

# Collaborative Learning by Boosting in Distributed Environments

Shijun Wang

*Diagnostic Radiology Department, National Institutes of Health, Bethesda, MD 20892, USA*

*sjwang05@gmail.com*

Changshui Zhang

*Department of Automation, Tsinghua University, Beijing 100084, China*

*zcs@mail.tsinghua.edu.cn*

## Abstract

*In this paper we propose a new distributed learning method called distributed network boosting (DNB) algorithm for distributed applications. The learned hypotheses are exchanged between neighboring sites during learning process. Theoretical analysis shows that the DNB algorithm minimizes the cost function through the collaborative functional gradient descent in hypotheses space. Comparison results of the DNB algorithm with other distributed learning methods on real data sets with different sizes show its effectiveness.*

## 1 Introduction

With rapid progress of the telecommunication and computer network technologies, we are facing more and more distributed applications with massive data sets. It is not feasible to centralize the data physically dispersed in diverse geographic locations in some circumstances. Because they may be too large to fit into the computer memory or they are private and can not be exposed to other sites. There are two main directions developed in the machine learning community to deal with massive data sets from multiple sites: data subsampling [1] and distributed approaches [2, 3, 4]. If we assume that we do not need all the data available, then we may subsample the data and use traditional data mining methods to handle them [1]. On the other hand, in order to make full use of data, the distributed or parallel approaches tries to learn from multiple data resources, e.g. parallel decision trees[2], DVote and DRvote [3], etc.

The DVote and DRvote algorithms [3] are based on the pasting votes method proposed by Breiman [6]. The two algorithms build hundreds or thousands of classifiers on small subsets of data in a distributed environment. Whereafter, A. Lazarevic et al. proposed a paral-

lel algorithm called distributed boosting algorithm (DB) [4] based on boosting[5]. In the DB algorithm the training processes on different sites are proceeded concurrently. The weights of samples on each site are updated according to the learned hypotheses. Afterwards the sum of weights of the samples on each site are broadcasted which helps each node to build a global weight distribution of all the instances. The training set of each site for the next round is sampled from the global distribution with all the samples belonging to other sites are deleted.

In this paper, we propose a new method called distributed network boosting algorithm (DNB) for classification problems in distributed environments. The DNB method is based on the collaborative functional gradient technique for combining hypotheses in functional space. Theoretical analysis shows that it can learn the target hypothesis through the cooperation between different sites with all the data in each site kept private. The DNB algorithm, unlike the distributed boosting algorithm [4] which tries to emulate the global distribution of samples by broadcast, just uses local information and requires less memory space than that of the DB algorithm.

## 2 The distributed network boosting algorithm

In distributed applications, each site will have a classifier or base learning algorithm. They compose a classifier network in which nodes are classifiers and links between nodes represent the relationship between classifier pairs [7]. If there is a link between node  $A$  and node  $B$ , then classifier  $A$  will exchange information with classifier  $B$  during the learning process. The topology of the classifier network is determined by real distributed applications. The link represents the commu-

nication link between two sites. The basic idea of out DNB method is that we try to learn the target hypothesis through the cooperation of classifiers distributed on different sites. To reduce the communication and computation cost incurred by exchanging samples between sites, here we exchange the learned hypotheses between neighbors during the training of the DNB algorithm. So the base learning algorithm in each site does not need to know the samples in its neighbor sites, they just need to know what their neighbors learned. This scenario is very similar to the propagation and accumulation process of knowledge in our human society.

The dynamic integration approach contains two phases. In the learning phase, given train set  $S_k, k = 1, 2, \dots, K$  for each base learning algorithm on different sites, each site  $k$  maintains a weight distribution  $D_{k,t}(i)$  for  $k = 1, \dots, K, t = 1, \dots, T$  of samples  $x_i, i = 1, 2, \dots, l_k$  on that site respectively ( $l_k$  is the number of train samples in node  $k$ ). Then a classifier on each site is built by the training instances sampled from the train data according to the weight distribution of the train data it holds. After that, the weights of the instances on each node are updated according to the classification result of the node and its neighbors. The classifier network is trained  $T$  rounds in such way. In the application phase, for a coming new sample at any site  $k$ , its label is determined by weighted vote of all the hypotheses learned by site  $k$  and its direct neighbors on the classifier network. The proposed DNB algorithm is described as follows:

**Input:** Network  $N$  with  $K$  nodes; Training set  $S_k$  for each site  $k, k = 1, 2, \dots, K$ ; Training rounds  $T$ .

