The Distributed Boosting Algorithm

Aleksandar Lazarevic
Center for Information Science and Technology
Temple University, Room 303A, Wachman Hall (038-24)
1805 N. Broad St., Philadelphia, PA 19122, USA
phone: +1-215-204-5908, fax: +1-215-204-5082
aleks@ist.temple.edu

Zoran Obradovic
Center for Information Science and Technology
Temple University, Room 303, Wachman Hall (038-24)
1805 N. Broad St., Philadelphia, PA 19122, USA
phone: +1-215-204-6265, fax: +1-215-204-5082
zoran@ist.temple.edu

ABSTRACT
In this paper, we propose a general framework for distributed boosting intended for efficient integrating specialized classifiers learned over very large and distributed homogeneous databases that cannot be merged at a single location. Our distributed boosting algorithm can also be used as a parallel classification technique, where a massive database that cannot fit into main computer memory is partitioned into disjoint subsets for a more efficient analysis. In the proposed method, at each boosting round the classifiers are first learned from disjoint datasets and then exchanged amongst the sites. Finally the classifiers are combined into a weighted voting ensemble on each disjoint data set. The ensemble that is applied to an unseen test set represents an ensemble of ensembles built on all distributed sites. In experiments performed on four large data sets the proposed distributed boosting method achieved classification accuracy comparable or even slightly better than the standard boosting algorithm while requiring less memory and less computational time. In addition, the communication overhead of the distributed boosting algorithm is very small making it a viable alternative to the standard boosting for large-scale databases.

Keywords
Boosting, distributed learning, classifier ensembles.

1. INTRODUCTION
The number and the size of databases are rapidly growing in various business and scientific fields thus resulting in an exceptional opportunity to develop automated data mining techniques for extracting useful knowledge from massive data sets. This problem may be further complicated by the fact that in many cases, the databases are located at multiple distributed sites. Data may be distributed across a set of sites or computers for several reasons. For example, several data sets concerning business information (e.g. telephone or credit card fraud) might be owned by separate organizations that have competitive reasons for keeping the data private. In addition, these data may be physically dispersed over many different geographic locations. However, business organizations may be interested in enhancing their own models by exchanging useful information about the data.

KDD 2001, August 2001, San Francisco, CA.

In this paper, we propose a novel technique of combining classifiers from multiple sites using a boosting technique [6]. Boosting uses adaptive sampling of patterns to generate a highly accurate ensemble of many weak classifiers whose individual global accuracy is only moderate. In boosting, the classifiers in the ensemble are trained serially, with the weights on the training instances adjusted adaptively according to the performance of the previous classifiers. The main idea is that the classification algorithm should concentrate on the instances that are difficult to learn.

Our distributed boosting algorithm is designed for learning when disjoint data sets from multiple sites cannot be merged together. However, it can also be applied to parallel learning, where the huge training data set is split into several sets that reside on a parallel computer with several processors. In the proposed method, the classifiers are first learned from disjoint datasets at each boosting round and then exchanged amongst the sites. The exchanged classifiers are then combined, and finally, their weighted voting ensemble is constructed on each disjoint data set. The ensemble that is applied to an unseen test set represents an ensemble of ensembles built locally on all distributed sites. The performance of ensembles is used to update the probabilities of drawing the data samples in succeeding boosting iterations. Our experimental results indicate that this method is computationally effective and comparable to or even slightly better in achieved accuracy than when boosting is applied to the centralized data.

2. RELATED WORK
To solve the problem of learning from very large and distributed databases, some researchers have proposed incremental learning techniques, usually involving direct modifications of standard learning algorithms, such as decision trees [12] and rule learner [4]. An alternative and fairly general method for distributed learning is to combine different multiple predictors in a "black-box" manner. Different meta-learning techniques explored at the Jam project [3] were proposed in order to coalesce the predictions of classifiers trained from different partitions of the training set. Similarly, a knowledge probing approach [7] for distributed learning from homogeneous data sites in the first phase learns a set of base classifiers in parallel, and in the second, the meta-learning is applied to combine the base classifiers. The advantage of the meta-learning approach is that it is algorithm-independent, it can be used to scale up many learning algorithms, and it ensures the privacy of data at multiple sites.

