Multi-Party Sparse Discriminant Learning

Jiang Bian[1], Haoyi Xiong[1,*], Wei Cheng[2], Wenqing Hu[1], Zhishan Guo[1] and Yanjie Fu[1]
[1]Missouri University Science and Technology, USA, [2]NEC Laboratories America, USA

***Abstract*—Sparse Discriminant Analysis (SDA) has been widely used to improve the performance of classical Fisher's Linear Discriminant Analysis in supervised metric learning, feature selection and classification. With the increasing needs of distributed data collection, storage and processing, enabling the Sparse Discriminant Learning to embrace the Multi-Party distributed computing environments becomes an emerging research topic. This paper proposes a novel Multi-Party SDA algorithm, which can learn SDA models effectively without sharing any raw data and basic statistics among machines. The proposed algorithm 1) leverages the direct estimation of SDA [1] to derive a distributed loss function for the discriminant learning, 2) parameterizes the distributed loss function with local/global estimates through bootstrapping, and 3) approximates a global estimation of linear discriminant projection vector by optimizing the "distributed bootstrapping loss function" with gossip-based stochastic gradient descent. Experimental results on both synthetic and real-world benchmark datasets show that our algorithm can compete with the centralized SDA with similar performance, and significantly outperforms the most recent distributed SDA [2] in terms of accuracy and F1-score.**

I. Introduction

The Fisher's Linear Discriminant Analysis (LDA) [3] is widely used in supervised learning and feature extraction. Given a set of training data, LDA can find the optimal discriminant projection that can project the high-dimensional data points to low dimensional space, and achieve optimal classification performances by minimizing the overlaps between difference classes in the low-dimensional space. To further improve the performances of LDA, Sparse Discriminant Analysis (SDA) [1] has been proposed to "recover" discriminant projection with sparsity pursuit. While a wide range of methods [1], [4]–[6] have been proposed, Cai et al. [1] studied a direct estimator that can estimate Sparse Discriminant Analysis (SDA) straightforwardly from labeled data with provable guarantee in asymptotic property and classification accuracy.

On the other hand, with the increasing needs of distributed data collection, storage and processing, Multi-Party computing [7] becomes an emerging computing paradigm that enables big data applications in a privacy-preserved manner. In a multi-party computing platform with no "raw data sharing" allowed, a machine learning model should be trained using all data stored in distributed machines (i.e., parties) without any cross-machine raw data sharing. Generally speaking, such multi-party distributed machine learning algorithms can be divided into two categories – *data-centric* and *model-centric* methods [8]–[12]. On each machine, the data centric algorithm first estimates the same set of parameters (of the model) using the local data, then aggregates the estimated parameters via model-averaging for global estimation. The model with aggregated parameters is considered as the trained model based on the overall data (from multiple parties). Meanwhile, model-centric algorithms require multiple machines to share the same loss function with "updatable parameters", and allow each machine to update the parameters in the loss function using the local data so as to minimize the loss. Compared with the data-centric, the model-centric methods usually can achieve better performances, as it minimizes the risk of the model [8], [12]. To advance the distributed performance of classical SDA, recently, Tian and Gu et al. [2] proposed a data-centric SDA algorithm. However, in literature, few researches have been carried out on the model-centric counterpart for SDA, which intuitively may receive better performance.

To fill the gap, we are motivated to propose a novel model-centric SDA learning algorithm for multi-party discriminant learning. In this paper, we propose Multi-Party SDA (namely *MPSDA*) that enables the direct estimation of SDA [1] to embrace the multi-party computing environment for sparse discriminant learning. Specifically, *MPSDA* first allocates the mean and covariance matrix estimation tasks to each machine and allows each machine to estimate its local mean vectors and covariance matrices based on the local data. Then, *MPSDA* estimates the global mean over all the data using the local means via the gossip-based stochastic gradient descent. Further, *MPSDA* proposes a distributed bootstrapping loss function and model the loss function using the global mean and local covariance matrices. Finally, a gossip-based stochastic gradient descent algorithm is employed to minimize the distributed bootstrapping loss function and estimate the global discriminant projection vector. Compared with the approach in [1], which aggregates all data on a single machine to learn the model, *MPSDA* can effectively approximate to the optimal solution without sharing any raw data. Compared with [2], which aggregates the locally learned models through model-averaging and hard-thresholding, *MPSDA* models and minimizes a distributed loss function based on SDA, parameterized with global/local estimates, straightforwardly.

The contributions of the proposed *MPSDA* algorithm are as follows:

- We study and formulate the problem of sparse discriminant learning on the top of multi-party computing platform, while assuming the raw data distributed on machines (parties) are not sharable. To the best of our knowledge, this is the first study on sparse discriminant analysis, by addressing 1) multi-party computing platform without sharing raw data, 2) model-centric learning with shared loss function, and 3) distributed optimization issues. Note that ***Multi-Party computing*** settings [13], ***usually***

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.86

leverage the secured communication and computation to keep the local data, on each party, private, while our work assumes the local raw data and basic statistics (on each machine) are not accessible by others. Thus we do not make further assumption on cryptography issue.

