Distributed and Parallel Databases, 11, 203–229, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

Boosting Algorithms for Parallel and Distributed Learning

ALEKSANDAR LAZAREVIC aleks@ist.temple.edu
ZORAN OBRADOVIC zoran@ist.temple.edu
Center for Information Science and Technology, Temple University, 303 Wachman Hall (038-24), 1805 N. Broad St., Philadelphia, PA 19122-6094, USA

Recommended by: Mohammed J. Zaki

Abstract. The growing amount of available information and its distributed and heterogeneous nature has a major impact on the field of data mining. In this paper, we propose a framework for parallel and distributed boosting algorithms intended for efficient integrating specialized classifiers learned over very large, distributed and possibly heterogeneous databases that cannot fit into main computer memory. Boosting is a popular technique for constructing highly accurate classifier ensembles, where the classifiers are trained serially, with the weights on the training instances adaptively set according to the performance of previous classifiers. Our parallel boosting algorithm is designed for tightly coupled shared memory systems with a small number of processors, with an objective of achieving the maximal prediction accuracy in fewer iterations than boosting on a single processor. After all processors learn classifiers in parallel at each boosting round, they are combined according to the confidence of their prediction. Our distributed boosting algorithm is proposed primarily for learning from several disjoint data sites when the data cannot be merged together, although it can also be used for parallel learning where a massive data set is partitioned into several disjoint subsets for a more efficient analysis. At each boosting round, the proposed method combines classifiers from all sites and creates a classifier ensemble on each site. The final classifier is constructed as an ensemble of all classifier ensembles built on disjoint data sets. The new proposed methods applied to several data sets have shown that parallel boosting can achieve the same or even better prediction accuracy considerably faster than the standard sequential boosting. Results from the experiments also indicate that distributed boosting has comparable or slightly improved classification accuracy over standard boosting, while requiring much less memory and computational time since it uses smaller data sets.

Keywords: parallel boosting, distributed boosting, heterogeneous databases, boosting specialized experts

1. Introduction

The recent, explosive growth of information available to business and scientific fields has resulted in an unprecedented opportunity to develop automated data mining techniques for extracting useful knowledge from massive data sets. Large-scale data analysis problems very often also involve the investigation of relationships among attributes in heterogeneous data sets where rules identified among the observed attributes in certain regions do not apply elsewhere. This problem may be further complicated by the fact that in many cases, the heterogeneous databases are located at multiple distributed sites. Therefore, the issues of modern data mining include not just the size of the data to be mined but also its location and homogeneity.

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. Another need for learning from multiple sources could be found when datasets have grown too large to fit into the main computer memory. A parallel approach to building a model in such a situation is aimed at solving the practical problem of how to learn from large data sets. For this purpose, various parallelized machine learning algorithms were proposed, e.g. parallel decision tree [26, 28, 29], parallel association rules [31] and parallel rule induction [23]. On the other hand, in order to solve the problem of learning from very large and/or distributed databases, some researchers have proposed incremental learning techniques. Usually these techniques involve direct modifications of standard learning algorithms, such as decision trees [30] and rule learner [5].

Since distributed learning usually involves several data sets from multiple sites, 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 [4, 22] were proposed in order to coalesce the predictions of classifiers trained from different partitions of the training set. The advantage of this 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.

In this paper, we propose a novel technique of combining classifiers from multiple sites using a boosting technique [9]. 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. Boosting has received extensive theoretical and empirical study [10, 19], but most of the published work focuses on improving the accuracy of a weak classifier over the same single, centralized data set that is small enough to fit into the main memory. So far, there has not been much research on using the boosting technique for distributed and parallel learning. The only exception was boosting for scalable and distributed learning [7], where each classifier was trained using only 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.

Our objective was to develop a boosting technique applicable to both parallel and distributed environments. The first boosting modification represents a "parallelized" version of the boosting algorithm for a tightly coupled shared memory system with a small number of processors (e.g. a dual processor system). This method is applicable when all the data can fit into the main memory, with the major goal of speeding up the boosting process. At each boosting round, all classifiers are trained on different samples drawn from the same training data set and then assigned to one of the data instances according to the confidence of their prediction. Empirical studies of several data sets have demonstrated that this method can achieve the same or slightly better classification accuracy considerably faster than the standard boosting algorithm.

The second boosting adaptation explored in our study is more suitable for distributed learning, since it assumes that the disjoint data sets from multiple sites cannot be merged together. However, it can also be applied to parallel learning, where the training data set is split into several sets that reside on different processors within a parallel computer. The data sets can be either homogeneous or heterogeneous, or can even contain different data distributions. In the proposed method, at each boosting round the classifiers are first learned on disjoint datasets and then exchanged amongst the sites. The exchanged classifiers are then combined and their weighted voting ensemble is constructed on each disjoint data set. The final ensemble represents an ensemble of ensembles built on all local distributed sites. The performance of ensembles is used to update the probabilities of drawing 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. Methodology

The modifications of the boosting algorithm that we propose in this paper are variants of the AdaBoost.M2 procedure [9], shown in figure 1. The algorithm supports multi-class

```
• Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$
• Let $B = \{(i, y): i = 1, \ldots, m, y \neq y_i\}$
• Initialize the distribution $D_1$ over the examples, such that $D_1(i) = 1/m$.
• For $t = 1, 2, 3, 4, \ldots T$
  1. Train a weak learner using distribution $D_t$
  2. Compute weak hypothesis $h_t$: $X \times Y \rightarrow [0, 1]$
  3. Compute the pseudo-loss of hypothesis $h_t$:
     $\varepsilon_t = \frac{1}{2} \cdot \sum_{(i,y)\in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$
  4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1 - h_t(x_i, y) + h_t(x_i, y_i))$
  5. Update $D_t$ : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{w_t}$
     where $Z_t$ is a normalization constant chosen such that $D_{t+1}$ is a distribution.
• Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$
```

Figure 1. The AdaBoost.M2 algorithm.

problems and proceeds in a series of T rounds. In every round, 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 single 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 often 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 many weaker hypotheses. In addition, a natural way of learning in a distributed environment is by combining classification predictors. Our objective, therefore, is to exploit all of these advantages in order to apply boosting to parallel and distributed learning.

As classifiers in all boosting experiments, we trained two-layer feedforward neural network (NN) models since such universal approximators were often reported to outperform the alternatives for classification of real life non-linear phenomena [12]. The NN classification models had the number of hidden neurons equal to the number of input attributes. To use NNs with AdaBoost.M2 algorithm, our implementation had the number of output nodes equal to the number of classes, where the predicted class is from the output with the largest response. In such an implementation, the output nodes compose a set of "plausible" labels, thus directly satisfying requirement of AdaBoost.M2 algorithm that the values of the output nodes indicate a "degree of plausibility" and all these plausibilities do not necessarily add up to 1. To optimize NN parameters we used resilient propagation [24] and Levenberg-Marquardt [11] learning algorithms.

Although there are known ways of combining NNs trained on different subsets in order to produce a single learner (e.g. Breiman's born again trees [3]) very often they do not provide as good accuracy as an ensemble of classifiers created using the boosting algorithm. Since our major objective was to improve the generalization capabilities of proposed methods, constructing a simple and more comprehensive model was not considered in this study.

2.1. Boosting for parallel learning

The idea of proposed parallel boosting is to speed up the learning process of the standard boosting. Given a tightly coupled shared memory system with a few processors, our goal is to train classifiers on each of the available processors, and achieve the maximal prediction accuracy faster than when learning on a single processor. We assume there are k processors in the system, and each of them has access to entire training data set. The proposed algorithm is shown in figure 2.

In the proposed method, the classifiers are constructed on each of k available processors at each boosting round. Each classifier is trained on a different sample $Q_{j,t}$ drawn from the same training set S according to the same distribution D_t. Instead of a single classifier built at each boosting iteration, in parallel boosting there are k classifiers that compete for the data examples according to the confidence of their predictions. The classifier with the highest prediction confidence for some data instance is responsible for making the prediction on

```
• Given: Set S $\{(x_1, y_1), \dots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \dots, C\}$
• Let $B = \{(i, y): i = 1,\dots,m, y \neq y_i\}$
• Initialize the distribution $D_1$ over the examples, such that $D_1(i) = 1/m$.
• For t = 1, 2, 3, 4, ... $T$
  1. For j = 1 ... $k$ (For each of $k$ processors)
     1.1. Draw a sample $Q_{j,t}$ for processor $j$ according to distribution $D_t$
     1.2. Train a weak learner on the sample $Q_{j,t}$ using processor $j$
     1.3. Compute weak hypothesis $h_{j,t}$: $X \times Y \rightarrow [0, 1]$
  2. Create hypothesis $h_t$ by selecting the best hypothesis $h_{j,t}$ for each data
     instance $x_i$: $h_t(i) = \max_{j=1,k} h_{j,t}(i)$, i = 1, ...,m.
  3. Compute the pseudo-loss of hypothesis $h_t$:
     $\varepsilon_t = \frac{1}{2} \cdot \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$
  4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2) \cdot (1 - h_t(x_i, y) + h_t(x_i, y_i))$
  5. Update $D_t$ : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{w_t}$
  where $Z_t$ is a normalization constant chosen such that $D_{t+1}$ is a distribution.
• Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$
```

Figure 2. Parallel boosting algorithm.

that data example. Therefore, there is a different classifier responsible for each data point inside the data set, while the hypothesis h_t for boosting iteration t, is a mixture of weak hypotheses $h_{j,t}$, $j = 1, \dots, k$. The distribution D_t is updated according to the performance of the mixed hypothesis h_t on the training set S, and it is used by each processor to draw samples in subsequent boosting rounds. The composite hypothesis h_t is also used when making a final hypothesis h_{fn}. It is very important to note that within the system only the classifiers are moved, not the data examples themselves.