Recently, boosting has received extensive theoretical and empirical study, but most of the published work focuses on improving the accuracy of a classifier over the same single, centralized data set that is small enough to fit into the main computer memory. So far, there has not been much research on using the boosting tech-

nique for distributed learning. The only exception was boosting for scalable and distributed learning [5], where each classifier was trained using a small fraction of the training set. In this distributed version, the classifiers were trained either from random samples (*r-sampling*) or from disjoint partitions of the data set (*d-sampling*). In *r-sampling*, a fixed number of examples were randomly picked from the weighted training set (without replacement), where all examples had equal chance of being selected. In *d-sampling*, the weighted training set was partitioned into a number of disjoint subsets, where the data from each site was taken as a *d*-sample. At each round, a different *d*-sample was given to the weak learner. Both methods can be used for learning over very large data sets, but *d-sampling* is more appropriate for distributed learning, where data at multiple sites cannot be pulled together to a single site. The reported experimental results indicated that their distributed boosting is either comparable to or better than learning single classifiers over the complete training set, but only in some cases comparable to boosting over the complete data set.

3. METHODOLOGY

The modifications of the boosting algorithm that we propose here are variants of the AdaBoost.M2 procedure [6], which proceeds in a series of T rounds. In every round t, a weak learning algorithm is called and presented with a different distribution D_t that is altered by emphasizing particular training examples. The distribution is updated to give wrong classifications higher weights than correct classifications. The entire weighted training set is given to the weak learner to compute the weak hypothesis h_t. At the end, all weak hypotheses are combined into a final hypothesis h_{fn}.

The boosting algorithm may be appropriate for distributed learning for several reasons: it can be applied to a wide variety of algorithms, it is superior to other combining methods and its weighted voting ensemble can easily scale the magnitudes of classifiers giving a large weight to a strong hypothesis thus correcting wrong classifications of weaker hypotheses. In addition, a natural way of learning in a distributed environment is by combining classification predictors. Our aim, hence, is to exploit all of these advantages in order to apply boosting to distributed learning.

As classifiers, we trained multilayer (2-layered) feedforward neural network models with the number of hidden neurons equal to the number of input attributes, and with the number of output nodes equal to the number of classes, where the predicted class is from the output with the largest response. We used two learning algorithms: resilient propagation [11] and Levenberg-Marquardt [8].

3.1 The Framework for Distributed Learning

The objective of our distributed boosting algorithm is to efficiently construct a prediction model using data at multiple sites such that the prediction accuracy is similar to boosting when all the data are centralized at a single site. Towards such an objective, we propose several modifications of the boosting algorithm within the general framework presented at Figure 1. All distributed sites perform the learning procedure at the same time.

Assume there are k distributed sites, where site j contains set S_j with m_j examples, $j = 1,\ldots k$. Data sets S_j contain the same attributes and do not necessarily have the same size. During the boosting rounds, site j maintains a local distribution $\Delta_{j,t}$ and local weights $w_{j,t}$ that directly reflect the prediction accuracy on that site.

