A Fast Online Learning Algorithm for Distributed Mining of BigData

[Extended Abstract]

Yu Zhang *
University of California, Los Angeles
yuzhang@ucla.edu

Daby Sow
IBM T.J. Watson Research Center
sowdaby@us.ibm.com

Deepak Turaga
IBM T.J. Watson Research Center
turaga@us.ibm.com

Mihaela van der Schaar
University of California, Los Angeles
mihaela@ee.ucla.edu

ABSTRACT

BigData analytics require that distributed mining of numerous data streams is performed in real-time. Unique challenges associated with designing such distributed mining systems are: online adaptation to incoming data characteristics, online processing of large amounts of heterogeneous data, limited data access and communication capabilities between distributed learners, etc. We propose a general framework for distributed data mining and develop an efficient online learning algorithm based on this. Our framework consists of an ensemble learner and multiple local learners, which can only access different parts of the incoming data. By exploiting the correlations of the learning models among local learners, our proposed learning algorithms can optimize the prediction accuracy while requiring significantly less information exchange and computational complexity than existing state-of-the-art learning solutions.

1. INTRODUCTION

BigData analytics involve processing heterogeneous data from various distributed data sources producing complementary datasets [1]. Hence, the data sets are not only characterized by their extremely large volumes but also by their heterogeneity and the distributed acquisition of data. Distributed data mining techniques have been proposed in the literature to process such distributed data sets [1]. Different from traditional centralized data mining systems where a single learner has full access to the global dataset [2], distributed data mining systems typically use ensemble learning techniques consisting of a hierarchy of multiple local learners operating on subsets of the global dataset at the lowest level of the hierarchy, and one or more ensemble learners combining the outputs of all the local learners [3]. While expanding current frontiers in knowledge acquisition by allowing the analysis of larger and more diverse datasets, such distributed data mining systems also come with significant design challenges that are the focal point of this work.

(1) *Limited data access.* In distributed data mining, each local learner has only limited access to the entire dataset [3]. There are two types of data partition [4]. In the instance-based distributed mining, each local learner accesses a subset of instances (with all features) of the entire dataset; while in the feature-based distributed mining, each local learner accesses a subset of the feature space of all instances. In this work, we specifically focus on the scenario with feature-distributed data.

(2) *Limited communication capability.* Due to the large data volume and the limited communication capabilities of individual learners, it is costly to centralize unprocessed data within the system, which makes the centralized mining expensive if not infeasible [3].

(3) *Coping with large rates of non stationary data.* The data accessible to local learners also grows fast and the statistical properties of the data may change dynamically over time.

To address these challenges, we propose a general framework for online distributed machine learning. The main novelties of this work are:

(1) *Online learning.* Different from the majority of learning algorithms in distributed data mining that are offline , our algorithm trains the learning models on different learners in an online manner, which only requires *one pass* over the training data.

(2) *Coordination among learners.* We also address the following questions in our design: (i) how the ensemble learner *optimally* selects the learning models of local learners? (ii) how local learners *optimally* update their learning models cooperatively?

(3) *Cross learner correlation exploitation.* We partition local learners into correlated groups by estimating the cross correlations of the learning models between them. Learners within the same group have high correlations and will be coordinated with each other in the training, while learners

*This work is supported in part by the National Science Foundation award under grant CNS 1016081

from different groups are trained independently with each other. Therefore, we can control both the *computation complexity* and *information exchange* incurred to local learners by adjusting cross correlation thresholds used to group them.

The rest of this paper is organized as follows. In Section 2, we derive conditions for optimality of the distributed learning and subsequently derive the corresponding algorithm. Section 3 illustrates the performance of our learning algorithm on an important problem in network security: network intrusion detection. In Section 4, we review the existing literature in distributed data mining. We end the paper in Section 5 with concluding remarks.

2. SYSTEM FRAMEWORK

We consider a distributed data mining system that consists of one ensemble learner and K local learners from the set $\mathcal{K} = \{1, \ldots, K\}$. Time is divided into discrete periods. In each period n, there is one data instance $\mathbf{x}^n = ((x_i^n)_{i=1}^d, y^n)$ entering the system, where $(x_i^n)_{i=1}^d$ are the features and $y \in \mathcal{Y}$ is the label for this instance. For each incoming instance, each local learner $k \in \mathcal{K}$ observes a subset of features $\mathcal{A}_k \subseteq \{1, \ldots, d\}$. We assume that $\bigcup_{k\in\mathcal{K}} \mathcal{A}_k = \{1, \ldots, d\}$ such that every feature is used by some local learners.