- To achieve the above goals, we design the *MPSDA* algorithm which leverages the direct estimation of SDA to derive a distributed loss function of the discriminant learning, parameterizes the distributed loss function with local/global estimates through bootstrapping, and approximates a global estimation of linear discriminant projection vector by optimizing the "distributed bootstrapping loss function".
- We evaluate *MPSDA* on both pseudo-random simulation and real-world benchmark datasets for binary classification. The results show that *MPSDA* can compete with the centralized SDA with similar performances, and significantly outperform the state-of-the-art distributed SDA [2].

The rest of the paper is organized as follows. In Section II, we review Fisher's Linear Discriminant Analysis and present the details of *MPSDA* algorithm. In Section III, we evaluate the proposed algorithms on synthetic datasets and benchmark datasets. In addition, we compare *MPSDA* with baseline centralized algorithms. Finally, we conclude the paper in Section IV.

II. Models and Proposed Algorithms

In this section, we present the algorithm design of *MPSDA*, where we first review the model of Fisher's Linear Discriminant Analysis, then we cover the key algorithms used in *MPSDA*.

A. Model Overview: Fisher's Linear Discriminant Analysis

Linear Discriminant Analysis (LDA), which leverages a linear combination of features that characterize or separate two or more classes of objects or events. LDA has been shown to perform well and enjoy certain optimality as the sample size tends to infinity while the dimension is fixed [1]. Given the LDA classifier $\psi_F(Z)$ based on the given p-dimensional data vector Z that is drawn from one of two distributions $\mathcal{N}(\mu_+, \Sigma)$ and $\mathcal{N}(\mu_-, \Sigma)$ with equal prior probabilities, the binary classification problem can be solved by

$$\psi_F(Z) = sign\Big\{(Z-\mu)^T\Theta(\mu_+ - \mu_-)\Big\}, \tag{1}$$

where $\mu = (\mu_+ + \mu_-)/2$; $\Theta = \Sigma^{-1}$ is the inverse covariance matrix; μ_+ and μ_- are the mean vectors of the positive samples and negative samples respectively; $\psi_F(Z)$ classifies Z into positive class if and only if $\psi_F(Z) = 1$. In practice, μ_+, μ_- and Θ are unknown, we need to estimate μ_+, μ_- and Θ from observations. Specifically, we assume the data Z is randomly drawn from $\mathcal{N}(\mu_+, \Sigma)$ and $\mathcal{N}(\mu_-, \Sigma)$ with equal priors.

A simple way to estimate μ_+, μ_- and Θ is to use their sample estimator: $\bar{\mu}_+$, $\bar{\mu}_-$, $\bar{\Theta} = \bar{\Sigma}^{-1}$, where $\bar{\Sigma}$ is pooled sample covariance matrix estimation [14] with respect to the two classes. Note that, under the High Dimensional Low Sample Size (HDLSS) settings, $\bar{\Sigma}$ is often singular [15] and $\bar{\Sigma}^{-1}$ usually does not exist [16]. Thus, to train LDA, researchers [1], [5] proposed to estimate the linear discriminate projection vector $\beta = \Theta(\mu_+ - \mu_-)$, instead of estimating Θ and $\mu_+ - \mu_-$ separately.

B. The Three-Stage Algorithm for MPSDA Training

To facilitate $\Theta(\mu_+ - \mu_-)$ estimation under the ***Multi-Party computing*** settings which have been defined previously, *MPSDA* first estimates the projection vector $\widehat{\beta}^*$ that approximates $\Theta(\mu_+ - \mu_-)$ over Gossip-based stochastic gradient descent, so as to avoid the singularity issue of the sample covariance matrix. Then, for any new data vector Z arriving at any machine, *MPSDA* outputs the classification result (i.e., ± 1) as the computing result of $sign\left((Z-\widehat{\mu})^T\widehat{\beta}^*\right)$.

1) Stage I – Global Mean Estimation: Given the local training samples T_+^j and T_-^j on each machine j, *MPSDA* first estimates the local mean vectors $\bar{\mu}^j$, $\bar{\mu}_+^j$ and $\bar{\mu}_-^j$. Then, *MPSDA* randomly picks up a starting machine (e.g., the j^{th} machine) and sends $(\mathbf{0}, \mathbf{0}, \mathbf{0}, 1)$ to the Algorithm. 1 on the machine, where $\mathbf{0}$ refers to a p-dimensional vector with all zero elements and 1 refers to the first update of the algorithm. Algorithm. 1 is a gossip-based stochastic gradient decent algorithm that intends to approximate the global means using the estimators listed in Eq. 2.