- On site j, ($j = 1\ldots k$) we are given set S_j $\{(x_{j,1}, y_{j,1}), \ldots ,(x_{j,m_j}, y_{j,m_j})\}$ $x_{j,i} \in \mathrm{X_j}$, with labels $y_{j,i} \in \mathrm{Y_j} = \{1, \ldots, \mathrm{C}\}$, $j = 1\ldots k$. Let $B_j = \{(i,y_j): i = 1, \ldots, m_j, y_j \neq y_{j,i}\}$.
- On each site j initialize the distribution $\Delta_{j,1}$ over the examples, such that $\Delta_{j,1}(i) = 1/m_j$, and compute the sums $\sum_{i \in S_j} \Delta_{j,1}(i)$ of all the elements in distributions $\Delta_{j,1}$.
- Each site j broadcasts the computed sums and makes a version of the global distribution $D_{j,1}$, by initializing the j-th interval $[\sum_{p=1}^{j-1} m_p + 1, \sum_{p=1}^{j} m_p]$ in the distribution $D_{j,1}$ with values $1/m_j$.
- Each site renormalizes the $D_{j,1}$ with a normalization factor such that $D_{j,1}$ is a distribution.
- For $j = 1 \ldots k$ (For all distributed sites)
  - For $t = 1, 2, 3, 4, \ldots T$
    1. Draw the indices of the examples according to the distribution $D_{j,t}$ and make a sample $Q_{j,t}$ from the instances whose indices belong to the j-th interval of $D_{j,t}$.
    2. Train a weak learner $L_{j,t}$ on the sample $Q_{j,t}$.
    3. Broadcast a classifier learner $L_{j,t}$ to all distributed sites.
    4. Create an ensemble $E_{j,t}$ by combining the learners $L_{j,t}, j = 1\ldots k$ from all distributed sites.
    5. Using the ensemble $E_{j,t}$ compute weak hypothesis $h_{j,t}$: $\mathrm{X} \times \mathrm{Y} \rightarrow [0, 1]$.
    6. Compute the pseudo-loss of hypothesis $h_{j,t}$: $$\varepsilon_{j,t} = \frac{1}{2} \cdot \sum_{(i,y) \in B_j} \Delta_{j,t}(i,y)(1 - h_{j,t}(x_{j,i}, y_{j,i}) + h_{j,t}(x_{j,i}, y_j))$$
    7. Set $\beta_{j,t} = \varepsilon_{j,t} / (1 - \varepsilon_{j,t})$
    8. Compute $w_{j,t}(i,y_j) = (1/2) \cdot (1 - h_{j,t}(x_{j,i}, y_j) + h_{j,t}(x_{j,i}, y_{i,j}) / acc_j^p$, $p \in \{0, 1, 2\}$
    9. Compute $V_{j,t} = \sum_{(i,y_j) \in B_j} w_{j,t}(i, y_j)$ and broadcast it to all distributed sites.
    10. Create a weight vector $U_{j,t}$, such that the j-th interval $[\sum_{p=1}^{j-1} m_p + 1, \sum_{p=1}^{j} m_p]$ is the weight vector $w_{j,t}$, while the values in the q-th interval, $q \neq j$, $q = 1, \ldots, k$ are set to $V_{q,t}/m_q$.
    11. Update $D_{j,t}$: $D_{j,t+1}(i, y_j) = (D_{j,t}(i, y_j) / Z_{j,t}) \cdot \beta_{j,t}^{U_{j,t}(i,y_j)}$, where $Z_{j,t}$ is a normalization constant chosen such that $D_{j,t+1}$ is a distribution. The values in the j-th interval of the $\mathrm{D_{j,t}}$ after normalization make the local distribution $\Delta_{j,t}$.
- Output the final hypothesis: $$h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} \sum_{j=1}^{k} (\log \frac{1}{\beta_{j,t}}) \cdot h_{j,t}(x, y_j)$$

Figure1. The distributed boosting framework

However, our goal is to emulate the global distribution D_t obtained through iterations when standard boosting is applied to a single data set obtained by merging all sets from distributed sites. In order to create such a distribution that will result in similar sampling as when all data are centralized, the weight vectors $w_{j,t}, j = 1,\ldots,k$ from all distributed sites are merged into a joint weight

vector w_t, such that the q-th interval of indices $[\sum_{p=1}^{q-1} m_p + 1, \sum_{p=1}^{q} m_p]$ in the weight vector w_t corresponds to the weight vector $w_{q,t}$ from the q-th site. The weight vector w_t is used to update the global distribution D_t as in step 5 at Figure 1. However, merging all the weight vectors $w_{j,t}$ requires a huge amount of time for broadcasting, since they directly depend on the size of the distributed data sets. In order to reduce this transfer time, instead of the entire weight vectors $w_{j,t}$, only the sums $V_{j,t}$ of all their elements are broadcast (step 9 in Figure 1). Since data site j samples only from set S_j, there is no need to know exact values of the elements in the weight vectors $w_{q,t}$ ($q \neq j$, $q = 1, \ldots k$) from other distributed sites. Instead, it is sufficient to know only how many data examples need to be sampled from the site q.

Therefore, each site j creates a weight vector $U_{j,t}$ (step 10, Figure 1), where its j-th interval $[\sum_{p=1}^{j-1} m_p + 1, \sum_{p=1}^{j} m_p]$ represents the weight vector $w_{j,t}$, while all the other intervals that correspond to the weight vectors from other distributed sites may be set arbitrarily such that the values inside the q-th interval of indices ($q \neq j$) sum to the value $V_{q,t}$. The simplest method is to set all values in the q-th interval to the value $V_{q,t}/m_q$. Using this method, expensive broadcasting of the huge weight vectors is avoided, while still preserving the information which site is more difficult to learn and where more examples need to be sampled.

