the mixture of experts approach described earlier. The cluster centroids created during the clustering process are stored. When a test example is presented to the ensemble of classifiers, the distance between each cluster centroid and a test example is found. The cluster corresponding to the closest cluster centroid is the region of example space in which the test example exists. Recall that the data of this cluster was "learned" by a single decision tree. We use only this decision tree, which is the "expert" on classifying instances in this region of example space.

Fuzzy voting is another intelligent technique for classifier combination. Recall that fuzzy clustering generates membership values for an example in all of the clusters. Given a test example, it's membership values can be calculated using the cluster centroids. These membership values can then be used to weight each classifier's output. Thus, the ensemble classification is a weighted (by fuzzy membership) combination of individual classifiers.

3.6 Bagging Ensembles of Classifiers

Once a dataset is partitioned, each cluster or disjoint subset represents a distinct training set. Thus, we could employ bagging as a method of improving individual classifier performance for a given subset. In particular, bagging of random partitioned subsets vs. bagging of clusters from clustered partitions may yield some insights into the phenomenon of bagging. Clustering generates localized classifiers, or classifiers able to accurately classify only a localized region of example space. Therefore, bagging in this context generates localized random samples. Can localized random views (i.e. localized bagging) of the data affect classifier performance similar to traditional bagging?

We investigate localized bagging by sampling the generated partitions at 80% of the partition size. Fifty bags are used per subset, generating a significant amount of diversity across bags. Voting within each bagged subset was done via simple majority among the 50 created bags. This approach was used with both clustering and random-partition classifier ensembles. For example, consider a dataset D of 200 items. When the dataset is randomly partitioned with $n = 2$, each subset D_i consists of 100 randomly chosen examples. Each 100-member subset D_i is then sampled (with replacement) 50 times. Each bag D_i^b consists of 80 sampled elements of subset D_i. All 50 bags D_i^b classify an unseen example and a majority decision is reached for subset D_i. Then a vote is taken among subsets D_i, and a final decision is made.

4 Experiments

We evaluate the proposed approaches to learning by experiments on 9 well-known machine learning datasets. See Table 1 for dataset details. In all experiments, 10-fold cross validation is used. Results are reported as the mean classification performance over the 10 folds. The number of clusters found by FCM was taken as the number of random partitions (stratified and random-disjoint) to create. Comparisons between methods is done via a two-tailed paired two sample for means t-test among fold results, setting the confidence level, $\alpha = 0.025$.

4.1 Voting Methods in Cluster Partitioning

Two methods of classifier voting when using FCM partitioning were discussed in the previous section. It was expected that the use of more information, in the form of fuzzy voting, would be a useful performance improvement. However, as can be seen in Table 2, the voting method did not affect the overall classification rate. Further investigation indicates that the clustering process often resulted in very crisp partitions. When the partitions are crisp, fuzzy voting is essentially the ask-expert method. Virtually all of the non-zero membership value is from the closest cluster. These results indicate that the simpler ask-expert method is as effective as fuzzy voting. In addition, this method requires only a single classifier (i.e. the expert) to classify the text example. Therefore, further experiments use the ask-expert

Dataset	Source	Size	Number of Features	Number of Classes
Iris	UCI	150	4	3
Pima Indian Diabetes	UCI	768	8	2
Page-block	UCI	5473	10	5
Phoneme	Elena	5,404	5	2
Satimage	UCI	6,435	35	6
Pendigits	UCI	10,992	16	10
Mammography	Local-USF	11,183	6	2
Letter	UCI	20,000	16	26
Shuttle	UCI	43,500	9	7

Table 1: Datasets used, with size and number of classes

method of voting.

4.2 Ensembles of Classifiers

The results of training and testing ensembles of classifiers according to the several partitioning methods described above can be seen in Table 2. In all datasets except Letter, there was no statistical significance between the clustering method and a C4.5 decision tree learned from the entire dataset. However, random partitioning performed statistically worse than C4.5 (single decision tree from the entire dataset) in 4 of the 9 datasets.

An interesting result that can be seen from Table 2 is that the stratified and random disjoint partitions perform very similar. In all but the letter dataset, the random disjoint ensemble was slightly better. This surprising result indicates that carefully preserving class distributions in partitions is not always necessary. Random partitioning avoids a pass over the dataset to collect class frequencies and can result in much faster data partitioning. As a result, further experiments did not consider the more expensive stratified disjoint partitioning strategy.

4.3 Bagging

The next phase of experiments was to investigate the bagging phenomenon within our ensembles of classifiers. The resulting clusters and partitions from Table 2 were bagged using the algorithm described in the previous section. As can be seen from Table 3, bagging generally improves the overall performance vs. C4.5 applied to the entire dataset. This is a surprising result, since increased partition performance does not necessarily imply overall increased performance. Most significantly, a random partition of a dataset, when combined with bagging performs better than a single decision tree learning the entire dataset. We