$$\widehat{\mu} = \underset{\mu \in \Re^{1\times p}}{\operatorname{argmin}} \frac{1}{m}\sum_{j=1}^{m} |\mu - \mu^j|_\infty, \tag{2}$$

so as to $\widehat{\mu}_+$, $\widehat{\mu}_-$. Specifically, the Algorithm. 1 first receives the input mean vectors (initialed as $\mathbf{0}$ in the first run), then it updates the input mean vectors using the local means, and randomly picks up the next machine and sends the updated mean vectors for further updating. Algorithm. 1 keeps picking up the next machine for the updating, until (1) the total number of updates t exceeds the maximal number of updates, or (2) the updating process converges (i.e., $max\left\{\left|\widehat{\mu} - \bar{\mu}^j\right|_\infty, \left|\widehat{\mu}_+ - \bar{\mu}_+^j\right|_\infty, \left|\widehat{\mu}_- - \bar{\mu}_-^j\right|_\infty\right\} \leq \Delta_{max}$). Once the updating process completes, Algorithm. 1 broadcasts all m machines with the final global mean estimations $\widehat{\mu}$, $\widehat{\mu}_+$ and $\widehat{\mu}_-$ for Algorithm. 2 computation. Note that the notation $\nabla|\widehat{\mu} - \bar{\mu}^j|_\infty$ refers to the gradient of function $|\widehat{\mu} - \bar{\mu}^j|_\infty$ over $\widehat{\mu}$ and can be implemented as:

$$\left(\nabla|\widehat{\mu} - \bar{\mu}^j|_\infty\right)_k = \begin{cases} sign((\widehat{\mu} - \bar{\mu}^j)_k), & \text{if } |(\widehat{\mu} - \bar{\mu}^j)_k| \\ & \text{is the maximal for } 1 \leq k \leq p \\ 0, \quad else \end{cases} \tag{3}$$

where $(\cdot)_k$ refers to the k^{th} element in the input vector.

2) Stage II – Local Covariance Matrix Estimation: Given the global mean vectors $\widehat{\mu}_+$ and $\widehat{\mu}_-$, *MPSDA* runs Algorithm. 2 in parallel on each machine without any inter-machine communication requirement. Specifically, this stage first estimates the sample covariance matrix $\bar{\Sigma}^j$ using the global mean vectors. Then, to handle the High-Dimensional Low Sample Size settings, the algorithm leverages the de-sparsified Graphical Lasso estimator [17] ($\widehat{D}^j$) to improve the estimation of inverse

Algorithm 1: Global Mean Vectors Estimation Algorithm on j^{th} machine

Data:
$\bar{\mu}^j$, $\bar{\mu}_+^j$, and $\bar{\mu}_-^j$ — the local mean vectors based on training samples on j^{th} machine
Parameter:
η — step size
Δ_{max} — maximumly allowed perturbation
t_{max} — maximum number of allowed updates
begin
/* On receiving the message from the previous machine */
RECEIVE ($\widehat{\mu}$, $\widehat{\mu}_+$, $\widehat{\mu}_-$, t)
/* Updating mean vectors on the j^{th} machine */
$\widehat{\mu} \leftarrow \widehat{\mu} - \eta \cdot \nabla|\widehat{\mu} - \bar{\mu}^j|_\infty$
$\widehat{\mu}_+ \leftarrow \widehat{\mu}_+ - \eta \cdot \nabla|\widehat{\mu}_+ - \bar{\mu}_+^j|_\infty$
$\widehat{\mu}_- \leftarrow \widehat{\mu}_- - \eta \cdot \nabla|\widehat{\mu}_- - \bar{\mu}_-^j|_\infty$
$t \leftarrow t + 1$
/* Checking convergence conditions */
$\Delta = max\left\{\left|\widehat{\mu} - \bar{\mu}^j\right|_\infty, \left|\widehat{\mu}_+ - \bar{\mu}_+^j\right|_\infty, \left|\widehat{\mu}_- - \bar{\mu}_-^j\right|_\infty\right\}$
if $\Delta \geq \Delta_{max}$ ***AND*** $t \leq t_{max}$ **then**
/* Not converged, continuing the algorithm */
$j_{next} \leftarrow$ **Draw a random number from** 1 **to** m;
SEND $(\widehat{\mu}, \widehat{\mu}_+, \widehat{\mu}_-, t)$ **to the** j_{next}^{th} **machine**;
else
/* Converged, sharing the estimates to all machines */
BROADCAST $(\widehat{\mu}, \widehat{\mu}_+, \widehat{\mu}_-)$ to **All machines**;
end
end

Algorithm 2: Local Covariance Matrix Estimation (with Global Mean) on the j^{th} Machine