As a result, each site at round t maintains its version $D_{j,t}$ of the global distribution D_t, and its local distribution $\Delta_{j,t}$. At each site j, the samples in boosting rounds are drawn according to the distribution $D_{j,t}$, but the sampled training set $Q_{j,t}$ for site j is created only from those data points that match the indices drawn from the j-th interval in the distribution $D_{j,t}$ (step 1, Figure 1). The classifiers $L_{j,t}$ are constructed on each of the samples $Q_{j,t}$ and then exchanged among the distributed sites at each boosting round t. Since all sites contain a set of classifiers $L_{j,t}$, $j = 1 \ldots k$, the next steps involve creating an ensemble $E_{j,t}$ by combining these classifiers and computing a composite hypothesis $h_{j,t}$. The local weight vectors $w_{j,t}$ are updated at each site j in order to give wrong classifications higher weights than correct classifications (step 8, Figure 1) and then their sums $V_{j,t}$ are broadcast to all distributed sites. Each site j updates its local version $D_{j,t}$ according to the created weight vector $U_{j,t}$. At the end, the composite hypotheses $h_{j,t}$ from different sites and different boosting iterations are combined into a final hypothesis h_{fn}.

3.2 The Variants of Distributed Boosting

We explore several variants of the proposed distributed boosting algorithm from Figure 1. The algorithms differ in (a) the method for combining the classifiers into an ensemble $E_{j,t}$ (step 4), (b) computing a representative hypothesis $h_{j,t}$ (step 5) and (c) updating the weights $w_{j,t}$ (step 8).

In the first distributed learning algorithm, denoted as *Competing Classifiers from Distributed Sites*, the learned classifiers $L_{j,t}$ from all distributed sites are combined such that each data instance on a local site is assigned to the classifier with the highest prediction confidence on that data pattern. As a result, the composite hypothesis $h_{j,t}$ uses a different classifier $L_{j,t}$ for each data example.

Unlike competing classifiers, other distributed learning methods involve combining classifiers in order to create an ensemble $E_{j,t}$ and hypothesis $h_{j,t}$. The simplest combining method is based on *Simple Majority Voting of Classifiers*. If the classifiers $L_{l,t}$, $l = 1,\ldots,k$, from all sites produce hypotheses $h_{l,j,t}$ on site j, then the hypothesis $h_{j,t}$ (step 5, Figure 1) is computed as:

$$h_{j,t} = \frac{1}{k} \sum_{l=1}^{k} h_{l,j,t}$$

More sophisticated techniques for distributed learning consider weighted combinations of classifiers. In *Weighted Majority Voting of Classifiers*, the weights $u_{l,j,t}$ of the classifiers $L_{l,t}$ from all sites are proportional to the accuracy they achieve on the local site j. Therefore, if the classifiers $L_{l,t}$ produce hypotheses $h_{l,j,t}$ on site j, then the hypothesis $h_{j,t}$ can be computed as:

$$h_{j,t} = \left(\sum_{l=1}^{k} u_{l,j,t} \cdot h_{l,j,t}\right) \Big/ \sum_{l=1}^{k} u_{l,j,t}$$

In *Confidence-based Weighting Method*, the classifiers from all sites are combined using the procedure similar to the boosting technique. If the classifiers $L_{l,t}$ at iteration t produce hypotheses $h_{l,j,t}$ on site j that maintains the distribution $\Delta_{j,t}$, then this technique of combining classifiers is defined at Figure 2.

$$\phi_{l,t} = \frac{1}{2} \cdot \sum_{(i,y) \in S_j} \Delta_{j,t}(i,y)(1 - h_{l,j,t}(x_{j,i}, y_{j,i}) + h_{l,j,t}(x_{j,i}, y_j))$$

$$\gamma_{l,t} = \phi_{l,t} / (1 - \phi_{l,t})$$

$$h_{j,t} = \arg\max_{y_j \in Y_j} \sum_{l=1}^{k} (\log \frac{1}{\gamma_{l,t}}) \cdot h_{l,j,t}$$

Figure 2. The confidence based technique for weighted combining classifiers from distributed sites

In order to further emphasize sampling from sites that are difficult for learning, dividing the weights $w_{j,t}$ by the factor acc_j^p ($p = 0, 1$ or 2) is considered, such that the difference between the weights from two sites is further increased. Here, acc_j corresponds to the local accuracy on corresponding site j, and the factor p indicates how much we like to increase the difference between the weights from different sites. All techniques for updating the weights $w_{j,t}$ are also integrated in all methods for distributed boosting involving combining learners.

4. EXPERIMENTAL RESULTS

Our experiments were performed on several data collections. The first one contained two synthetic spatial data sets with 6561 instances generated using our spatial data simulator [10] such that the generated data resembled statistics of real-life spatial data. One data set was used for training and another one for out of sample testing. Since random splitting for spatial domains likely results in overly optimistic estimates of prediction error (due to spatial correlation in data), the training data set was spatially split into 3 disjoint data sets, each with 2187 examples. The obtained spatial data sets stemmed from similar homogeneous distributions and had 5 continuous attributes and 3 equal size classes.