2.2. *Boosting for distributed learning in homogeneous databases*

2.2.1. The general framework of distributed boosting. The objective of a distributed learning algorithm is to efficiently construct a prediction model using data at multiple sites such that prediction accuracy is similar to learning 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 3. All distributed sites perform the learning procedure at the same time.

Assume there are k distributed sites, where site j contains data set S_j with m_j examples, $j = 1, \dots, k$. 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 the local weights $w_{j,t}$ that directly reflect the prediction accuracy on that site. 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

```
• On site $j$, (j = 1...$k$) we are given set $S_j$ $\{(x_{j,1}, y_{j,1}), \ldots, (x_{j,m_j}, y_{j,m_j})\}$ $x_{j,i} \in X_j$, with labels
  $y_{j,i} \in Y_j = \{1, \ldots, C\}$, j = 1...$k$. Let $B_j = \{(i, y_j)$: i = 1,...,$m_j$, $y_j \neq y_{j,i}\}$.
• 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 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 ... $k$ (For all distributed sites)
  • For t = 1, 2, 3, 4, ... $T$
    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 $Q_{j,t}$ if $Q_{j,t}$ is sufficiently large; otherwise as a
       weak learner $L_{j,t}$ use the most accurate learner $L_{j,p}$, $p < t$.
    3. Broadcast trained learner $L_{j,t}$ to all distributed sites.
    4. Create an ensemble $E_{j,t}$ by combining the learners $L_{j,t}$, $j = 1 \ldots k$.
    5. Using the ensemble $E_{j,t}$ compute weak hyposthesis $h_{j,t}$: $X \times Y \to [0, 1]$.
    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 distribute 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 $D_{j,t}$
        after normalization make the local distribution $\Delta_{j,t}$.
• Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} \sum_{j=1}^{k} (\log \frac{1}{\beta_{j,t}}) \cdot h_{j,t}(x, y_j)$
```

Figure 3. The distributed boosting framework.

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 (step 5, 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, figure 3). 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 the number of data examples need to be sampled from the site q. Therefore, each site j creates a weight vector $U_{j,t}$ (step 10, figure 3), 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 the all 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 to do this is to set all values in 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 $D_{j,t}$. It is evident that the samples $Q_{j,t}$ from distributed sites do not necessarily have the same size through iterations, but the total number of examples drawn from all distributed sites is always the same. The motive for using "unbalanced" sampling from distributed sites is to simulate drawing the same instances as in the standard boosting. At each boosting round t, the classifiers $L_{j,t}$ are constructed on each of the samples $Q_{j,t}$ and then exchanged among the distributed sites. However, if the sample $Q_{j,t}$ is not sufficiently large, the most accurate classifier $L_{j,p}$, $(p < t)$ constructed so far is used. The minimum size of the sample that may be used for training of classifiers represents the size of random data sample for which only a small, predefined accuracy loss is achieved when comparing to accuracy obtained by learning from entire training set. 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 3) 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}.

In order to simulate the boosting on centralized data, our intention was to draw more data instances from the sites that are more difficult for learning. The weights $w_{j,t}$ computed in step 8, directly reflect the prediction capability for each data point, thus satisfying our goal to sample more examples from the sites that are more difficult to learn. In order to further emphasize sampling from the sites that are difficult for learning, we consider dividing the weights $w_{j,t}$ by the factor acc_j^p ($p = 0$, 1 or 2), 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 (larger value p results in larger difference between the weights).

2.2.2. The variants of distributed boosting.

We explore several variants of the proposed distributed boosting algorithm (figure 3). The variants differ in creating an ensemble $E_{j,t}$ obtained by combining the classifiers $L_{j,t}$ (step 4).

The simplest method for combining classifiers into an ensemble $E_{j,t}$ is based on *Simple Majority Voting of Classifiers from All Sites*. If the classifiers $L_{l,t}, l = 1, \ldots, k$, from all sites

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

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

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

Figure 4. The confidence-based technique for weighted combining classifiers from distributed sites.

produce hypotheses $h_{l,j,t}$ on site j, then the hypothesis $h_{j,t}$ (step 5, figure 3) is computed as:

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

More sophisticated techniques for distributed boosting consider weighted combinations of classifiers. In *Weighted Majority Voting of Classifiers from All Sites*, 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:

$$h_{j,t} = \frac{\sum_{l=1}^{k} u_{l,j,t} \cdot h_{l,j,t}}{\sum_{l=1}^{k} u_{l,j,t}} \tag{2}$$

We also consider *Confidence-based Weighting Classifiers from All Sites*, where the classifiers from all sites are combined using the procedure similar to boosting technique. If the classifiers $L_{l,t}$ at iteration t produce hypotheses $h_{l,j,t}$ on data set S_j from site j that maintains the distribution $\Delta_{j,t}$, then this technique of combining classifiers is defined at figure 4.

2.3. Boosting for distributed learning in heterogeneous databases

We consider two scenarios when learning from heterogeneous databases among the distributed sites: (1) all heterogeneous databases with a similar mixture of distributions; (2) databases with different but homogeneous distributions. In both scenarios, all data sites have the same set of attributes.

2.3.1. Learning from heterogeneous databases with a similar mixture of distributions. Our previous research shows that in heterogeneous databases where several more homogeneous regions exist, standard boosting does not enhance the prediction capabilities as significantly as for homogeneous databases [14]. In such cases it is more useful to have several local experts each with expertise in a small homogeneous region of the data set [15]. A possible approach to this problem is to cluster the data first and then to assign a single specialized classifier to each discovered cluster. Therefore, we combine this boosting

```
• On site $j$, (j = 1…$k$) we are given set $S_j$ $\{(x_{j,1}, y_{j,1}), \ldots, (x_{j,m_j}, y_{j,m_j})\}$ $x_{j,i} \in X_j$, with labels
  $y_{j,i} \in Y_j = \{1, \ldots, C\}$. 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 $\Delta_{j,i}(i) = 1/m_j$, and
  compute the sums $\sum_{i \in S_j} \Delta_{j,1}(i)$ of all the elements in distributions $\Delta_{j,1}$.
• Each site 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 … $k$ (For all distributed sites)
  • While ($t < T$) or (global accuracy on set $S_j$ starts to decrease)
    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. Find relevant attributes for the sample $Q_{j,t}$ using an unsupervised wrapper technique
       around a clustering algorithm.
    3. Apply clustering on the sample $Q_{j,t}$ using the most relevant attributes identified in step 2
       and obtain $c_j$ clusters $Q_{j,t,l}$, $l = 1, \ldots c_j$.
    4. Train weak learners $L_{j,t,l}$ on the sets $Q_{j,t,l}$, $l = 1, \ldots c_j$ respectively.
    5. Broadcast classifiers $L_{j,t,l}$ to all distributed sites.
    6. Using a convex hull mapping method (Figure 7) obtain clusters $S_{j,t,l}$ and their distributions
       $D_{j,t,l}$ on the entire set $S_j$ that corresponds to clusters $Q_{j,t,l}$ identified on the sample $Q_{j,t}$.
    7. For each cluster $S_{j,t,l}$ create an ensembles $E_{j,t,l}$ by combining learners $L_{j,t,l}$, $l = 1, \ldots, c_j$.
    8. Compute a weak hypothesis $h_{j,t,l}$, $l = 1, \ldots, c_j$ by applying an ensemble $E_{j,t,l}$.
    9. Merge the hypotheses $h_{j,t,l}$ into a unique hypothesis $H_{j,t}$ (Figure 6).
    10. Compute the pseudo-loss of hypothesis $H_{j,t}$:
        $\varepsilon_{j,t} = \frac{1}{2} \cdot \sum_{(i,y) \in S_j} \Delta_{j,t}(i,y)(1 - H_{j,t}(x_{j,i}, y_{j,i}) + H_{j,t}(x_{j,i}, y_j))$
    11. Set $\beta_{j,t} = \varepsilon_{j,t} / (1 - \varepsilon_{j,t})$
    12. Compute $w_{j,t}(i, y_j) = (1/2) \cdot (1 - H_{j,t}(x_{j,i}, y_j) + H_{j,t}(x_{j,i}, y_{i,j}) / acc_j^p$, $p \in \{0, 1, 2\}$
    13. Compute $V_{j,t} = \sum_{(i,y_j)} w_{j,t}(i, y_j)$ and broadcast to all sites.
    14. 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 $V_{q,t}/m_q$.
    15. 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 in $D_{j,t}$
        after normalization make the local distribution $\Delta_{j,t}$.
• Output the final hypothesis: $H_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} \sum_{j=1}^{k} (\log \frac{1}{\beta_{j,t}}) \cdot H_{j,t}(x, y_j)$
```

Figure 5. The distributed boosting specialized experts for sites with heterogeneous distributions.

specialized experts approach with already proposed distributed boosting in order to further improve it. The general idea for boosting specialized experts in a distributed environment is shown in figure 5.