For each time period n, each local learner $k \in \mathcal{K}$ makes a local prediction $\hat{y}_k^n$ of the label y^n based on its observable part $(x_j^n)_{j\in\mathcal{A}_k}$ from the incoming instance $\mathbf{x}^n$. Each local learner then submits its prediction to the ensemble learner. The ensemble learner gathers the local predictions and use them to generate a final ensemble prediction for this instance $\mathbf{x}^n$. In what follows, we present techniques to train both the local learners as well as the ensemble learners with the goal to maximize the accuracy of the ensemble prediction. To realize this, we assume that each local learner k maintains a finite number of prediction functions defined as $\mathcal{F}_k = \left\{f : \prod_{j\in\mathcal{A}_k} \mathcal{X}_j \to \mathbb{R}\right\}$, where $\mathcal{X}_j$ denotes the value space of feature x_j. Here, each prediction function can be interpreted as a (trained) classifier that is ready to use. In this work, we assume that each local learner combines its available prediction functions using a linear regressor in order to make its local prediction. Let $\mathbf{f}_k = (f_{k1}, \ldots, f_{k|\mathcal{F}_k|})$ represent all the prediction functions for a local learner k. Local learner k maintains a weight vector $\mathbf{b}_k \in \mathbb{R}^{|\mathcal{F}_k|}$ and its local prediction at period n is generated by $\hat{y}_k^n = \langle \mathbf{b}_k, \mathbf{f}_k((x_j^n)_{j\in\mathcal{A}_k})\rangle$.

The ensemble learner then aggregates the local predictions from all the local learners and uses a linear regressor as its prediction function to generate the final output as $\hat{y}^n = \sum_{k\in\mathcal{K}} w_k \hat{y}_k^n$, where $\mathbf{w} = \{w_k\}_{k=1}^K$ is the vector of weights assigned to local learners. In this work, we assume that the prediction of each local learner is unbiased and hence, the sum of all weights is normalized to 1: $\sum_{k=1}^K w_k = 1$.

We aim at designing algorithms that are able to determine learning models minimizing the mean-square error on the predictions over the given instances, under regularization constraints of the weight vectors of learners. The corresponding optimization problem at each period n is expressed as follows:

$$\begin{aligned} \min_{\{\mathbf{b}_k\}_{k=1}^K} \min_{\mathbf{w}} & \sum_{m=1}^{n} \left(y^m - \sum_{k=1}^{K} w_k \langle \mathbf{b}_k, \mathbf{f}_k((x_j^m)_{j\in\mathcal{A}_k})\rangle\right)^2, \\ \text{s.t.} & \parallel \mathbf{w} \parallel_1 = 1, \\ & \parallel \mathbf{b}_k \parallel_2 \leq \lambda_k, \ \forall k \in \mathcal{K}. \end{aligned} \tag{1}$$

Algorithm 1 Ensemble weight update

$\nabla^n = 2(\langle \mathbf{w}, \mathbf{g}^n\rangle - y^n)\mathbf{g}^n$
$\mathbf{w}^{n+1} = (1 - \frac{1}{n})\mathbf{w}^n - \frac{1}{\alpha n}\nabla^n$
Project $\mathbf{w}^{n+1}$ to the ℓ_1 ball $\parallel \mathbf{w} \parallel_1 = 1$

To solve this problem, we propose online algorithms updating the weight vector $\mathbf{w}$ at the ensemble learner and the learning model $\mathbf{b}_k$ at each local learner. In the rest of this section, we discuss details of our proposed algorithms.

The update of $\mathbf{w}$ at the ensemble learner, when $(\mathbf{b}_k^n)_{k=1}^K$ are fixed, can be regarded as a regression problem over $\mathbf{w}$. By rewriting the square loss in the matrix form, it can be seen that the ensemble learner needs to update $\mathbf{w}$ by solving the following problem:

$$\min_{\mathbf{w}} \parallel \mathbf{y}^n - \mathbf{G}^n\mathbf{w} \parallel_2^2, \text{s.t.} \ \parallel \mathbf{w} \parallel_1 = 1. \tag{2}$$

Here $\mathbf{y} = (y^1, y^2, \ldots, y^n)^T$ and $\mathbf{G}^n \in \mathbb{R}^{n\times K}$ is the history of local predictions from local learners with $[\mathbf{G}^n]_{mk} = \hat{y}_k^m, \ \forall m \leq n$.