The other three data collections were Waveform, LED and Covertype data sets from the UCI repository [2]. For the Waveform set, 50000 instances with 21 continuous attributes and three equally sized classes were generated. The generated data were randomly

split into five sets of 10000 examples each, where four of them were used for distributed learning, and the fifth data set was used as a test set. The LED data set was generated with 10000 examples and 10 classes, where four sets with 1500 examples were used for training in a distributed environment, and the set with 4000 examples was used for testing. The Covertype data set, currently one of the largest databases in the UCI Database Repository, contains 581012 examples with 54 attributes and 7 target classes representing the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System [1]. In Covertype data set, 40 attributes are binary columns representing soil type, 4 attributes are binary columns representing wilderness area, and the remaining 10 are continuous topographical attributes. Since the training of neural network classifier would be very slow if using all 40 attributes representing a soil type variable, we transformed them into 7 new ordered attributes. These 7 attributes were determined by computing the relative frequencies of each of 7 classes in each of 40 soil types. Therefore, we used a 7-dimensional vector with values that could be considered continuous and therefore more appropriate for use with neural networks. This resulted in the transformed data set with 21 attributes. The 149982 data instances separated into 8 disjoint data sets were used for distributed learning, while the 431032 data examples were used for out of sample testing.

4.1 Time Complexity Analysis

The major advantage of the proposed distributed boosting algorithm is that it requires significantly less computational time per each boosting round since the classifiers are learned on smaller data sets. Figure 3 shows how the time required for training neural networks (NN) depends on the number of examples in the training set for all four reported data sets when measured on a *Pentium III* processor with 768 MB of main memory. Analyzing the Figure 3a, it is evident that the time needed for constructing a NN classifier on the three times reduced synthetic spatial training set resulted in more than three times faster computing time, while for LED and Waveform data sets, four times smaller data set caused more than four times faster learning (Figure 3b, 3c). Finally, for Covertype data set, time needed for training a NN on an eight times smaller data set was more than eight times smaller than time required for training a NN when using the entire training set (Figure 3d).

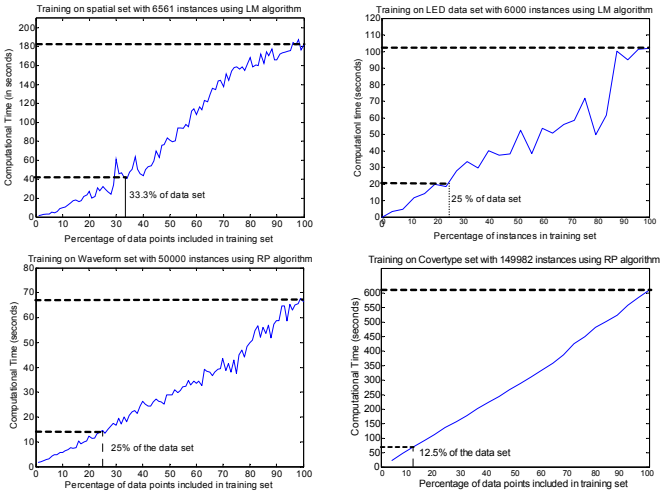

Figure 3. The time needed for learning neural network (NN) classifiers for different sizes of four different data sets

In order to estimate the speedup of the proposed distributed boosting algorithm, we need to consider a communication overhead that involves time required for broadcasting the NN classifiers and the sums $V_{j,t}$ of the weight vectors $w_{j,t}$ to all sites. The size of the NN classifiers is directly proportional to the number of input, hidden and output nodes, and is relatively small in practice. (e.g., our implementation of a two-layered feedforward NN with 5 input and 5 hidden nodes required only a few KB of memory). The broadcasting of such small classifiers results in very small communication overhead, and when the number of the distributed sites grows, time needed for broadcasting increases linearly. However, the true estimate of the communication overhead among the distributed sites depends on the actual implementation of the communication amongst them. Assuming that the communication overhead for small number of distributed sites is negligible comparing to the time needed for training a NN classifier, the proposed distributed boosting algorithm achieves a linear speedup (Figure 4). The scale up is usually measured when increasing the number of sites and keeping the number of data examples per site constant. It is obvious that in such situation, time needed for training NN classifiers on distributed sites is always the same regardless of the number of sites. The only variable component is the communication overhead that is negligible for small number of sites (up to 10). Therefore it is apparent that the achieved scale up is close to linear.