Similar to learning from homogeneous distributed databases (Section 2.2), all k sites again maintain their own versions $D_{j,t}$ of the distribution D_t, and the final hypothesis H_{fn} represents the combination of hypotheses $H_{j,t}$ from different sites and different boosting

- Compute the convex hulls $CH_{j,t,l}$, $l = 1,\ldots,c_j$, for each of clusters $Q_{j,t,l}$ discovered when clustering the sample $Q_{j,t}$.
- Determine clusters $Q_{j,t,l}$ on the entire data set S_j according to convex hull mapping (Figure 7). All points inside the convex hull $CH_{j,t,l}$ belong to the *l-th* cluster $S_{j,t,l}$.
- Compute the pseudo-loss of hypothesis $h_{j,t,l}$

$$\varepsilon_{j,t,l} = \frac{1}{2} \sum_{(x_{i,j},y_{i,j})\in S_{j,t,l}} D_{j,t,l}(i^j, y^j)(1 - h_{j,t,l}(x_{i,j}, y_{i,j}) + h_{j,t,l}(x_{i,j}, y^j))$$

- Set $\beta_{j,t,l} = \varepsilon_{j,t,l} / (1 - \varepsilon_{j,t,l})$
- Merge all $h_{j,t,l}$, $l = 1,\ldots c_j$ into a unique weak hypothesis $H_{j,t}$ and all $\beta_{j,t,l}$, $l = 1,\ldots c_j$ into a unique $\beta_{j,t}$ according to their convex hull belonging (example fitting in the *l-th* convex hull $CH_{j,t,l}$ has the hypothesis $h_{j,t,l}$ and the value $\beta_{j,t,l}$).

Figure 6. The algorithm for merging specialized hypotheses $h_{j,t,l}$ into a composite hypothesis $H_{j,t}$.

iterations. However, in this scenario $H_{j,t}$ is not the composite hypothesis that corresponds to a classifier ensemble created on the site j at iteration t, but it represents a composite hypothesis obtained by combining classifier ensembles $E_{j,t,l}$ constructed on c_j clusters identified on site j at iteration t. Due to the similar mixture of distributions, assume that the number of discovered clusters is the same on all sites. Ensembles $E_{j,t,l}$ are constructed by combining classifiers $L_{j,t,l}$ that are learned on clusters $Q_{j,t,l}$, $l = 1, \ldots, c_j$, obtained by applying clustering algorithm on the sample $Q_{j,t}$. All classifiers $L_{j,t,l}$ are then exchanged among all sites, but only learners $L_{j,t,q}$, $q = 1, \ldots, c_j$, that correspond to the q-th cluster $Q_{j,t,q}$ are combined to create an ensemble $E_{j,t,q}$ on each site j and to compute a corresponding hypothesis $h_{j,t,l}$. Merging the hypotheses $h_{j,t,l}$ that correspond to ensembles $E_{j,t,l}$ into a composite hypothesis $H_{j,t}$ is performed using the algorithm described in figure 6. Therefore, the final classifier corresponding to the final hypothesis H_{fn} is computed by combining the classifiers from discovered clusters at different sites and different iterations.

In merging hypotheses $h_{j,t,l}$, data points from different clusters $S_{j,t,l}$ have different pseudo-loss values $\varepsilon_{j,t,l}$ and different parameter values $\beta_{j,t,l}$. For each cluster $S_{j,t,l}$, $l = 1, \ldots, c_j$, from iteration t, defined by the convex hull $CH_{j,t,l}$, there is a pseudo-loss $\varepsilon_{j,t,l}$ and the corresponding parameter $\beta_{j,t,l}$ (figure 6). Both the pseudo-loss value $\varepsilon_{j,t,l}$ and parameter $\beta_{j,t,l}$ are computed independently for each cluster $S_{j,t,l}$ where a particular classifier $L_{j,t,l}$ is responsible. Before updating distribution $D_{j,t}$, a unique vector $\beta_{j,t}$ is created such that the i-th position in the vector $\beta_{j,t}$ is equal to $\beta_{j,t,l}$ if the i-th pattern from the entire sets S_j belongs to the cluster $S_{j,t,l}$ identified at iteration t. Similar, the hypotheses $h_{j,t,l}$ are merged into a single hypothesis $H_{j,t}$. Since we merge $\beta_{j,t,l}$ into $\beta_{j,t}$ and $h_{j,t,l}$ into $h_{j,t}$, updating the distribution $D_{j,t}$ can be performed in the same way as in the distributed boosting algorithm (Section 2.2).

Our distributed boosting algorithm for heterogeneous databases involves clustering at step 3 (figure 5). Therefore, there is a need to find a small subset of attributes that uncover "natural" groupings (clusters) from the data according to some criterion. For this purpose we adopt the wrapper framework in unsupervised learning [6], where we apply the clustering algorithm to attribute subsets in the search space and then evaluate the subset by a criterion function that utilizes the clustering result. If there are d attributes, an exhaustive search of 2^d possible attribute subsets to find one that maximizes our selection criterion is computationally intractable. Therefore, in our experiments, fast sequential forward selection search is applied. Like in [6] we also accept the scatter separability trace for attribute selection criterion.

This procedure, performed at step 2 of every boosting iteration, results in r relevant attributes for clustering. Thus, for each round of boosting algorithm at each site j, there are relevant attribute subsets that are responsible for distinguishing among homogeneous distributions in the sample $Q_{j,t}$. In order to find those homogeneous regions, clustering at each boosting iteration is performed. Two clustering algorithms are employed: standard k-means algorithm and density based clustering algorithm, called DBSCAN [25], designed to discover clusters of arbitrary shape efficiently.

As a result of clustering, on each site j several distributions $D_{j,t,l}$ $(l = 1, \dots, c_j)$ are obtained, where c_j is the number of discovered clusters at site j. For each of c_j clusters discovered in the data sample $Q_{j,t}$, a weak learner $L_{j,t,l}$ is trained using the corresponding data distribution $S_{j,t,l}$, and a weak hypothesis $h_{j,t,l}$ is computed. Furthermore, for every cluster $Q_{j,t,l}$ identified at the sample $Q_{j,t}$, its convex hull $CH_{j,t,l}$ is identified in the attribute space used for clustering, and these convex hulls are applied to the entire training set in order to find the corresponding clusters $S_{j,t,l}$ (figure 7) [17]. All data points inside the convex hull $CH_{j,t,l}$ belong to the *l-th* cluster $S_{j,t,l}$ discovered at iteration t on site j. Data points outside the identified convex hulls are attached to the cluster containing the closest data pattern. Therefore, instead of a single global classifier constructed in every boosting iteration, there are c_j classifiers $L_{j,t,l}$ and each of them is applied to the corresponding cluster $S_{j,t,l}$.

When performing clustering during boosting iterations, it is possible that some of the discovered clusters have insufficient size for training a specialized classifier. Hence, instead of training a specialized classifier on such a cluster with an insufficient amount of data, classifiers from previous iterations that were constructed on the corresponding clusters detected through the convex hull matching are consulted (figure 7) and one with the maximal *local* prediction accuracy is employed.