Using an Ordinary Least Square (OLS) estimator, the problem (2) can be solved as:

$$\mathbf{w}^n = \frac{\left((\mathbf{C}^n)^T\mathbf{C}^n\right)^{-1}\mathbf{1}}{\mathbf{1}^T\left((\mathbf{C}^n)^T\mathbf{C}^n\right)^{-1}\mathbf{1}}, \tag{3}$$

with the minimum (empirical) ensemble training residual being

$$r(\mathbf{C}^n) = \frac{1}{\mathbf{1}^T\left((\mathbf{C}^n)^T\mathbf{C}^n\right)^{-1}\mathbf{1}}. \tag{4}$$

The matrix $\mathbf{C}^n \in \mathbb{R}^{n\times K}$ stores the training residuals of the local learners with $[\mathbf{C}^n]_{mk} = y^m - \hat{y}_k^m, \ \forall m \leq n$. Thus, $\left(\mathbf{C}^n\right)^T\mathbf{C}^n$ can be interpreted as the empirical covariance matrix between local learners.

Storing $(\mathbf{C}^n)^T\mathbf{C}^n$ and computing its inverse in each time period is computationally expensive. Therefore, we propose an online algorithm using the stochastic gradient descent method [16] to update $\mathbf{w}$ on-the-fly, without requiring knowledge of the entire history of training residuals. To do so, instead of estimating the loss function in (2), we estimate its gradient $\nabla^n = 2(\langle \mathbf{w}, \mathbf{g}^n\rangle - y^n)\mathbf{g}^n$, where $\mathbf{g}^n = \{\hat{y}_k^n\}_{k=1}^K$. The algorithm for the update of $\mathbf{w}$ at the ensemble learner in each period n is illustrated in Algorithm 1, with α specifying its learning rate.

With $\mathbf{w}$ being updated, we also need to update the $\{\mathbf{b}_k\}_{k=1}^K$ in order to minimize $r(\mathbf{C}^n)$, which is equivalent to maximizing its inverse:

$$\max_{\mathbf{C}^n} \mathbf{1}^T\left((\mathbf{C}^n)^T\mathbf{C}^n\right)^{-1}\mathbf{1}. \tag{5}$$

The basic learning procedure at each local learner in each time period n can be briefly formulated as follows. After the ensemble learner updates $\mathbf{w}$, it sends the training information to the local learners. Each local learner then uses this information to update its own weight vector $\mathbf{b}_k$ maintained for its classifiers. Intuitively, it would be the best to train all local learners in a cooperative manner. That is, when each local learner updates its learning model, it takes into consideration the training residuals of all the other local learners in order to minimize the error in the final prediction output by the ensemble learner. However, such approach

Algorithm 2 Correlated weight update

$\hat{y}_k^n = \langle \mathbf{b}_k, \mathbf{f}_k((x_j^n)_{j\in\mathcal{A}_k})\rangle$.
Determine the set of correlated local learners to local learner k as Cov_k
$\nabla_k^n = 2\sum_{i\in Cov_k} w_i^{n+1}(\hat{y}_k^n - y^n)w_k^{n+1}\mathbf{f}_k\big((x_j^n)_{j\in\mathcal{A}_k}\big)$
$\mathbf{b}_k^{n+1} = \mathbf{b}_k^n(1-\frac{1}{n}) - \frac{1}{\lambda_k n}\nabla_k^n$
Project $\mathbf{b}_k^{n+1}$ to the ℓ_2 ball $\| \mathbf{b}_k \|_2 \leq \lambda_k$

may introduce unnecessary information exchange and possible over-fitting problems especially when some local learners are loosely correlated and need to be trained independently with each other. Neither of these properties are ideal for the distributed data mining, which requires from the learners not only good predictive accuracy but also the capability to adapt quickly to time-varying data dynamics with moderate information exchange. This motivates our proposal of the next training algorithm over the local learners.

By checking the covariance matrix $\mathbf{C}^T\mathbf{C}$, we propose Algorithm 2 with correlated update among local learners to trade the prediction accuracy for less information exchange and faster convergence speed. The basic idea of this algorithm is to analyze the matrix $\mathbf{C}^T\mathbf{C}$. When $\frac{|[\mathbf{C}^T\mathbf{C}]_{k_1k_2}|}{\sqrt{|[\mathbf{C}^T\mathbf{C}]_{k_1k_1}||[\mathbf{C}^T\mathbf{C}]_{k_2k_2}|}}$ is large, which indicates a high correlation (either positive or negative) between local learners k_1 and k_2, we decide to train them cooperatively. On the other hand, when this value is close to 0, local learners k_1 and k_2 are loosely correlated or even completely independent with each other. In this case, we train them independently with each other. Such an approach divides local learners into multiple correlated groups according to their correlations, with local learners within the same correlated group being trained cooperatively. This algorithm ensures the performance of the ensemble learner, while also having the potential to significantly reduce the information exchange and speed up the convergence by avoiding the over-fitting problem between loosely correlated local learners, especially when $\mathbf{C}^T\mathbf{C}$ is sparse. Detailed analysis of this algorithm can be found in the full version of the paper.