**Initialize:** For each node  $k, D_{k,1}(i) = 1/l_k$  where  $l_k$  is the number of samples in  $S_k, k = 1, 2, \dots, K$ .

**Do for:** 1. Generate a replicate training set  $T_{k,t}$  of size  $l_k$ , by weighted sub-sampling with replacement from train set  $S_k$  for  $k = 1, 2, \dots, K$ ;

2. Train the classifier (node)  $C_k$  in the classifier network with respect to the training set  $T_{k,t}$  and obtain hypothesis

$h_{k,t} : x \mapsto \{-1, +1\}$  for  $k = 1, 2, \dots, K$ ;

3. Calculate weighted error:

$$\varepsilon_{k,t} = \sum_{x_i \in S_k} D_{k,t}(i) I[y_i \neq h_{k,t}(x_i)], k = 1, 2, \dots, K$$

( $I$  is the indication function);

4. Hypothesis weight  $\alpha_{k,t} = 0.5 * \log((1 - \varepsilon_{k,t})/\varepsilon_{k,t}), k = 1, 2, \dots, K$ ;

5. Update the weight of instance  $i$  of node  $k$ :  $D_{k,t+1}(i) =$

$$D_{k,t}(i) \beta \left( -\alpha_{k,t} y_i h_{k,t}(x_i) - \sum_n \alpha_{n,t} y_i h_{n,t}(x_i) \right) / Z_{k,t},$$

where node  $n$  is the direct neighbor of node  $k, Z_{k,t}$  is a normalization const.

**Output:** Final hypotheses

$$H_{k,T}(x) = \sum_{t=1}^T (\alpha_{k,t} h_{k,t}(x) + \sum_n \alpha_{n,t} h_{n,t}(x)), \quad (1)$$

$k = 1, 2, \dots, K$  and  $n$  is the direct neighbor of node  $k$ .

### 3 Collaborative functional gradient technique for combining hypotheses in distributed environments

The ensemble learning problem is closely related with optimization problem in hypothesis space [8]. In this section, we propose a collaborative functional gradient technique in hypotheses space and show that the DNB algorithm minimizes the classification objective function through it.

At first, we assume that in a distributed environment, every site or node has its own training set  $S_k, k = 1, \dots, K$ . For each node  $k$ , we define the output of the ensemble at training round  $t$  as:

$$H_{k,t}(x) = \sum_{s=1}^t \alpha_{k,s} h_{k,s}(x) + \sum_n \sum_{s=1}^t \alpha_{n,s} h_{n,s}(x), \quad (2)$$

where node  $n$  is the direct neighbor of node  $k$  and  $\alpha_{k,s}$  is the weight of hypothesis  $h_{k,s}, k = 1, 2, \dots, K, s = 1, 2, \dots, t$ . The above formulation shows that the ensemble output of node  $k$  contains not only all the hypotheses learned in the node  $k$ , but also the hypotheses learned by its neighbors during past training process.

Like AdaBoost [5], we define the optimization objective function  $C$  of sample margins in each node  $k$  at training rounds  $t$  as

$$C(H_{k,t}) = \frac{1}{l_k} \sum_{i=1}^{l_k} \exp \left\{ -y_i \sum_{s=1}^t \alpha_{k,s} h_{k,s}(x_i) \right\} \times \exp \left\{ -y_i \sum_n \sum_{s=1}^t \alpha_{n,s} h_{n,s}(x_i) \right\} \quad (3)$$

To minimize the objective function, a natural way is finding the negative gradient direction of  $C$  in hypotheses space. Because it is very hard to directly find the negative gradient direction of the cost function  $C$ , we choose an alternative way:

At training round  $t + 1$ , we expect the inner product of the new learned hypothesis  $h_{k,t+1}$  and the negative gradient direction of cost function  $C$  at  $H_{k,t}$

$$\begin{aligned} & - \langle \nabla C(H_{k,t}), h_{k,t+1} \rangle \\ &= - \frac{1}{l_k} \sum_{i=1}^{l_k} y_i h_{k,t+1}(x_i) C'(y_i H_{k,t}(x_i)) \end{aligned} \quad (4)$$