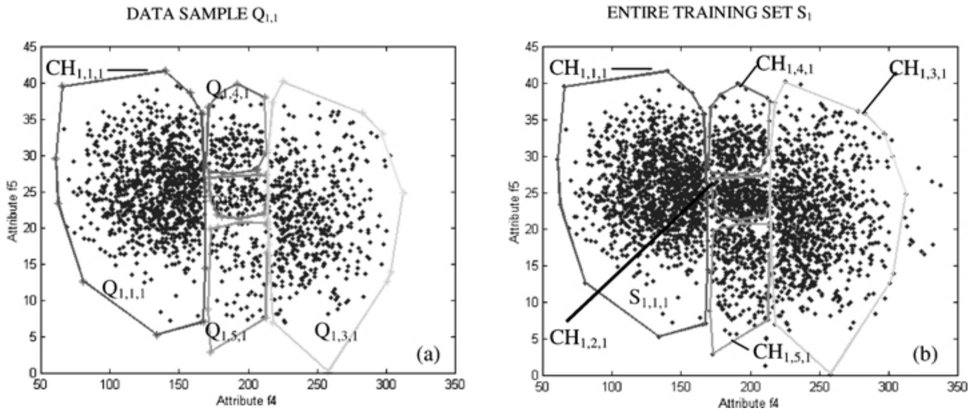

Figure 7. Mapping convex hulls ($CH_{1,1,l}$) of clusters $Q_{1,1,l}$ discovered in data sample $Q_{1,1}$ to the entire training set S_1 in order to find corresponding clusters $S_{1,1,l}$. For example, all data points inside the contours of convex hull $CH_{1,1,1}$ (corresponding to cluster $Q_{1,1,1}$ discovered on $Q_{1,1}$) belong to cluster $S_{1,1,1}$ identified on S_1.

2.3.2. Learning from different homogeneous distributions. An alternative boosting algorithm for learning in a heterogeneous distributed environment is proposed, where k distributed sites contain databases with the same attributes but with different homogeneous distributions. An unseen test data set may belong to one of these distributions or may be a mixture of distributions from the multiple sites. Here, the method for distributed boosting from Section 2.2 will not provide satisfactory prediction accuracy, since the classifiers learned from one data distribution will have poor performance on a data set with a different distribution. Therefore, when making prediction there is a need to identify appropriate classifiers and to determine a measure of this appropriateness.

When the test data set contains only one distribution, only classifiers that are constructed on data sets that stem from distributions similar to the distribution from the test set are combined. For determining similar distributions, the difference between them is computed using Mahalanobis distance [8], since our previous research indicated that it could be an effective technique for distinguishing between two mixtures of distributions [21]. Given data sets S_1 and S_2 from distributed sites 1 and 2, the Mahalanobis distance between them is computed as:

$$d_{mah} = \sqrt{\left(\mu_{S_1} - \mu_{S_2}\right) \cdot \Sigma^{-1} \cdot \left(\mu_{S_1} - \mu_{S_2}\right)^T}, \tag{3}$$

where $\mu_{S_1}$and $\mu_{S_2}$are mean vectors of the data sets S_1 and S_2 respectively, and Σ is the sample covariance matrix [8]:

$$\Sigma = \frac{(m_1 - 1) \cdot \Sigma_1 + (m_2 - 1) \cdot \Sigma_2}{(m_1 + m_2 - 2)}, \tag{4}$$

with Σ_1 and Σ_2 denoting covariance matrices of S_1 and S_2. The Mahalanobis distance is computed without violating data privacy, since only the number of points (m_j), mean vectors (μ_{S_j}) and covariance matrices (Σ_j) are exchanged among the sites. Therefore, the distributed boosting algorithm (figure 3) is applied only to those sites that have the most similar distributions to the distribution from the test data set.

However, when the test data set contains a mixture of distributions from multiple sites, for each test point it is necessary to determine the originating distribution. For this purpose, given k data sets S_j, $j = 1, \ldots, k$, the Mahalanobis distance is computed between a new instance r and the distributions corresponding to each of the sets S_j:

$$d_{mah} = \sqrt{\left(r - \mu_{S_j}\right) \cdot \Sigma_{S_j}^{-1} \cdot \left(r - \mu_{S_j}\right)^T}, \tag{5}$$

where μ_{S_j} and $\Sigma_{S_j}^{-1}$ represent the mean vector and the covariance matrix of the data set S_j. The test data instances are classified into groups, such that all points inside one group are closest to one of the distributions from the k distributed sites. An ensemble of classifiers on each of distributed sites is constructed independently using the standard boosting approach. The classifier ensemble E_j is applied to the test subset whose instances are closest to the distribution of the data set S_j. In addition, when two or more distributed sites have the distributions sufficiently similar to one of the groups from the test data set, the distributed boosting algorithm from Section 2.2 is used to learn from sites with similar distributions.

3. Experimental results

Our experiments were performed on several data sets. When experimenting with parallel and distributed boosting in homogeneous environments, five data collections were used. The first one contained two synthetic spatial data sets with 6,561 instances generated using our spatial data simulator [20] such that the distributions of generated data were approximately Gaussian but samples exhibit spatial correlation similar to the distributions 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 is likely to result in overly optimistic estimates of prediction error (due to spatial correlation in data [18]), the training data set was spatially partitioned into three disjoint data sets used for distributed learning, each with 2,187 examples (figure 8). The obtained spatial data sets stemmed from similar homogeneous distributions and had five continuous attributes and three equal size classes. The other four data collections were Covertype, Pen-based digits, Waveform and LED data sets from the UCI repository [2]. The Covertype data set, currently one of the largest databases in the UCI Database Repository, contains 581,012 examples with 54 attributes and 7 target classes representing the forest cover type for 30 $\times$ 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 a transformed data set with 21 attributes. The 149,982 data instances were used for training in parallel boosting, while the same instances but separated into 8 disjoint data sets were used for distributed learning. The remaining 431,032 data examples were used for out of sample testing. For the Pen-based digit data set, containing 16 attributes and 10 classes, original training data set with 7,494 instances was randomly split into 6 disjoint subsets used for learning, each with 1,249 examples, while the data set with 3,498 instances was used for out of sample testing. For the Waveform set, 50,000 instances with 21 continuous attributes and three equally sized classes were generated. The generated data were split into 5 sets of 10,000 examples each, where 4 of them were merged and used as training set for parallel boosting, while the same 4 but separated were used for distributed learning. The fifth data

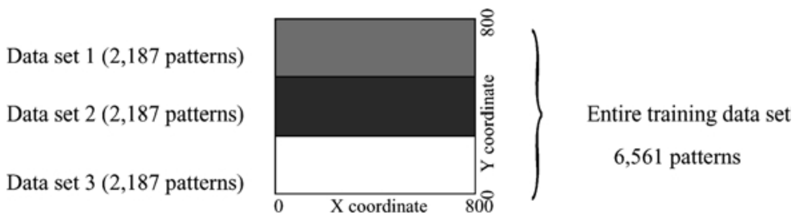
Data set 1 (2,187 patterns)
Data set 2 (2,187 patterns)
Data set 3 (2,187 patterns)
0
X coordinate
800
0
Y coordinate
800
Entire training data set
6,561 patterns

Figure 8. Partitioning the spatial training data set into three disjoint subsets.

set was used as a test set. The LED data set was generated with 10,000 examples and 10 classes, where 4 sets with 1,500 examples were used for training in parallel and distributed environment, and the set with 4,000 examples was used for testing.

For all proposed algorithms performed on all five data sets the reported classification accuracies were obtained by averaging over 10 trials of the boosting algorithm. When applying boosting techniques on neural network classifiers, the best prediction accuracies were achieved using the Levenberq-Marquardt learning algorithm.

3.1. Results for parallel boosting

When experimenting with the parallel boosting algorithm, neural network classifiers were constructed on two, three and four processors during boosting iterations, since our experimental results indicated that no significant differences in prediction accuracy were found when more neural networks were involved. In order to examine how the performance of parallel boosting depends on the size of the data used for learning, the size of the training data set was varied. The results for synthetic spatial data set and for Covertype data set are presented respectively at figure 9(a) and (b), while the summary of the results for all five data sets are reported in Table 1.

It is evident from the charts obtained for the synthetic spatial data sets (figure 9), that the parallel boosting method achieved slightly better prediction accuracy than the standard boosting method in less number of boosting rounds. The reduction in the number of

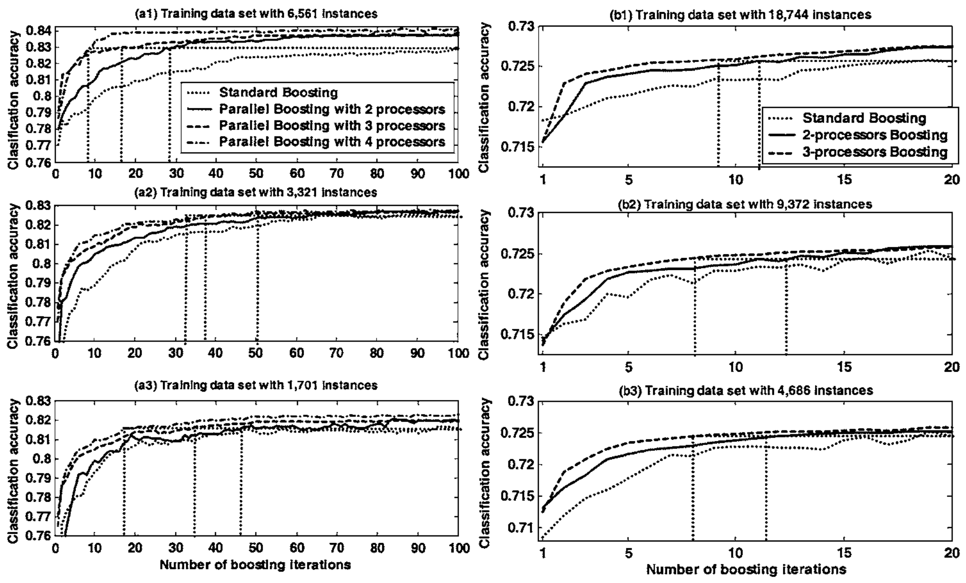

Figure 9. Out of sample averaged classification accuracies for standard boosting applied on a single processor and for parallel boosting applied on 2, 3 and 4 processors. Both algorithms are applied on (a) synthetic spatial data set (b) Covertype data set.