A key step in this algorithm is to determine the set of correlated local learners at the ensemble learner. Since the covariance matrix is stored at the ensemble learner, it can set up a threshold value δ_k for each local learner k, and determine that a local learner i is correlated with the local learner k if and only if $\frac{|[\mathbf{C}^T\mathbf{C}]_{ik}|}{|[\mathbf{C}^T\mathbf{C}]_{kk}|} > \delta_k$. In this paper, the threshold values $\{\delta_k\}$ in each period are set as follows:

$$\delta_k^1 = 1, \forall k \in \mathcal{K}; \delta_k^{n+1} = \frac{\delta_k^n}{|Cov_k| + 1} \tag{6}$$

There are several insights from the correlation threshold update (6). First, the threshold value starts at its maximum value 1 and hence, the local learners are trained independently at the beginning of the learning procedure in order to speed up the learning process. Second, the threshold value decreases exponentially over time and hence, $\lim_{n\to\infty}\delta_k^n = 0, \forall k \in \mathcal{K}$. This ensures that the truly correlated local learners will always be trained cooperatively in the long run. Third, the correlation threshold of a local learner who correlates with a larger group of other learners, i.e. whose Cov_k is larger, decreases faster.

3. EXPERIMENT RESULTS

Name	*Description*
NB	Naive Bayes
NBI	Incremental Naive Bayes
LBR	Lazy Bayesian Rules
MLP	Multilayer perceptron
BFT	A best-first decision tree

Table 1: Base Classifiers

Algorithm	*Precision*	*Recall*
Algorithm 2	85.4%	82.1%
AdaBoost	88.2%	84.3%
L-2 Boosting	72.2%	69.5%
Meta-L	73.1%	71.8%

Table 2: Performance of Different Algorithms

In this section, we provide preliminary experiment results to show the efficacy of the distributed algorithms proposed in Section II on the KDD Cup dataset from UCI KDD archive. This dataset contains about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic, which was processed into about five million connection records, among which we randomly select 50000 records as the training dataset. Each connection records is labeled either as a "normal" connection or as an attack.

The predictive model built up by our distributed algorithm classifies each connection with a label $y \in \{1, 0\}$, where 0 indicates a "normal" connection and 1 indicates an attack . Among the 50000 records used as the training dataset, 35.4% of them are attacks. Each record contains 40 features.

Each local learner maintains all or part of the base classifiers (probably with different parameter settings) as listed in Table 1. We use two metrics - precision and recall - to measure the performance of the distributed learning system.

In the first experiment, we utilize 10 local learners to examine the performance of the algorithms presented in Section II. In our experiment, we choose 10000 training data instances and 10000 test data instances. For comparison, we introduce three meta-learning algorithms as the benchmark, all of which use train the data instances in an off-line manner: the AdaBoost algorithm [15], the L-2 Boosting algorithm [8], the algorithm with local update (Meta-L) [14], respectively. L-2 Boosting simply takes the output of individual local learners as its input, and use an L_2 linear regression to refit a model. However, there is no feedback from the ensemble learner to local learners and hence, the model of each local learner is fixed during the training. In Meta-L, the ensemble learner simply combines the outputs of local learners in an additive form and does not adaptively adjust the weights it assigns to different local learners.

The results are shown in Table 2. It can be observed that our distributed learning algorithms achieves close performance as that of AdaBoost, while also significantly outperform L-2 Boosting and Meta-L. Hence, it is of paramount importance to update the model of the ensemble learner and those of the local learners cooperatively towards a direction that increases the overall prediction accuracy.

In the first experiment, the convergence speeds of our proposed algorithm was not assessed. In the next experiment, we examine the run-time behavior of Algorithm 2. The setting adopted for local learners is the same as the first experiment. Figure 1 shows the result. It is observed that with

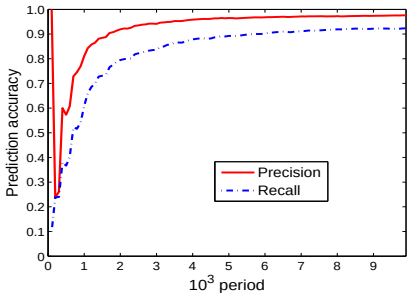

Figure 1: The evolution of the prediction accuracy.

the correct exploitation on the correlations between local learners, the convergence speed of our algorithm is fast.