Data:
T^j — training sample on $j = 1, 2, ..., m$ machine
Input:
$\widehat{\mu}_+$ — global positive mean vector
$\widehat{\mu}_-$ — global negative mean vector
λ_{glasso} — Graphical Lasso regularization parameter
Output:
$\bar{\Sigma}_p^j$ — the local covariance matrix (with global mean) on the j^{th} machine
begin
/* Sample covariance matrix estimation */
$\bar{\Sigma}_+^j = (T_+^j - \widehat{\mu}_+)(T_+^j - \widehat{\mu}_+)^T$
$\bar{\Sigma}_-^j = (T_-^j - \widehat{\mu}_-)(T_-^j - \widehat{\mu}_-)^T$
$\bar{\Sigma}^j = \frac{1}{2}(\bar{\Sigma}_+^j + \bar{\Sigma}_-^j)$
/* Precision matrix estimation through Graphical Lasso [18] */
$\widehat{\Theta}^j \leftarrow glasso(\Sigma^j)$
/* De-sparsify precision matrix */
$\widehat{D}^j \leftarrow \left(2\widehat{\Theta}^j - \widehat{\Theta}^j\bar{\Sigma}^j\widehat{\Theta}^j\right)$
/* Obtain the de-sparsified covariance matrix */
$\widehat{\Sigma}^j \leftarrow (\widehat{D}^j)^{-1}$
/* Continuing on next machine */
end

covariance matrix. Finally, matrix inverse is used to estimate the covariance matrix $\widehat{\Sigma}^j$ on the j^{th} machine.

3) Stage III – Sparse Discriminant Projection Vector Estimation: Given the local covariance matrix $\widehat{\Sigma}$ on each machine j and the global mean vectors $\widehat{\mu}_+$, $\widehat{\mu}_-$, this stage intends to approximate the global estimation of $\widehat{\beta}^*$ via gossip-based stochastic gradient decent. Specifically, *MPSDA* randomly picks up a starting machine (e.g., the j^{th} machine) and sends $(\mathbf{0}, 1)$ to Algorithm. 3 on the machine, where $\mathbf{0}$ refers to a p-dimensional vector with all zero elements and 1 refers to the first update of the algorithm. Algorithm. 3 indeed minimizes the following loss function over the m machines through gossip-based stochastic gradient decent:

$$\widehat{\beta}^* \leftarrow \underset{\beta \in \mathbb{R}^p}{argmin}\ \lambda \cdot |\beta|_1 + \frac{1}{m}\sum_{j=1}^{m}\left|\widehat{\Sigma}^j\beta - (\widehat{\mu}_+ - \widehat{\mu}_-)\right|_\infty, \quad (4)$$

where λ is a regularization parameter. Specifically, Algorithm. 2 first receives the input $\widehat{\beta}^*$ for updating (initialed as $\mathbf{0}$ in the first run), then it updates the inputed $\widehat{\beta}^*$ vector using $\widehat{\Sigma}^j$ and $\widehat{\mu}_+/\widehat{\mu}_-$, and randomly picks up the next machine and sends the updated $\widehat{\beta}^*$ for further updating. Algorithm. 2 keeps picking up the next machine for the updating, until (1) the times of updates t exceeds the maximal number of updates, or (2) the updating process converges. Once the updating process completes, Algorithm. 3 broadcasts all m machines with the final global estimation of $\widehat{\beta}^*$. To this end, each machine already receives $\widehat{\beta}^*$ and $\widehat{\mu}$ as the trained SDA model.

Note that in Algorithm. 1, we set the parameters as following: maximum number of allowed updates is $t_{max} = 10^6$, step size is $\eta = 10^{-2}$ and maximum allowed perturbation is $\Delta_{min} = 10^{-8}$. In Algorithm. 3, the parameter settings are $t_{max} = 10^4$, $\eta = 10^{-6}$ and $\Delta_{min} = 10^{-4}$.

C. Remark on the Algorithm

Suppose the size of training set on each machine n is sufficiently large and all these samples are drawn i.i.d. from Gaussian distributions $\mathcal{N}(\mu_+, \Sigma)$ and $\mathcal{N}(\mu_-, \Sigma)$. We can assume that the local sample covariance matrix $\bar{\Sigma}^j$ estimated from local raw data on each (the j^{th}) machine should follow an inverse wishart distribution $\mathcal{W}^{-1}(\Sigma, v(n))$, where $v(n)$ is a function on n for the degree of freedom. With infinite number of machines $m \to \infty$ and infinite number of gossip message passing (i.e., $t \to \infty$), the algorithm can converge to the minimum of $\widehat{R}(\beta)$ (as the loss function $\widehat{R}$ is convex [1]), where

$$\widehat{R}(\beta) = \mathbb{E}_{\widehat{\Sigma} \sim \mathcal{W}^{-1}(\Sigma, v(n))}\left(\lambda \cdot |\beta|_1 + \left|\widehat{\Sigma}\beta - (\widehat{\mu}_+ - \widehat{\mu}_-)\right|_\infty\right). \quad (5)$$

According to the definition of Bayes estimator [19] and KKT conditions, $\widehat{R}(\beta)$ indeed approximates the following loss function,

$$\underset{\beta \in \Re^p}{\text{argmin}}\left\{|\beta|_1 \quad s.t. |\Sigma\beta - (\widehat{\mu}_+ - \widehat{\mu}_-)|_\infty \leq \varepsilon\right\}, \quad (6)$$

through sampling the Wishart Distribution, where Σ refers to the (unknown) true covariance matrix and $\widehat{\mu}_+$, $\widehat{\mu}_-$ refer to the estimated mean vectors. Note that the solution Eq. 6 is actually the Direct Estimator of Sparse LDA proposed by Cat et al. [1].