Table 1. A comparative analysis of parallel boosting speedup for different number of processors.

Number of processors	Number of parallel boosting iterations needed for achieving the same maximal accuracy as standard boosting trained on a single processor				
	Synthetic spatial	Covertype	Pen-based digits	LED	Waveform
1	100	20	12	10	8
2	28	11	6	7	6
3	14	9	5	6	5
4	9	8	4	5	5

boosting rounds required for achieving the maximal prediction accuracy was especially apparent when learning on larger data sets (figure 9(a1) and (b1)). However, when the size of entire training data set decreased, the parallel boosting became less superior to the standard boosting method (figure 9 (a2), (a3), (b2) and (b3)). This phenomenon was probably due to the overfitting of the neural networks constructed at boosting iterations, since the composite classifier computed through competing neural network classifiers was probably overfitted more than a single neural network classifier. Although overfitting may be sometimes useful when combining classifiers [27] (it increases the variance of the combined models and therefore their diversity too), when small training data sets are available, diversity of the classifiers could not be properly emphasized because the classifiers were constructed on the samples drawn from the same training set but with insufficient number of points for achieving reasonable generalizability. Therefore, when the time for constructing a classifier ensemble is an issue, the speed of achieving the maximal prediction accuracy may be especially valuable when tightly coupled computer systems with a few processors are available.

Since in the proposed parallel boosting there was only one accessed data set, the scaleup properties was not considered, but instead, we determined the speedup, i.e. the decrease in the number of iterations the parallel boosting needed for achieving the same accuracy comparing to standard boosting applied on a single processor. However, the parallel boosting does not provide speedup directly but indirectly, since each processor samples from the same "global" data set and several classifiers are computed per iteration instead of one. Results shown at Table 1 illustrate that the parallel boosting has very good speedup for synthetic spatial data, good for covertype and pen-based digit data sets, but fairly poor for LED and Waveform data sets probably due to their homogeneous distributions.

3.2. Results for distributed boosting in homogeneous environments

3.2.1. Time complexity analysis. We performed experiments on all five reported data sets and compared the computational time needed for training neural network (NN) classifiers using the standard and distributed boosting approach. 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 10 shows

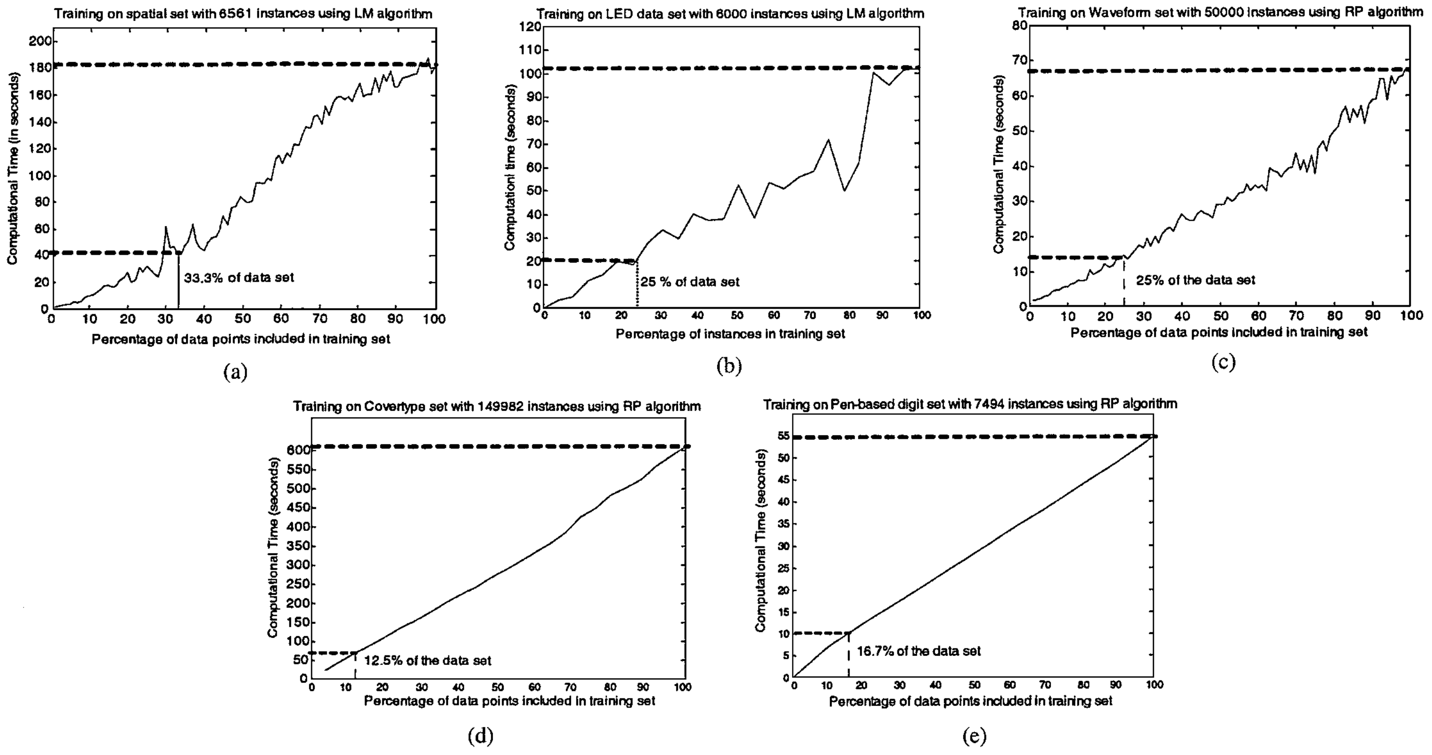

Figure 10. The time needed for learning NN classifiers for different sizes of five different data sets. (a) Spatial synthetic data set; (b) LED data set; (c) Waveform data set; (d) Covertype data set; (e) Pen-based digit data set.

how the time required for constructing NNs depends of the number of examples in training set for all three reported data sets when measured on a *Pentium III* processor with 768 MB of main memory. Analyzing the figure 10(a), 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. Similarly, when constructing NNs on LED and Waveform data sets, four times smaller data set caused more than four times faster learning (figure 10(b) and (c)). 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 10(d)). Finally, training a NN on a six times reduced Pen-based digit data set (figure 10(e)) resulted in 5.5 times faster training.

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 11). 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 also close to linear.

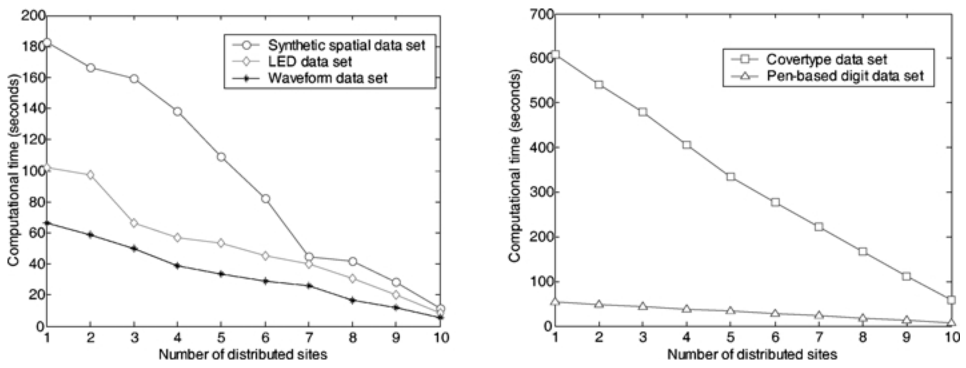

Figure 11. The speedup of the distributed boosting algorithm for different data sets.