is maximized. After a simple transformation, we get

$$\begin{aligned}
& - \langle \nabla C(H_{k,t}), h_{k,t+1} \rangle \\
& = \frac{1}{l_k^2} \sum_{i: y_i = h_{k,t+1}(x_i)} \exp \left\{ -y_i \sum_{s=1}^t \alpha_{k,s} h_{k,s}(x_i) \right\} \\
& \times \exp \left\{ -y_i \sum_n \sum_{s=1}^t \alpha_{n,s} h_{n,s}(x_i) \right\} \\
& - \frac{1}{l_k^2} \sum_{i: y_i \neq h_{k,t+1}(x_i)} \exp \left\{ -y_i \sum_{s=1}^t \alpha_{k,s} h_{k,s}(x_i) \right\} \\
& \times \exp \left\{ -y_i \sum_n \sum_{s=1}^t \alpha_{n,s} h_{n,s}(x_i) \right\}.
\end{aligned}$$

If we define the weight of sample  $i$  at round  $t$  in node  $k$  as

$$\begin{aligned}
D_{k,t+1}(i) & = \exp \left\{ -y_i \sum_{s=1}^t \alpha_{k,s} h_{k,s}(x_i) \right\} \\
& \times \exp \left\{ -y_i \sum_n \sum_{s=1}^t \alpha_{n,s} h_{n,s}(x_i) \right\}.
\end{aligned} \tag{5}$$

Then finding  $h_{k,t+1}$  to maximize  $-\langle \nabla C(H_{k,t}), h_{k,t+1} \rangle$  at round  $t+1$  is equivalent to finding  $h_{k,t+1}$  to minimize

$$\sum_{i: h_{k,t+1}(x_i) \neq y_i} D_{k,t+1}(i), \tag{6}$$

which means that we should find hypothesis  $h_{k,t+1}$  that has the minimum weighted error rate under current weight distribution of samples. Because the optimization is done by the cooperation of learners on the classifier network, we call it collaborative functional gradient technique in hypotheses space.

According to the definition of the update of sample weight above, if we represent it as iterative form according to training round  $t$ , then

$$\begin{aligned}
D_{k,t+1}(i) & = D_{k,t}(i) \exp \{ -y_i \alpha_{k,t} h_{k,t}(x_i) \} \\
& \times \exp \left\{ -y_i \sum_n \alpha_{n,t} h_{n,t}(x_i) \right\}.
\end{aligned} \tag{7}$$

It is just the weight update equation of the DNB algorithm. The above equation shows that: for any node  $k$ , the DNB algorithm not only considers the hypothesis learned at training round  $t$ , but also the hypotheses learned by its neighbors when the weights of samples are updated. So the DNB algorithm implements the collaborative functional gradient technique in hypotheses space through importing of neighbor information and minimizes the cost function step by step.

## 4 Experiments

In this section, we compare the distributed network boosting algorithm with distributed boosting and Divote algorithms on several benchmark data sets selected from UCI repository (<http://www.ics.uci.edu/>). In the experiments below, the C4.5 decision tree algorithm is employed as a base classifier in all the ensemble methods without pruning.

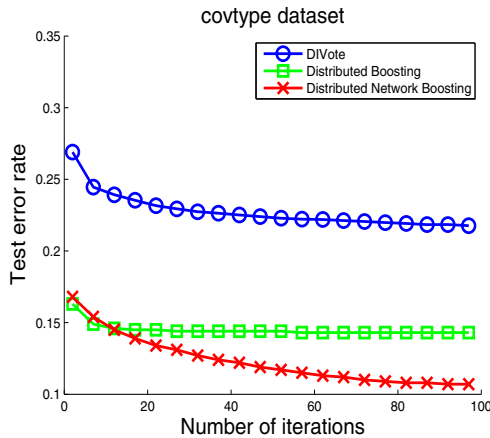
We first tested the distributed ensemble learning methods on several small-sized datasets whose samples are less than 4000. We performed statistical tests to compare the three ensemble learning algorithms. For each problem we generated 50 random partitions into training and test sets with proportion 6 : 4. For all the distributed algorithms, the training samples were uniformly distributed in 4 nodes (sites). Then we trained each distributed learning algorithm and computed its test error on each partition. The boosting iterations were all 100. For the Divote [3], the size of each bite was 40. For the DNB algorithm, the 4 nodes were fully connected.