Dataset	Number of Partitions/ Clusters	Full C4.5	Stratified Disjoint	Random Disjoint	FCM Cluster *Ask-Expert*	FCM Cluster *Fuzzy Voting*
Iris	3	95.30	94.00	94.00	94.00	94.00
Pima	2	73.90	72.52	74.21	73.82	73.82
Page-block	2	96.90	96.90	96.82	96.95	96.95
Phoneme	5	86.50+	82.68	83.44	85.99	86.01
Satimage	9	86.30	87.61	87.44+	86.01	86.01
Pendigits	4	96.57+	96.15	96.06	96.42	96.49
Mammography	2	98.50	98.43	98.51	98.40	98.40
Letter	2	88.10*+	83.52	83.54	86.08	86.22
Shuttle	3	99.96+	99.92	99.92	99.95	99.95

Table 2: Partitioning Results vs. C4.5. The '+' represents C4.5-Random statistically significant winner. The '*' represents C4.5-Cluster statistically significant winner.

also tried 50 bags of 100% size with Iris and Pima, due to the much smaller size. We found that average accuracy over 10 folds for Iris increased to 95.3%; there was no improvement observed with Pima. To be consistent, we maintained 80% bag sizes with all the datasets, as for very large datasets it may not be possible to bag at 100% or more.

5 Conclusion

In this paper, we present a novel approach to distributed learning using clustering. This intelligent method of partitioning a dataset is compared to simpler, random methods of partitioning. In general, intelligent partitioning of a dataset provides better performance than random partitioning, and generally performs as well as C4.5 over the entire dataset. The results presented in this paper suggest that for very large datasets, the creation of ensembles of classifiers can perform reasonably well. Note that for some of these datasets partitioning causes a learning algorithm to be data starved for one or more classes; with train datasets that do not fit in memory this effect would likely not be observed. This is important for extremely large datasets that cannot be learned by a single classifier; distributed learning can be used in these instances. Ideally, we would like to look at generating n partitions a dataset using HCM or random selection, and chart out the performance. However, due to space limitations, we have not included that set of experiments in this paper.

Interestingly, our results indicate that bagging of individual partitions can yield better results than learning from the entire dataset. It is surprising as bagging, in effect, sees much less data due to bagging of smaller sized data subsets. Even in the case of random partitioning, where any individual classifier created on a subset often performs significantly

Dataset	C4.5	Bagging 50 bags	Random-Disjoint Bag 50 bags	Cluster Bag 50 bags
Iris	95.30	94.67	94.00	93.33
Pima	73.90	76.30	75.78	77.34
Page-block	96.90	97.55	97.11	97.26*
Phoneme	86.50	89.15	85.77	88.71*
Satimage	86.30	90.89	87.61+	86.76
Pendigits	96.57	98.42	97.22+	98.18*
Mammography	98.50	98.76	98.52	98.78*
Letter	88.10	93.54	90.82+	93.01*
Shuttle	99.96*+	99.94	99.89	99.93

Table 3: Bagging Results. The '+' represents C4.5-Random statistically significant winner.

The '*' represents C4.5-Cluster statistically significant winner.

worse than a single classifier learned on the entire dataset, bagging of disjoint subsets can improve performance. We believe this is due to the same effects that cause bagging to improve performance in general - bagging produces diverse classifiers from the data partitions, despite the smaller number of examples within a partition. We have thus proposed a novel and effective three-stage learning technique - partition, bag each partitioned subset, and learn.

6 Acknowledgements

This research was partially supported by the United States Department of Energy through the Sandia National Laboratories ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789 and by Tripos, Inc. We would like to thank Kevin Bowyer and Philip Kegelmeyer for various insightful discussions during our research. We would also like to thank Philip Kegelmeyer for his bagging script.

References

[1] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1,2), 1999.

[2] Amine M. Bensaid, L. O. Hall, et al. Validity-guided (re)clustering with applications to image segmentation. *IEEE Transactions on Fuzzy Systems*, 4(2):112–123, May 1996.

[3] James C. Bezdek and Sankar K. Pal, editors. *Fuzzy Models For Pattern Recognition.* IEEE Press, New Jersey, 1991.

[4] K.W. Bowyer, N.V. Chawla, T.E. Moore, L.O. Hall, and W.P Kegelmeyer. A parallel decision tree builder for mining very large visualization datasets. In *Proceedings of IEEE-System, Man and Cybernetics*, 2000.

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

[6] P. Chan and S. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, pages 39–44, 1995.

[7] P. Chan and S. Stolfo. Scaling leaning by meta-learning over disjoint and partially replicated data. In *9th Florida Artificial Intelligence Research Symposium*, pages 151–155, 1996.

[8] Ke Chen, Dahong Xie, and Huisheng Chi. A modified hme architecture for text-dependent speaker identification. *IEEE Transactions on Neural Networks*, 7(5), Sep 1996.

[9] T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[10] L.O. Hall, K.W. Bowyer, W.P Kegelmeyer, T.E. Moore, and C. Chao. Distributed learning on very large data set. In *Workshop of Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[11] L.O. Hall, N.V. Chawla, K.W. Bowyer, and W.P Kegelmeyer. Learning rules from distributed data. In *Workshop of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[13] Perry Moerland. Classfication using localized mixtures of experts. In *Proceedings of the 1999, 9th International Conference on Artificial Neural Networks (ICANN99)*, Edinborough, 1999.

[14] D.N Provost, F.J amd Hennessy. Scaling up: Distribtuted machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI '96*, pages 74–79, 1996.

[15] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, 1999.

[16] J. R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[17] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

[18] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, August 1991.