3.2.2. Prediction accuracy comparison. To explore whether our distributed boosting algorithm can reach similar prediction accuracy as the standard boosting algorithm on a centralized data set, experiments were first performed using simple majority and weighted majority algorithms for learning from homogeneous databases (figure 12). In addition to comparison to standard boosting on centralized data, we also compared distributed boosting algorithms to simpler algorithms for distributed learning. The first one was "distributed bagging" where we used voting over classifier ensembles constructed independently on distributed sites using bagging procedure, while the second algorithm employed voting over classifier ensembles built separately on distributed sites using boosting method. Since the ensembles are constructed independently on each of distributed sites, the only communication includes exchanging the classifier ensembles built on each site at the end of procedure. However, voting over ensembles built using boosting method was consistently

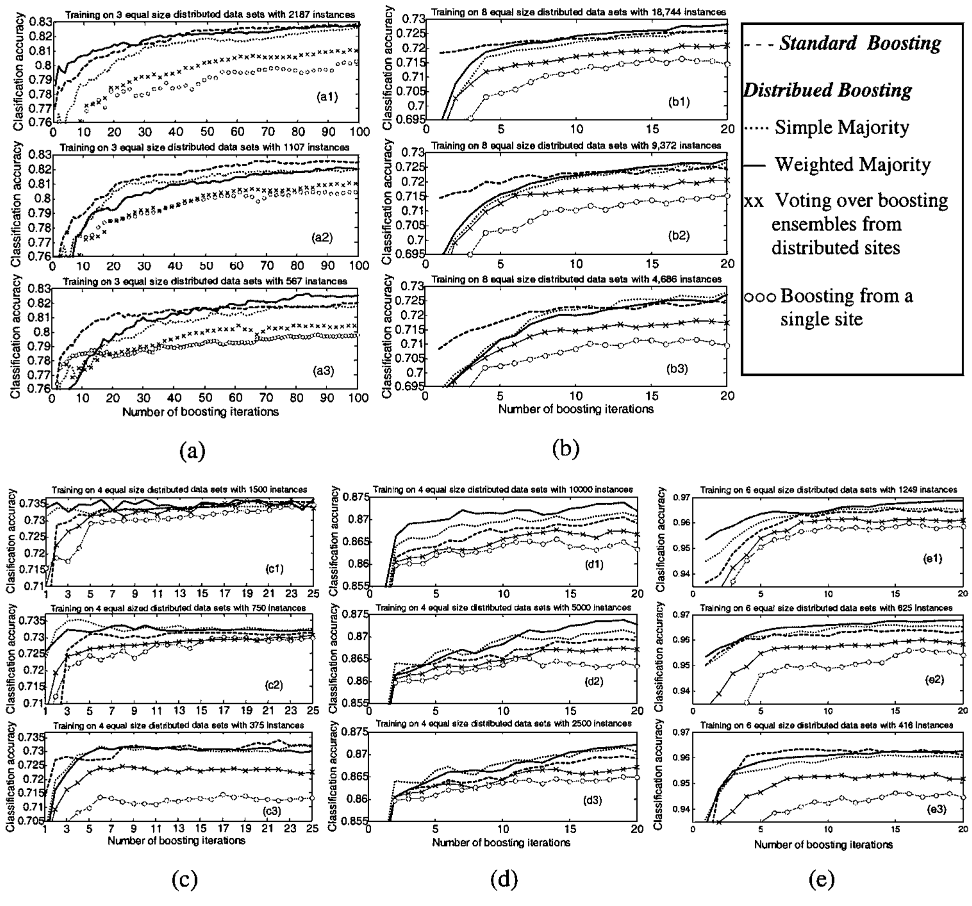

Figure 12. Out of sample classification accuracies of different distributed boosting algorithms. (a) Synthetic spatial data set; (b) Covertype data set; (c) LED data set; (d) Waveform data set; (e) Pen-based digit data set.

more accurate than "distributed bagging", and for simplicity reasons only those results are reported.

For each of the graphs shown at figure 12, 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. Similar to experiments with parallel boosting, changing the size of the data sets on distributed sites was used to investigate how the performance of the distributed boosting algorithm varied with the number of data points used for learning (figure 12).

Results from experiments performed on the synthetic spatial data sets and Covertype data sets indicate that the methods of simple and weighted majority voting of the classifiers constructed on multiple sites were successful when learning from distributed sites, since they achieved approximately the same classification accuracies as the standard boosting algorithm on merged data (figure 12(a) and (b)). 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. Figure 12(a) and (b) also demonstrate that the simple and weighted majority algorithms were more successful than the voting over boosting ensembles built independently on distributed sites. Finally, all distributed algorithms were more accurate than the boosting method applied to a single distributed site (figure 12(a) and (b)), thus indicating that the data distributions on distributed sites were different enough since learning from a single site could not achieve the same generalizability as when learning from a centralized data set.

When performing the experiments on the LED, Waveform and Pen-based digit data sets (figure 12(c), (d) and (e)), simple and weighted majority distributed boosting algorithms were consistently comparable in prediction accuracy to standard boosting on the centralized data. However, these majority algorithms also showed similar prediction accuracy to the voting over classifier ensembles built using boosting method, and were only slightly more accurate then the boosting method applied to a single distributed site, probably due to high homogeneity of data.

In addition, the effect of dividing the sampling weights $w_{j,t}$ by the factor acc^p, $(p = 0, 1, 2)$ was investigated for all three proposed 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 a drop in prediction accuracy. To alleviate this problem, the minimum size of the original data set, needed to be sampled from that site in order to cause only small and prespecified accuracy drop, is determined by empirical evaluation to be 15% of the original data set size. Otherwise, the best classifier built so far on a particular site was used when making a classifier ensemble. In our experiments on synthetic spatial data (figure 13(a)), 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 13(a3)), while preserving maximal prediction accuracy for experiments with large data sets (figure 13(a1)). The performed experiments on Covertype data sets showed similar behavior as the experiments on

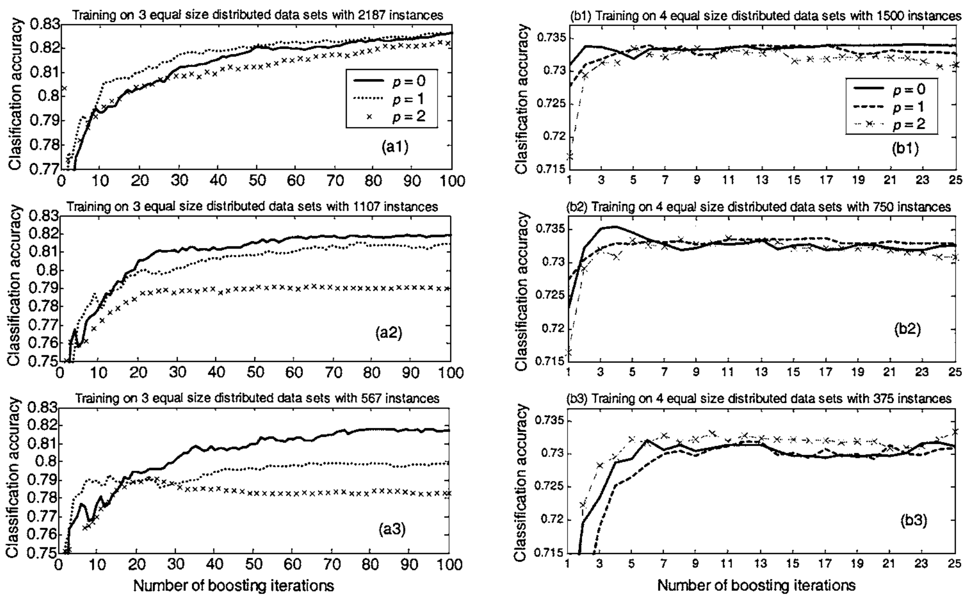

Figure 13. Out of sample averaged classification accuracies of the simple majority boosting algorithm with different weight modifications for distributed learning over (a) three synthetic spatial sites; (b) four sites containing LED data sets.

synthetic data set, while the experiments on LED (figure 13(b)), Waveform and Pen-based digit data sets showed similar prediction accuracy for all explored factors for updating the sampling weights $w_{j,t}$ (Table 2). This was probably due to homogeneous distributions in these data sets, where there were no extremely difficult examples that need to be emphasized.

Finally, for distributed boosting in a homogeneous environment, we also performed experiments 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 14).