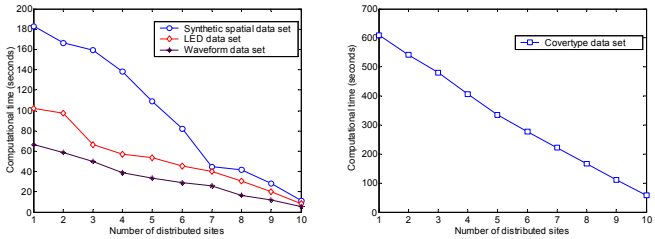

Figure 4. The speedup of the distributed boosting algorithm for different data sets

4.2 Prediction Accuracy Comparison

To explore whether our distributed boosting can reach similar prediction accuracy as the standard boosting algorithm on a centralized data set, experiments were first performed on three disjoint synthetic spatial data sets using competing classifiers among sites, simple majority and weighted majority algorithms (Figure 5a). For each of the graphs shown at Figure 5a, the results were achieved for $p = 0$, as in most of the cases this modification was more accurate than if using $p = 1$ or $p = 2$ for dividing the weight vector $w_{j,t}$ by the factor acc^p. To investigate how the performance of distributed boosting varied with the number of data points used for learning, changing the size of the data sets on distributed sites was used (Figure 5). The three disjoint data sets used for training in distributed boosting were merged into a centralized training data set for standard boosting. All boosting methods were tested on the same data set with 6561 instances.

When applying the distributed boosting to Covertype data set, eight disjoint, equally-sized data sets were used for learning, while the data set with 431032 examples was used for out of sample testing (Figure 5b). For experiments performed on LED and Waveform data sets, four disjoint, equally-sized data sets were used for learning and unseen data sets with 4000 and 10000 patterns were used for testing respectively on LED and Waveform data sets. However, the experiments performed on these data sets

showed similar prediction accuracy for all proposed variants of boosting algorithm probably due to high homogeneity of data (Figure 5c, 5d).

Results from experiments performed on the synthetic spatial data sets indicate that the methods of voting the classifiers constructed on multiple sites achieved approximately the same classification accuracies as the standard boosting algorithm on merged data (Figure 5a), while the method of competing classifiers was always significantly less accurate than standard boosting (Figure 5a). This was probably due to the fact that none of the classifiers constructed on the multiple sites were sufficiently competent for prediction on the unseen test set, and the prediction results were comparable or even slightly worse than when making predictions from a single distributed site (Figure 5a).

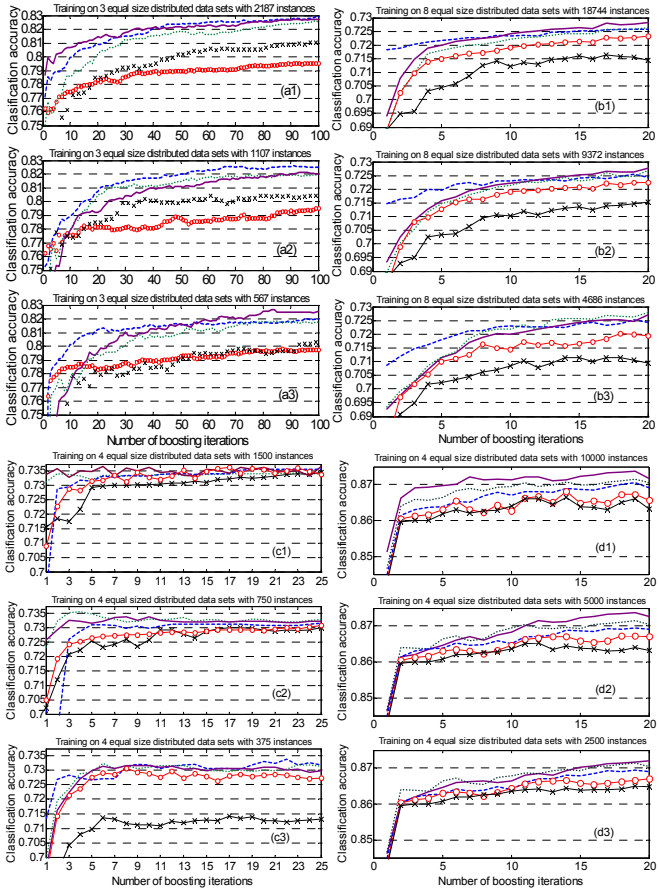

Figure 5. Out of sample averaged classification accuracies of different boosting algorithms distributed over (a) 3 synthetic spatial data sets; (b) 8 Covertype data sets (c) 4 LED data sets; (d) 4 Waveform data sets (- - - Standard Boosting, … Simple Majority, — Weighted Majority, ooo Competing among classifiers, xxx Boosting from a single site)