III. Experiment

In this section, we use both synthetic data and real-world data to evaluate the performance of *MPSDA* algorithm. Specifically,

Algorithm 3: $\widehat{\beta}^*$ Estimation on the j^{th} machine

Data:
$\widehat{\Sigma}^j$ — the local covariance matrix on the j^{th} machine
Parameter:
η — step size
Δ_{min} — minimum allowed perturbation
t_{max} — maximum number of allowed updates
λ — regularization parameter
begin
  /* On receiving the message from the previous machine */
  RECEIVE $(\widehat{\beta}^*, t)$
  /* Selecting the k^{th} row of vector $(\widehat{\Sigma}^j\widehat{\beta}^* - (\widehat{\mu}_+ - \widehat{\mu}_-))$ with the maximal absolute value */
  $k \leftarrow \underset{1\le k'\le p}{argmax} \left|\left(\widehat{\Sigma}^j\widehat{\beta}^* - (\widehat{\mu}_+ - \widehat{\mu}_-)\right)_{k'}\right|$
  /* Updating each row of $\widehat{\beta}^*$ on the j^{th} machine */
  $\beta' \leftarrow \langle 0, 0, \ldots, 0\rangle^T$
  /* initializing β with a p-dimensional 0 vector */
  foreach $1 \le l \le p$ **do**
    /* Note: $\widehat{\Sigma}^{j,k,l}$ is the scaler on the k^{th} row and the l^{th} column of the matrix $\widehat{\Sigma}^j$ */
    $g_l \leftarrow sign(\widehat{\beta}_l^*)\cdot\lambda + sign(\widehat{\Sigma}^{j,k}\widehat{\beta}^* - (\widehat{\mu}_+ - \widehat{\mu}_-))\cdot\widehat{\Sigma}^{j,k,l}$
    /* Update each row of local β based on $\widehat{\beta}^*$ */
    $\beta'_l \leftarrow \widehat{\beta}_l^* - \eta\cdot g_l$;
  end
  $t \leftarrow t+1$
  /* Checking convergence conditions */
  $\Delta = \left|\widehat{\beta}^* - \beta'\right|_1$
  /* Update $\widehat{\beta}^*$ after calculating the Δ */
  $\widehat{\beta}^* \leftarrow \beta'$
  if $\Delta \ge \Delta_{max}$ ***AND*** $t \le t_{max}$ **then**
    /* Not converged, continuing the algorithm */
    $j_{next} \leftarrow$ **Draw a random number from** 1 **to** m;
    SEND $(\widehat{\beta}^*, t)$ **to the** j_{next}^{th} **machine**;
  else
    /* Converged, sharing the estimates to all machines */
    BROADCAST $\widehat{\beta}^*$ to **All machines**;
  end
end

we compare our algorithm with distributed SDA algorithm and centralized SDA algorithm. For centralized SDA, all samples are collected on one machine based on the algorithm proposed by [1]. For distributed SDA, we adopt the algorithm proposed by [2] which estimate the global estimator by aggregating local unbiased estimators through averaging with hard threshold.

A. Synthetic Data Experiments

Experiment Setup: To validate our algorithm, we evaluate our algorithm on a synthesized dataset, which are obtained through a pseudo-random simulation. The synthetic data are generated by two predefined Gaussian distributions $\mathcal{N}(\mu_+^*, \Sigma^*)$ and $\mathcal{N}(\mu_-^*, \Sigma^*)$ with equal priors. The settings of μ_+^*, μ_-^* and Σ^* are as follows: Σ^* is a $p\times p$ symmetric and positive-definite matrix, where $p = 200$, each element $\Sigma_{i,j}^* = 0.8^{|i-j|}$, $1\le i\le p$ and $1\le j\le p$. μ_+^* and μ_-^* are both p-dimensional vectors, where $\mu_+^* = \langle 1, 1, \ldots, 1, 0, 0, \ldots, 0\rangle^T$ (the first 10 elements are all 1's, while the rest $p-10$ elements are 0's) and $\mu_-^* = \mathbf{0}$. While noting that the number of samples from two Gaussian distributions are equal on each machine.(Settings of the two Gaussian distributions first appear in [2].) In order to evaluate the performance of algorithms for comparison, we obtain the accuracy, F1-score, ROC curve and AUC from the classification results. Specifically, accuracy and F1-score are calculated by maximizing the accuracy/F1-score across all possible cutoffs in ROC curve and AUC stands for the area under the ROC curve. Usually, a higher AUC means the model has a better fit on the datasets.