Table 2. 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	Pen-based digit	LED	Waveform	Covertype
Simple majority	$p = 0$	82.7	96.5	73.4	87.0	72.6
	$p = 1$	82.6	96.3	73.3	86.9	72.7
	$p = 2$	82.2	96.1	73.1	86.8	72.5
Confidence-based weighting	$p = 0$	84.3	97.1	73.4	87.2	73.1
	$p = 1$	82.9	96.5	73.6	87.1	73.2
	$p = 2$	82.1	96.1	73.4	87.1	73.0

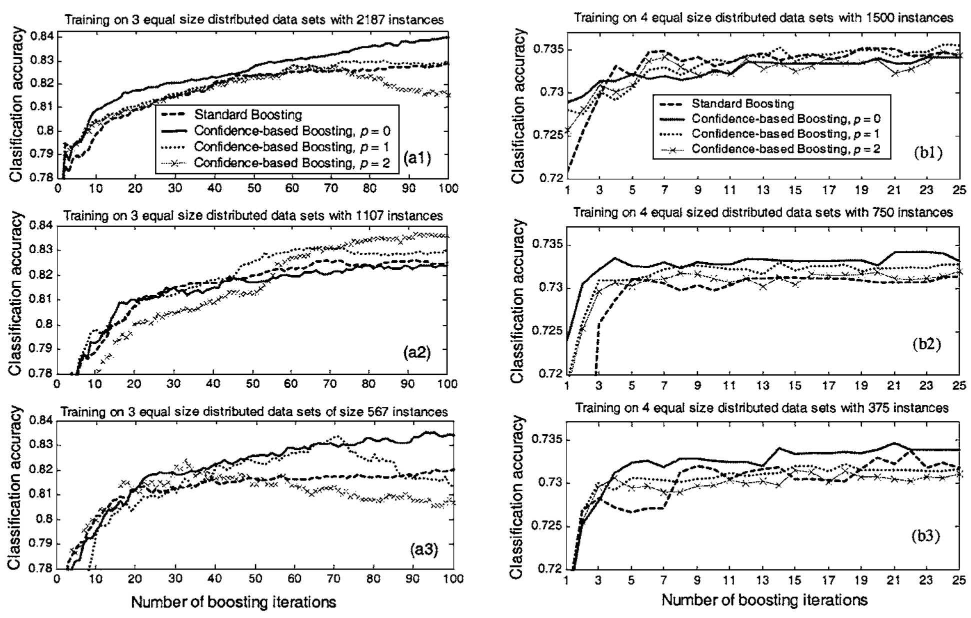

Figure 14. Out of sample averaged classification accuracies of confidence-based weighted combining classifiers in distributed boosting over (a) three synthetic spatial sites (b) four sites containing LED data sets.

The graphs in figure 14 show that the confidence-based combining classifiers achieved similar accuracy as the standard boosting applied on centralized data and the other methods considered for distributed boosting. In addition, for some values of parameter p, the confidence-based distributed boosting slightly outperformed all other boosting methods. The performed experiments on Covertype data sets using the confidence-based distributed boosting showed similar effects as experiments on synthetic spatial data sets, while on the other hand, the experiments performed on Waveform and Pen-based digit data sets demonstrated similar behavior to experiments carried out on LED data sets. Therefore, these results were not reported here. Unlike parallel 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 (figure 14(a3) and (b3)). The increase in prediction accuracy with the decreasing the data sets was probably due to the fact that the data sets on multiple sites 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. In the parallel boosting experiments (Section 3.1), the classifiers were learned over the same training set, and therefore this diversity was not apparent.

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 2).

3.3. Results for distributed boosting specialized experts in heterogeneous environments

For distributed boosting with heterogeneous databases, due to difficulties finding appropriate real life heterogeneous data sets, only synthetic spatial data sets generated through the spatial data simulator [2] were used, where we were able to perform controlled experiments on life-like heterogeneous data of various complexity. In our experiments for distributed learning explained in Section 2.3.1, all distributed sites had similar heterogeneous distributions with the same sets of attributes. Four independently generated synthetic spatial data sets were used each corresponding to five homogeneous data distributions made using our spatial data simulator. The attributes $f4$ and $f5$ were simulated to form five distributions in their attribute space ($f4$, $f5$) using the technique of feature agglomeration [20]. Furthermore, instead of using a single model for generating the target attribute on the entire spatial data set, a different data generation process using different relevant attributes was applied per each distribution. The degree of relevance was also different for each distribution. All four data sets had 6561 patterns with five relevant ($f1, \ldots, f5$) and five irrelevant attributes ($f6, \ldots, f10$), where the three data sets were used as the data from the distributed sites, and the fourth set was used as the test data set. The experiments for distributed boosting involving different homogeneous databases (Section 2.3.2) were performed on six synthetic spatial data sets each with a different distribution. Five of them were used for learning, and the sixth was a test set.

For distributed boosting specialized experts (Section 2.3.1), experiments were performed when all data sites had similar but a heterogeneous distribution. In addition to comparison to standard boosting and boosting specialized experts on centralized data, this distributed algorithm was also compared to the mixture of experts [13] method, adapted for distributed learning, where voting over mixture of experts from distributed sites is used to classify new instances.

Figure 15 shows that both methods of boosting specialized experts in centralized and distributed environments resulted in improved generalization (approximately 76–77% as compared to 72–73% obtained through standard and distributed boosting). Furthermore, both methods of boosting specialized experts outperformed the "distributed" version of mixture of experts, which achieved $75.2\% \pm 0.5\%$ classification accuracy and also the mixture of experts method applied on centralized data ($75.4\% \pm 0.4\%$ classification accuracy). When comparing to standard and distributed boosting, it was also evident that the methods of boosting specialized experts required significantly fewer iterations in order to reach the maximal prediction accuracy. After the prediction accuracy was maximized in our experiments, the overall prediction accuracy on a validation set, as well as the total classification accuracy on the test set, started to decline. The data set from one of the distributed sites used for learning served as a validation set. The phenomenon of deteriorating the classification accuracy was probably due to the fact that in the later iterations only data points that were difficult for learning were drawn and therefore the size of some identified clusters during those iterations started to decrease thus causing a deficiency in the number of drawn data

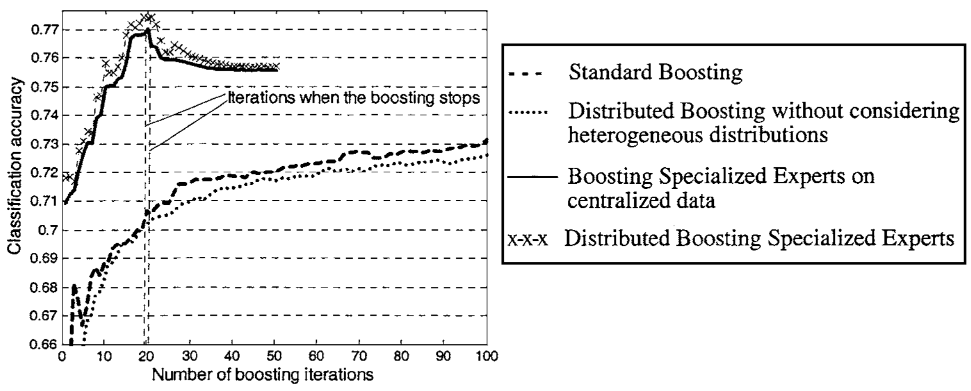

Figure 15. Out of sample averaged classification accuracies of distributed learning over three heterogeneous sites for synthetic spatial test data set (6561 instances) with 3-equal size classes and stemming from a similar mixture of 5 homogeneous distributions.

examples needed for successful learning. As a consequence, the prediction accuracy on these clusters begun to decline and hence the total prediction accuracy decreased, too. It is interesting to observe that the effect of accuracy drop is not noticed in the standard boosting, since the same number of examples is drawn in each iteration and only a single classifier is constructed on these examples, thus avoiding to have insufficient number of examples for learning.

A criterion for stopping the boosting algorithm early was to stop the procedure when the classification accuracy on the validation set started to decline. However, after approximately 20 additional boosting iterations the prediction accuracy was stabilized (figure 15). Although in practice the prediction accuracy on the test set does not necessarily start to drop in the same iteration as for validation set, in our experiments this difference was usually within two to three boosting iterations and did not significantly affect the total generalizability of the proposed method. However, the thorough inspection of noticed phenomenon prevails the scope of this paper and requires experiments on more data sets.

Two groups of experiments were performed when learning from sites with different homogeneous distributions and with the same set of attributes (Section 2.3.2). In the first group of experiments, five data sets with different distributions were used for learning and a data set with a distribution similar to the distribution from one of the existing multiple sites was used for testing (figure 16). The second group of experiments was related to learning from the same five data sets, but testing on the same data set with five homogeneous distributions as in the experiments when sites had similar heterogeneous distributions (figure 17).

The experimental results for distributed boosting in heterogeneous environment demonstrated that the method relying on computing Mahalanobis distance among the sites outperformed both the standard and alternative distributed boosting methods (figure 17). The result was a consequence of high heterogeneity in synthetic data sets. In such cases, the classifiers constructed on the sites with distribution very different from the distribution on the test data set only decreased accuracy of classifier ensembles.