The comparison results are shown in table 1. We also tested the C4.5 with complete data available for comparison. In the table we show the average error rate (err) and standard deviation (std) of the 50 random tests for each data set tested by each algorithm. We also show the results of significance tests ( $t$ -hypothesis test at significance level 0.05) of the C4.5, Divote, DB algorithms with the DNB algorithm. “+” and “-” mean that there is a significant difference between the results of the two algorithms compared. From the table we can find that the DNB algorithm shows better generalization ability compared to the Divote and DB algorithms. It also achieves higher accuracy compared with the C4.5 algorithm with centralized data.

To show the performance of the distributed network boosting algorithm on large-scale data sets, we compared the Divote, DB and DNB algorithms on the covtype dataset. The covtype data set was selected from UCI repository which has 581,012 instances in total. It has 7 classes and 54 continuous attributes. We split the data into training set and test set with the size 100,000 and 481,012 respectively. For each distributed learning algorithm, we partitioned the training set into ten disjoint partitions. For the Divote algorithm, the size of each bite was 800. For the DNB algorithm, a fully connected network was used. Figure 1 shows the test error rates of the three methods on each boosting iteration. The DNB algorithm achieves higher accuracy compared with other two methods on this large scale dataset.

**Table 1. Comparison results of the DNB algorithm with the C4.5, DIVote and DB algorithms.**

Name	C4.5	DIVote	DB	DNB			
	<i>err</i> ± <i>std</i>	<i>err</i> ± <i>std</i>	<i>err</i> ± <i>std</i>	S1	S2	S3	<i>err</i> ± <i>std</i>
credit-g	.285 ± .020	.265 ± .020	.255 ± .017	+	+		.252 ± .017
heart-c	.245 ± .042	.169 ± .032	.212 ± .029	+	-	+	.184 ± .031
ionosphere	.114 ± .027	.086 ± .024	.110 ± .031	+		+	.088 ± .028
kr-vs-kp	.009 ± .003	.022 ± .004	.013 ± .004		+	+	.010 ± .003
sick	.015 ± .004	.022 ± .003	.020 ± .003	-	+	+	.017 ± .004
soybean	.110 ± .022	.218 ± .028	.099 ± .023	+	+	+	.088 ± .015
splice	.067 ± .007	.069 ± .011	.073 ± .015	+	+	+	.058 ± .007
tic-tac-toe	.179 ± .024	.228 ± .021	.061 ± .024	+	+	+	.050 ± .021
vehicle	.288 ± .024	.291 ± .021	.260 ± .026	+	+	+	.245 ± .016
vowel	.266 ± .028	.388 ± .032	.154 ± .027	+	+		.148 ± .024



**Figure 1. Comparisons of the Divote, DB and DNB algorithms on the test set under different training iterations for the covtype dataset.**

## 5 Conclusions

In this paper, we propose a new distributed learning algorithm called the distributed network boosting algorithm. During the training process, the hypothesis learned in each round and each node is shared by its neighbor nodes (sites). With the help of exchanged hypotheses, the learned classifiers become more diverse and are robust to noise. The theoretic analysis shows that it minimizes the classification cost function through the collaborative functional gradient descent in hypotheses space. To validate the proposed method, we compare it with the C4.5, DIVote and distributed boosting algorithms on the UCI datasets with different sizes. Experimental results show that the DNB algorithm has higher generalization ability compared with others.

**Acknowledgement:** This work was supported by the National 863 project(No. 2006AA10Z210).

## References

- [1] F. Provost and V. Kolluri "A survey of methods for scaling up inductive algorithms," *Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 131-169, 1999.
- [2] 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.
- [3] N.V. Chawla, L.O. Hall, K.W. Bowyer and W.P. Kegelmeyer "Learning ensembles from bites: A scalable and accurate approach," *JMLR*, vol. 5, pp. 421-451, 2004.
- [4] A. Lazarevic and Z. Obradovic "Boosting Algorithms for Parallel and Distributed Learning," *Distributed and Parallel Databases*, vol. 11, pp. 203-229, 2002.
- [5] Y. Freund and R.E. Schapire "Experiments with a new boosting algorithm," *ICML*, 1996.
- [6] L.R. Breiman "Pasting Small Votes for Classification in Large Databases and On-Line," *Machine Learning*, vol. 36, pp. 85-103, 1999.
- [7] S. Wang and C. Zhang "Network game and boosting," *The 16th European Conference on Machine Learning*, vol. 3720, pp. 461-472, 2005.
- [8] L. Mason, J. Baxter, P.L. Bartlett and M. Frean "Functional gradient techniques for combining hypotheses," *Advances in Large Margin Classifiers*, 2000.