4. RELATED WORKS

Several good surveys of various ensemble learning techniques can be found in [5], including voting [6], bagging [7], boosting [8], stacked generalization and cascading [9][10], etc. All these methods share the same principle: each local learner trains its own learning model independently, which requires no communication with other local learners at all. Hence, local learners are trained in a non-cooperative way which leaves unexplored the hidden rules that inform how different local learners are correlated and how they should be coordinated in training.

In contrast to such non-cooperative methods, cooperative distributed data mining techniques seek to improve prediction accuracy at the price of some mild communication costs between local learners. Examples of such methods are collective data mining [11][12] and distributed optimization techniques [13] that try to determine the information required to be shared among the local learners so that an optimal ensemble learner can be decomposed into an additive form without compromising its performance when compared to a learner that is trained by a centralized algorithm. However, collective data mining often requires extensive information exchanges by requiring local learners to share their learning models instead of training residuals, in order to be able to derive efficient decompositions.

The approach of cooperative learning with residual sharing is also discussed in [14]. However, there are two main reasons that prevent the algorithms in [14] to be applied to large-scale distributed mining applications. First, the algorithm in [14] requires each learner to update its learning model based on training residuals obtained from *all* the other learners. In applications where a large number of learners are in use, this requirement also imposes significant information exchange during the training process. Second, the algorithm in [14] is an off-line method, which is subject to the "curse of dimensionality" with its computation complexity growing exponentially as volumes of data increase, while our approach uses online learning techniques that are not subject to these constraints.

5. CONCLUSION

In this paper we proposed a general framework for designing online learning algorithms for large-scale distributed data mining applications. We focused on distributed linear regression problems on feature-distributed data across different local learners. The optimal regressors for the ensemble learner and the local learners were rigorously derived. We then designed two online algorithms that can converge to the optimal regressors: a cooperative update algorithm and a correlated update algorithm. We show that the correlated update algorithm significantly outperforms the cooperative update algorithm in terms of the required computational complexity and communication costs, with mild compromise on the prediction accuracy. The obtained results indicate that by smartly utilizing the correlation information among local learners, our proposed online learning algorithms can flexibly balance among computational complexity, communication cost and prediction accuracy based on the requirements of the application and the end users.

6. REFERENCES

[1] B. Park and H. Kargupta, *Distributed data mining: Algorithms, systems, and applications*, Distributed data mining handbook, pp. 341-358, 2002.

[2] M. Kantardzic, *Data mining: Concepts, models, methods, and algorithms*, Wiley-IEEE Press, 2011.

[3] P. Chan and S. Stolfo, *Experiments on multistrategy learning by meta-learning*, CIKM '93, 1993.

[4] S. Agrawal, V. Narasayya and B. Yang, *Integrating vertical and horizontal partitioning into automated physical database design*, SIGMOD '04, 2004.

[5] M. Sewell, *Emsemble learning*, UCL Research Note, 2007.

[6] S. McConnell and D. Skillicorn, *Building predictors from vertically distributed data*, CASCON '04, 2004.

[7] A. Lazarevic and V. Kumar, *Feature bagging for outlier detection*, KDD '05, 2005.

[8] P. Buhlmann and B. Yu, *Boosting with the L2 loss: Regression and classification*, J. American Stat. Assoc., 98(2003), pp. 324-339.

[9] C. Perlich and G. Swirszcz, *On cross-validation and stacking: building seemingly predictive models on random data*, ACM SIGKDD Explorations Newsletter, 12(2010), pp. 11-15.

[10] D. Wolpert, *Stacked generalization*, Neural Networks, 5(1992), pp. 241-259.

[11] H. Kargupta, B. Park, D. Hershberger and E. Johnson, *ACollective data mining: a ner perspective toward distributed data analysis*, Advances in Distributed and Parallel Know. Discovery, 1999.

[12] H. Kargupta, B. Park, D. Hershberger and E. Johnson, *Collective data mining from distributed, vertically partitional feature space*, KDD '98, 1998.

[13] G. Mateos, J. Bazerque and G. Giannakis, *Distributed sparse linear regression*, IEEE Trans. on Signal Process., 58 (2010), pp. 5262-5276.

[14] H. Zheng, S. Kulkarni and H. Poor, *Attribute distributed learning: models, limits, and algorithms*, IEEE Trans. on Signal Process., 59(2011), pp. 386-398.

[15] Y. Freund and R. Schapire, *A decition-theoretic generalization of on-line learning and an application to boosting*, Computational Learning Theory, 904(1995), pp. 23-37.

[16] S. Shalev-Shwartz, Y. Singer and N. Srebro, *Pegasos: Primal estimated sub-gradient solver for SVM*, ICML '07, 2007.