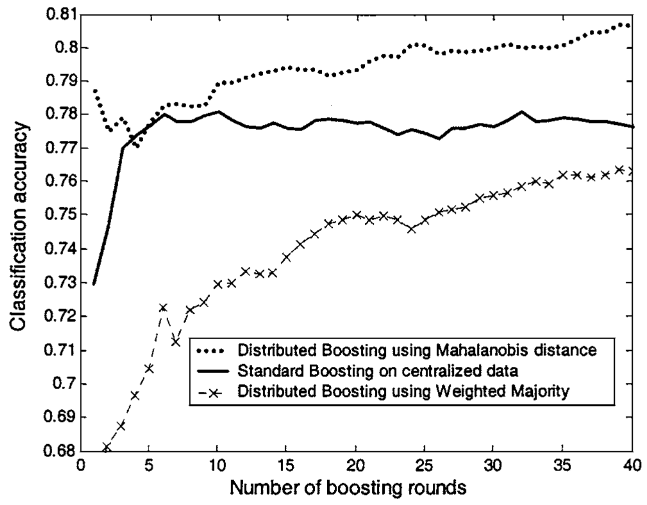

Figure 16. Testing distributed boosting methods on data stemming from a distribution similar to one of distributions from 5 learning sites.

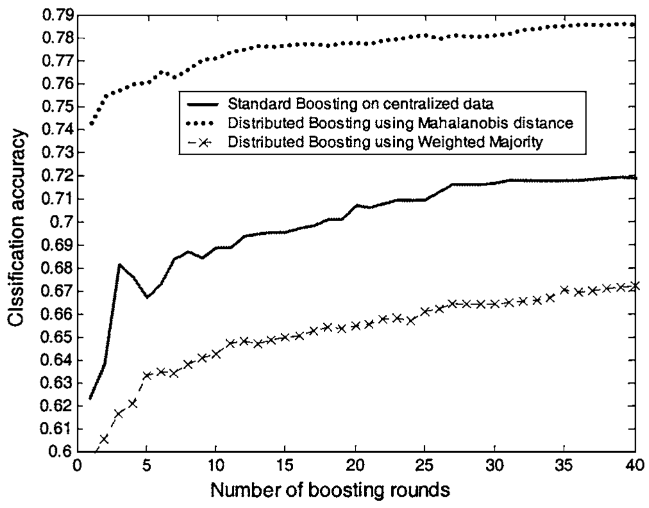

Figure 17. Testing distributed boosting methods on data stemming from a mixture of distributions from 5 learning sites.

4. Conclusion

A framework for parallel and distributed boosting is proposed. It is intended to efficiently learn classifiers over large, distributed and possibly heterogeneous 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 questions 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.

The proposed boosting algorithms successfully overcome these concerns under a variety of consideration, thus offering a fairly general method for effective and efficient learning in parallel and distributed environments.

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 [16] may be necessary to increase system throughput, while still maintaining the achieved prediction accuracy.

Although performed experiments have provided evidence that the proposed methods can be successful for parallel and 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 number of distributed sites and their sizes to the achieved prediction accuracy and to establish a satisfactory trade off.

Finally, the proposed methods can be adapted for on-line learning when new data become available periodically and when it is computationally expensive to rebuild a single classifier or an ensemble on the entire data set.

Acknowledgments

The authors are grateful to Dragoljub Pokrajac for providing simulated data and for his help in implementation design and to Celeste Brown for her useful comments.

References

1. J. Blackard, “Comparison of neural networks and discriminant analysis in predicting forest cover types,” Ph.D. dissertation, Colorado State University, 1998.
2. C.L. Blake and C.J. Merz, “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. L. Breiman and N. Shang, “Born again trees,” ftp://ftp.stat.berkeley.edu/pub/users/breiman/BAtrees.ps, 1996.

4. P. Chan and S. Stolfo, “On the accuracy of meta-learning for scalable data mining,” Journal of Intelligent Integration of Information, L. Kerschberg (Ed.), 1998.
5. S. Clearwater, T. Cheng, H. Hirsh, and B. Buchanan, “Incremental batch learning.” in Proc. of the Sixth Int. Machine Learning Workshop, Ithaca, NY, 1989, pp. 366–370.
6. J. Dy and C. Brodley, “Feature subset selection and order identification for unsupervised learning,” in Proc. of the Seventeenth Int. Conf. on Machine Learning, Stanford, CA, 2000, pp. 247–254.
7. W. Fan, S. Stolfo, and J. Zhang, “The application of Adaboost for distributed, scalable and on-line learning,” in Proc. of the Fifth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Diego, CA, 1999, pp. 362–366.
8. B. Flury, “A first course in multivariate statistics, Springer-Verlag: New York, NY, 1997.
9. Y. Freund and R.E. Schapire, “Experiments with a new boosting algorithm,” in Proc. of the Thirteenth Int. Conf. on Machine Learning, San Francisco, CA, 1996, pp. 325–332.
10. J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” The Annals of Statistics, vol. 38, no. 2, pp. 337–374, 2000.
11. M. Hagan and M.B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” IEEE Trans. on Neural Networks, vol. 5, pp. 989–993, 1994.
12. S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall: Englewood Cliffs, NJ, 1999.
13. M. Jordan and R. Jacobs, “Hierarchical mixture of experts and the EM algorithm,” Neural Computation, vol. 6, no. 2, pp. 181–214, 1994.
14. A. Lazarevic, T. Fiez, and Z. Obradovic, “Adaptive boosting for spatial functions with unstable driving attributes,” in Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Kyoto, Japan, 2000, pp. 329–340.
15. A. Lazarevic and Z. Obradovic. “Boosting localized classifiers in heterogeneous databases,” in Proc. on First Int. SIAM Conf. on Data Mining, Chicago, IL, 2001.
16. A. Lazarevic and Z. Obradovic, “The effective pruning of neural network ensembles,” in Proc. of IEEE Int. Joint Conf. on Neural Networks, Washington, D.C., 2001, pp. 796–801.
17. A. Lazarevic, D. Pokrajac, and Z. Obradovic, “Distributed clustering and local regression for knowledge discovery in multiple spatial databases,” in Proc. 8th European Symp. on Art. Neural Networks, Bruges, Belgium, 2000, pp. 129–134.
18. A. Lazarevic, X. Xu, T. Fiez, and Z. Obradovic, “Clustering-Regression-Ordering steps for knowledge discovery in spatial databases,” in Proc. IEEE/INNS Int. Conf. on Neural Networks, Washington D.C., No. 345, Session 8.1B, 1999.
19. L. Mason, J. Baxter, P. Bartlett, and M. Frean. “Function gradient techniques for combining hypotheses,” in Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans (Eds.), MIT Press: Cambridge, MA, 2000, chap. 12.
20. D. Pokrajac, T. Fiez, and Z. Obradovic, “A data generator for evaluating spatial issues in precision agriculture,” Precision Agriculture, in press.
21. D. Pokrajac, A. Lazarevic, V. Megalooikonomou, and Z. Obradovic, “Classification of brain image data using measures of distributional distance,” in Proc. 7th Annual Meeting of the Organization for Human Brain Mapping, London, UK, 2001.
22. A. Prodromidis, P. Chan, and S. Stolfo, “Meta-Learning in distributed data mining systems: Issues and approaches,” in Advances in Distributed Data Mining, H. Kargupta and P. Chan (Eds.), AAAI Press: Menlo Park, CA, 2000.
23. F. Provost and D. Hennesy, “Scaling Up: Distributed machine learning with cooperation,” in Proc. of the Thirteenth National Conf. on Artificial Intelligence, Portland, OR, 1996, pp. 74–79.
24. M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: The RPROP algorithm,” in Proc. of the IEEE Int. Conf. on Neural Networks, San Francisco, CA, 1993, pp. 586–591.
25. J. Sander, M. Ester, H.-P. Kriegel, and X. Xu “Density-Based clustering in spatial databases: The algorithm GDBSCAN and its applications,” Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 169–194, 1998.
26. J. Shafer, R. Agrawal, and M. Mehta, “SPRINT: A scalable parallel classifier for data mining,” in Proc. of the 22nd Int. Conf. on Very Large Data Bases, Mumbai (Bombay), India, 1996, pp. 544–555.
27. P. Sollich and A. Krogh, “Learning with ensembles: How over-fitting can be useful.” Advances in Neural Information Processing Systems, vol. 8, pp. 190–196, 1996.

28. M. Sreenivas, K. AlSabti, and S. Ranka, "Parallel out-of-core decision tree classifiers," in Advances in Distributed Data Mining, H. Kargupta and P. Chan (Eds.), AAAI Press: Menlo Park, CA, 2000.
29. A. Srivastava, E. Han, V. Kumar, and V. Singh, "Parallel formulations of decision-tree classification algorithms," Data Mining and Knowledge Discovery, vol. 3, no. 3, pp. 237–261, 1999.
30. P. Utgoff, "An improved algorithm for incremental induction of decision trees," in Proc. of the Eleventh Int. Conf. on Machine Learning, New Brunswick, NJ, 1994, pp. 318–325.
31. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "Parallel algorithms for discovery of association rules," Data Mining and Knowledge Discovery: An International Journal, special issue on Scalable High-Performance Computing, vol. 1, no. 4, pp. 343–373, 1997.