When performing the experiments on the Covertype data set (Figure 5b), the voting algorithms for distributed learning were consistently comparable in prediction accuracy to standard boosting on the centralized data. The method of competing classifiers was almost as accurate as standard boosting for large data sets, but slightly worse than standard boosting for smaller data sets (Figure 5-b3). It is also noticeable that for achieving the maximal prediction accuracy the larger number of boosting iterations was needed for smaller data sets than for larger ones.

In addition, the effect of dividing the sampling weights $w_{j,t}$ by the factor acc^p, (p = 0,1,2) was investigated for the three distributed boosting methods. In general, in the presence of sites that are significantly more difficult for learning than the others, a small increase in the sampling weights $w_{j,t}$ resulted in achieving the maximal prediction accuracy in a fewer number of boosting rounds. However, a larger acc^p factor (p = 2) could cause drawing insufficiently large samples from the sites that were easy to learn in later boosting iterations. As a consequence, the factor acc^2 (p = 2) could possibly result in method instability and in a drop of prediction accuracy. To alleviate this problem, when constructing a classifier we required that the size of the data sample drawn in the current boosting round had to be at least 15% of the original data set size on that site. Otherwise, the best classifier built so far on a particular site was used when making a classifier ensemble. In our experiments, increasing the weights $w_{j,t}$ usually resulted in deteriorating the classification accuracy and in instability of the proposed method for smaller data sets (Figure 6-a3), while preserving maximal prediction accuracy for experiments with large data sets (Figure 6-a1).

The performed experiments on LED, Waveform and Covertype data sets showed similar prediction accuracy for all explored factors for updating the sampling weights $w_{j,t}$ (Table 1). This was probably due to homogeneous distributions in these data sets, where there were no extremely difficult examples that need to be emphasized.

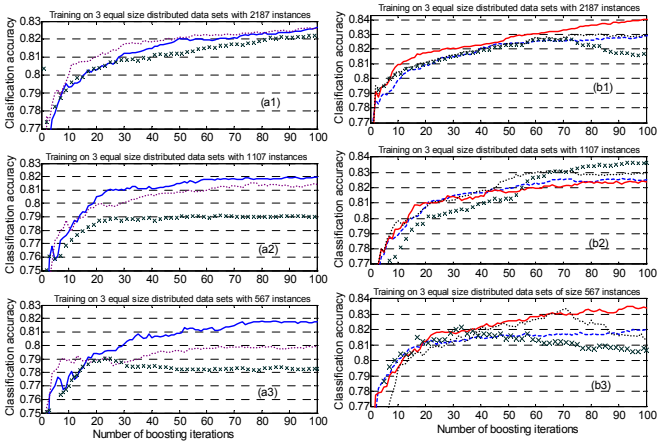

Figure 6. Out of sample averaged classification accuracies of the (a) simple majority boosting algorithm and (b) confidence-based weighted combining classifiers on 3 synthetic spatial data sets (- - - Standard Boosting, — p = 0; … p = 1; xxx p = 2)

Table 1. Final classification accuracies (%) for different distributed algorithms applied on four different data collections when dividing the weights $w_{j,t}$ by the factor acc^p

Method ↓ Data set →		Spatial	LED	Waveform	Covertype
Simple majority	p = 0	82.7	73.4	87.0	72.6
	p = 1	82.6	73.3	86.9	72.7
	p = 2	82.2	73.1	86.8	72.5
Confidence-based weighting	p = 0	84.3	73.4	87.2	73.1
	p = 1	82.9	73.6	87.1	73.2
	p = 2	82.1	73.4	87.1	73.0

Finally, we also performed experiments on 3 synthetic spatial data sets using the confidence-based method of combining classifiers with all three modifications for dividing the weights $w_{j,t}$ by the factor acc^p (p=0,1,2) (Figure 6b). The graphs in Figure 6b show that the confidence-based combining classifiers slightly outperformed standard boosting applied on centralized data as well as the other methods considered for distributed boosting. The improvement in prediction accuracy was more significant when learning from smaller data sets, but instability was also more evident for smaller data sets (Fig. 6-b3). The increase in prediction accuracy with decreasing the data sets was probably due to the fact that the data sets were homogeneous and more data points were needed in order to improve the generalizability of our models. When the number of data instances decreased, there were not enough examples to learn data distribution on a single site, but the variety of data instances from multiple sites still helped in achieving diversity of built classifiers.