We report the best results based on fine-tuned parameters for all methods. Also we fix the testing samples at 400 for all the following experiments.

Setting 1 – Fix the total training sample size and vary the number of machines: To investigate the effect of number of machines m, we fix the total training sample size $N = 20000$ and vary the number of machines. Figure 1 shows how the accuracy, F1-score and AUC of *MPSDA*, centralized SDA and distributed SDA change as the number of machine grows. For each m, we repeat each algorithm for 10 times and report the average value. From Figure 1, we can find that *MPSDA* algorithm outperforms distributed SDA algorithm on accuracy, F1-score and AUC. It is unsurprising that centralized SDA outperforms both *MPSDA* and distributed SDA on accuracy, F1-score and AUC.

Setting 2 – Fix the training sample size on each machine and vary the number of machines: We alter the settings to evaluate the effect of averaging. We increase the number of machines m linearly as the total training sample size N, that is, the sample size on each machine n is fixed. More specifically, we choose $n = 400$. Fig. 2 displays the accuracy, F1-score and AUC of the three algorithms. The result shows that the performance of *MPSDA* still outperforms distributed SDA algorithm on accuracy, F1-score and AUC. Similarly, centralized SDA outperforms both *MPSDA* and distributed SDA algorithm. We notice that the performance of *MPSDA* is close to the performance (accuracy, F1-score and AUC) of centralized SDA when the number of machines is equal to or less than 20. The same situation occurs when the number of machines is equal to or greater than 100.

Summary: In synthetic data experiments, we compare the performance of *MPSDA* with distributed SDA and centralized SDA in two settings. At most circumstance, centralized SDA has the best performance compared to other two algorithms. Typically, the performance of *MPSDA* can approach the performance of centralized SDA in Setting 2 with the sample size on each machine increased ($\geqslant 100$) or stayed relatively low ($\leqslant 20$). Note that, in both settings, *MPSDA* outperforms distributed SDA significantly. The *Receiver Operating Characteristic* (ROC) of both settings are also attached in Fig. 1 and Fig. 2, respectively, which further validate our findings.

B. Benchmark Data Experiments

Experiment Setup: To verify the effectiveness of *MPSDA* algorithm on real datasets, we use Phishing datasets [20] to conduct the comparison. Specifically, we set the size of total training samples varied from 200 to 2000 with 400 testing

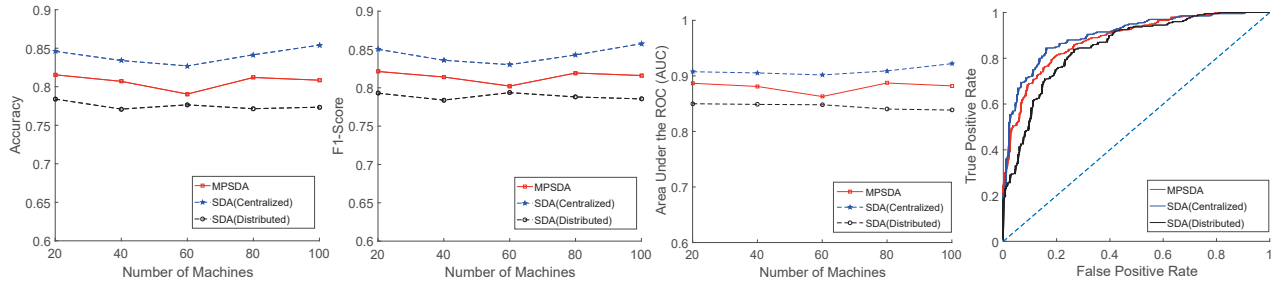


Fig. 1: Performance Comparison among *MPSDA*, SDA (Centralized) and SDA (Distributed) on synthetic datasets. We compare the Accuracy, F1-Score, AUC and ROC curve of each algorithm **when the total training sample size is fixed as 20000**. (Note that the ROC curve is drawn when the number of machines is 100)

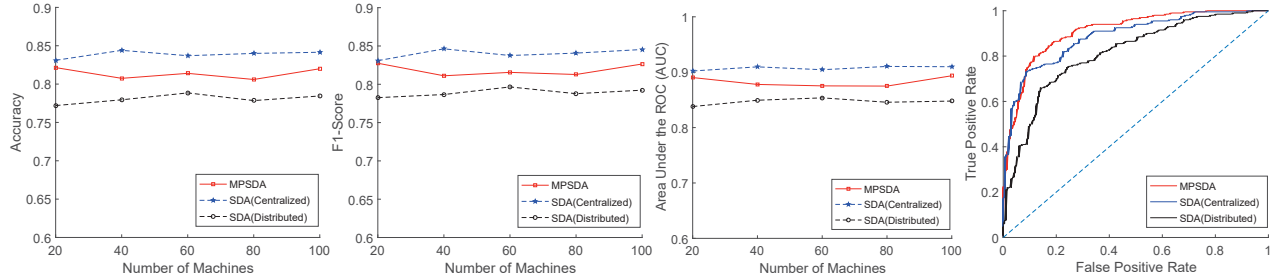


Fig. 2: Performance Comparison among *MPSDA*, SDA (Centralized) and SDA (Distributed) on synthetic datasets. We compare the Accuracy, F1-Score, AUC and ROC curve of each algorithm **when the training sample size on each machine is set as 400**. (Note that the ROC curve is drawn when the number of machines is 100)

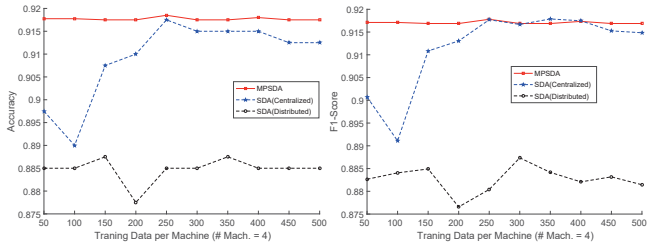
TABLE I: Accuracy Comparison among *MPSDA*, SDA (Centralized) and SDA (Distributed) on **Phishing Datasets**. Noting that, *MPSDA*' values are **bold** since it outperforms SDA (distributed).

Algorithm	Total Training Set Size									
	200	400	600	800	1000	1200	1400	1600	1800	2000
Distributed Algorithm ($number\ of\ machines,\ m = 4$)										
MPSDA	**0.918±0.001**	**0.918±0.001**	**0.918±0.000**	**0.918±0.000**	**0.919±0.002**	**0.918±0.000**	**0.918±0.000**	**0.918±0.002**	**0.918±0.000**	**0.918±0.000**
SDA (Distributed)	0.885±0.000	0.885±0.000	0.888±0.000	0.878±0.000	0.885±0.000	0.885±0.000	0.888±0.000	0.885±0.000	0.885±0.000	0.885±0.000
Centralized Algorithm										
SDA (Centralized)	0.898±0.000	0.890±0.000	0.908±0.000	0.910±0.000	0.918±0.000	0.915±0.000	0.915±0.000	0.915±0.000	0.913±0.000	0.913±0.000
Ye-LDA	0.932±0.024	0.949±0.017	0.947±0.020	0.954±0.016	0.954±0.018	0.948±0.019	0.951±0.015	0.945±0.020	0.953±0.016	0.950±0.017
Linear SVM	0.984±0.010	0.998±0.002	0.998±0.002	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001
Kernel SVM	0.969±0.025	0.995±0.004	0.996±0.004	0.998±0.002	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001
Random Forest	0.947±0.027	0.962±0.017	0.984±0.012	0.962±0.020	0.991±0.007	0.987±0.008	0.985±0.007	0.960±0.018	0.993±0.005	0.995±0.004
Decision Tree	0.981±0.016	0.994±0.006	0.998±0.002	0.997±0.003	0.997±0.003	0.999±0.001	0.998±0.002	0.998±0.002	0.998±0.002	0.999±0.001

samples, while the numbers of dimensions p are $p = 54$ respectively. The number of machines is fixed at 4. We repeat each algorithm for 10 times and report the average value.

In this experiment, we compare the classification accuracy and F1-score of *MPSDA* with distributed SDA and centralized SDA on each benchmark datasets. Figure. 3(a)(b) presents the performance of each algorithms on Phishing datasets. We can observe that *MPSDA* obviously outperforms distributed SDA and centralized SDA when the training sample size is smaller than 250, even when the training sample size is greater than 250, *MPSDA* is still comparable to centralized SDA and obviously superior to distributed SDA.

Further, we compare *MPSDA* algorithm with other centralized baseline algorithms in the same setting. For comparison, we categorize *MPSDA* and the baseline algorithms into groups of distributed algorithms and centralized algorithms. The distributed algorithms include *MPSDA* and distributed SDA. The centralized algorithms include centralized SDA, centralized two-stage LDA (Ye-LDA), centralized Linear SVM, centralized Kernel SVM, centralized Random Forest and



(a) Accuracy on Phishing Datasets (400 Testing Samples)

(b) F1-Score on Phishing Datasets (400 Testing Samples)

Fig. 3: Performance Comparison among *MPSDA*, SDA (Centralized) and SDA (Distributed) with Phishing Datasets (Machine Number $m = 4$ with 400 Testing samples)

centralized Decision Tree. All the algorithms are fine-tuned. Table. I presents the accuracy with standard deviation of each algorithm in varying total training sample size. We notice that for two groups, the centralized algorithms have overall better performance compared to distributed algorithms. For comparison in distributed group, *MPSDA* significantly outperforms distributed SDA on Phishing datasets.