Due to homogeneous distributions, the experiments performed on LED, Waveform and Covertype data sets again demonstrated the small observable difference in accuracy between the standard boosting and all variants of confidence-based distributed boosting algorithms when p = 0, 1 and 2 (Table 1).

5. CONCLUSIONS

A framework for distributed boosting is proposed. It is intended to efficiently learn stable non-linear classifiers over large and distributed homogeneous databases that cannot fit into the computer main memory. Experimental results on several data sets indicate that the proposed boosting techniques can effectively achieve the same or even slightly better level of prediction accuracy than standard boosting when applied to centralized data, while the cost of learning and memory requirements are considerably lower.

This paper raised several interesting issues that recently have gained a lot of attention. First, successful learning from very large and potentially distributed databases imposes major performance challenges for data mining, since learning a monolithic classifier can be prohibitively slow due to the requirement that all the data need to be held in the main memory. Second, many distributed data sets cannot be merged together due to a variety of practical constraints including data dispersed over many geographic locations, security services and competitive interests. Third, the prediction accuracy of employed data mining algorithms is of fundamental impact for their successful application. Finally, the computational time required for constructing a prediction model is becoming more important as the amount of available data is constantly growing. Our experiments performed on several data sets indicate that the proposed boosting techniques successfully overcome these concerns, thus offering a fairly general method for effective and efficient learning in distributed environment.

Although performed experiments have provided evidence that the proposed methods can be successful for distributed learning, future work is needed to fully characterize them especially in distributed environment with heterogeneous databases, where new algorithms for selectively combining classifiers from multiple sites with different distributions are worth considering. It would also be interesting to examine the influence of the larger number of distributed sites and their sizes to the achieved prediction accuracy, speedup and scale up and to establish a satisfactory trade off.

A possible drawback of the proposed methods is that a large number of classifiers and their ensembles are constructed from available data sets. In such situation, the methods of post-pruning the classifiers [9] may be necessary to increase system throughput, while still maintaining the achieved prediction accuracy.

6. ACKNOWLEDGMENTS

Partial support from NSF-CSE-IIS-9711532 to Zoran Obradovic and Keith Dunker and from INEEL LDRD Program under DOE Idaho Operations Office Contract DE-AC07-99ID13727. The authors are grateful to Dragoljub Pokrajac and Vasilis Megalooikonomou for their useful comments.

7. REFERENCES

[1] Blackard, J. Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types, Ph.D. dissertation, Colorado State University, (1998).

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

[3] Chan, P. and Stolfo, S. On the Accuracy of Meta-learning for Scalable Data Mining, Journal of Intelligent Integration of Information, (Kerschberg L. Ed.), (1998).

[4] Clearwater, S., Cheng, T., Hirsh, H., Buchanan, B. Incremental Batch Learning. In Proceedings of the 6th International Machine Learning Workshop, (1989), 366-370.

[5] Fan, W., Stolfo, S., Zhang, J. The Application of AdaBoost for Distributed, Scalable and On-line Learning, in Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (1999), 362-366.

[6] Freund, Y., and Schapire, R. E. Experiments with a New Boosting Algorithm, in Proceedings of the 13th International Conference on Machine Learning, (1996), 325-332.

[7] Guo, Y., Sutiwaraphun, J. Probing Knowledge in Distributed Data Mining, Proceedings of the 3rd Pacific-Asia Conference in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, Springer, (1999), 1574:443-452.

[8] Hagan, M., Menhaj, M.B. Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks (1994), 5, 989-993.

[9] Lazarevic, A., Obradovic, Z. The Effective Pruning of Neural Network Ensembles, in Proceedings of the IEEE International Joint Conference on Neural Networks, (2001), in press.

[10] Pokrajac D., Fiez T., Obradovic Z. A Spatial Data Simulator for Agriculture Knowledge Discovery Applications, in review.

[11] Riedmiller, M., Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, in Proceedings of the IEEE International Conference on Neural Networks, (1993), 586-591.

[12] Utgoff, P. An Improved Algorithm for Incremental Induction of Decision Trees, in Proceedings of the 11th International Conference on Machine Learning, (1994), 318-325.