Summary: In benchmark data experiments, we first compare the performance of *MPSDA* with distributed SDA and centralized SDA on real-world benchmark datasets. In most instances, *MPSDA* can compete with centralized SDA, even outperform centralized SDA on Mushrooms and Phishing datasets. Like the results on synthetic datasets, *MPSDA* overall outperforms distributed SDA on three benchmark datasets. Then, we additionally compare *MPSDA* with other centralized baseline algorithms. The result shows that these well-tuned centralized baseline algorithms dominantly outperform *MPSDA* and distributed SDA. While in distributed algorithm group, *MPSDA* still outperforms distributed SDA. Also we compare the time consumption of *MPSDA* algorithm (1.13×10^3 seconds) with centralized SDA algorithm (3.97 seconds) on Mushrooms datasets (4 machines with 2000 total training samples). Note that the communication time between each machine account for large proportion in the total time consumption of *MPSDA*. Actually, on each machine, *MPSDA* only takes 0.93 seconds which is much less than the centralized SDA algorithm. (The experiment platform is Windows OS with 2.8GHz CPU)

IV. Conclusion

In this paper, we proposed *MPSDA* - a set of novel Multi-Party SDA algorithms. Specifically, *MPSDA* is designed to enable sparse discriminant learning effectively without sharing any raw data and basic statistics (means and covariance matrix estimated using the data on specific machine) between machines. *MPSDA* proposed three-stage training procedure for SDA estimation on top of Multi-Party computing platform, where gossip-based stochastic gradient descent algorithms are used to minimize a bootstrapping loss function derived from Cai et al. [1]. Note that, during the optimization procedures, only the gradients of loss function are exchanged between machines in a gossip manner and no raw data or basic statistics are shared directly. The experimental results on synthetic datasets and real-world benchmark datasets for classification show that that *MPSDA* is comparable to the centralized SDA with similar performance. Furthermore, *MPSDA* significantly outperforms state-of-the-art distributed SDA algorithm based on model average in most cases.

References

[1] Tony Cai and Weidong Liu. A direct estimation approach to sparse linear discriminant analysis. *Journal of the American Statistical Association*, 106(496):1566–1577, 2011.

[2] Lu Tian and Quanquan Gu. Communication-efficient distributed sparse linear discriminant analysis. *arXiv preprint arXiv:1610.04798*, 2016.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Ed)*. Wiley, 2001.

[4] Line Clemmensen, Trevor Hastie, Daniela Witten, and Bjarne Ersbøll. Sparse discriminant analysis. *Technometrics*, 53(4), 2011.

[5] Jun Shao, Yazhen Wang, Xinwei Deng, Sijian Wang, et al. Sparse linear discriminant analysis by thresholding for high dimensional data. *The Annals of Statistics*, 39(2):1241–1265, 2011.

[6] Daniela M Witten and Robert Tibshirani. Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):615–636, 2009.

[7] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *Annual International Cryptology Conference*, pages 521–536. Springer, 2006.

[8] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

[9] Lingfei Wu, Kesheng John Wu, Alex Sim, Michael Churchill, Jong Y Choi, Andreas Stathopoulos, Choong-Seock Chang, and Scott Klasky. Towards real-time detection and tracking of spatio-temporal features: Blob-filaments in fusion plasma. *IEEE Transactions on Big Data*, 2(3):262–275, 2016.

[10] Jie Chen, Lingfei Wu, Kartik Audhkhasi, Brian Kingsbury, and Bhuvana Ramabhadrari. Efficient one-vs-one kernel ridge regression for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2454–2458. IEEE, 2016.

[11] Lingfei Wu, Ian EH Yen, Jie Chen, and Rui Yan. Revisiting random binning features: Fast convergence and strong parallelizability. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1265–1274. ACM, 2016.

[12] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[13] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *International Journal of Information Security*, 11(6):403–418, 2012.

[14] M Hashem Pesaran, Yongcheol Shin, and Ron P Smith. Pooled mean group estimation of dynamic heterogeneous panels. *Journal of the American Statistical Association*, 94(446):621–634, 1999.

[15] Finbarr O'Sullivan. A statistical perspective on ill-posed inverse problems. *Statistical science*, pages 502–518, 1986.

[16] Sarunas Raudys and Robert PW Duin. Expected classification error of the fisher linear classifier with pseudo-inverse covariance matrix. *Pattern Recognition Letters*, 19(5):385–392, 1998.

[17] Jana Jankova, Sara van de Geer, et al. Confidence intervals for high-dimensional inverse covariance estimation. *Electronic Journal of Statistics*, 9(1):1205–1229, 2015.

[18] Daniela M Witten, Jerome H Friedman, and Noah Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

[19] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[20] John C Platt. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208, 1999.