Learning Privately over Distributed Features: An ADMM Sharing Approach

Yaochen Hu
University of Alberta
yaochen@ualberta.ca

Peng Liu
University of Kent
P.Liu@kent.ac.uk

Keshi Ge
University of Alberta
gekeshi@nudt.edu.cn

Linglong Kong
University of Alberta
lkong@ualberta.ca

Bei Jiang
University of Alberta
bei1@ualberta.ca

Di Niu
University of Alberta
dniu@ualberta.ca

Abstract

Distributed and federated learning has been widely studied for the case of distributed samples. We study an important yet less visited distributed learning problem where features are inherently distributed, and sharing of raw data or model parameters among parties is prohibited. We propose an ADMM sharing framework to approach risk minimization over distributed features, where each party only needs to share a single value for each sample in the training process, thus minimizing the data leakage risk. We introduce a novel differentially private ADMM sharing algorithm and bound the privacy guarantee with carefully designed noise perturbation. Experiments show that the proposed algorithm converges efficiently, demonstrating advantage over SGD for data with high dimensional features.

1 Introduction

The effectiveness of a machine learning model does not only depend on the quantity of samples, but also on the availability of high-quality features. Recently, a wide range of distributed or federated learning schemes, including gradient-based methods [1, 2] and ADMM-based methods [3, 4, 5], have been proposed to enable learning from distributed samples, since collecting data centrally will incur compliance overhead and privacy concerns. Most existing schemes, however, are under the umbrella of *data parallel* schemes, where multiple parties possess different training samples.

An equally important scenario is to collaboratively learn from distributed *features*, where multiple parties possess different features of a same sample, yet do not wish to share these features with each other. Examples include a user's behavioural data logged by multiple apps, a patient's record stored at different hospitals and clinics, a user's investment behavior logged by multiple financial institutions, etc. The question is—how can we train a joint model to make predictions about a sample leveraging the potentially rich and vast features possessed by other parties, in a federated learning fashion?

The motivation of gleaning insights from vertically partitioned data dates back to association rule mining [6, 7]. A few recent studies [8, 9, 10, 11, 12, 13, 14] have revisited vertically partitioned features under the setting of distributed learning, which is motivated by the increase in feature dimensionality as well as the opportunity of cooperation between multiple parties that may hold different aspects of information about the same samples.

In this paper, we propose an ADMM algorithm to solve the empirical risk minimization (ERM) problem, a general optimization formulation of many machine learning models visited by a number of recent studies on distributed machine learning [10, 15]. We propose an ADMM-sharing-based distributed algorithm to solve ERM, in which each participant does not need to share any raw features

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

or local model parameters to other parties. Instead, each party only transmits a single value for each sample to other parties, thus largely preventing the local features from being disclosed.

To further provide privacy guarantees, we present a privacy-preserving version of the ADMM sharing algorithm, in which the transmitted value from each party is perturbed by a carefully designed Gaussian noise to achieve the notion of ϵ, δ-differential privacy [16, 17]. For distributed features, the perturbed algorithm ensures that the probability distribution of the values shared is relatively insensitive to any change to a single feature in a party's local dataset. Experimental results on two realistic datasets suggest that the proposed ADMM sharing algorithm can converge efficiently. Compared to the gradient-based method, our method can scale as the number of features increases and yields robust convergence. The algorithm can also converge with moderate amounts of Gaussian perturbation added, therefore enabling the utilization of features from other parties to improve the local machine learning task.

Related Work. [18, 19] apply differential privacy (DP) to collaborative machine learning, with an inherent tradeoff between the privacy cost and utility achieved by the trained model. Recently, DP has been applied to ADMM algorithms to solve multi-party machine learning problems [3, 4, 20, 21]. However, all the work above is targeting the data-parallel scenario, where samples are distributed among nodes. The uniqueness of our work is to enable privacy-preserving machine learning among nodes with vertically partitioned features, i.e., the feature-parallel setting, which is equally important yet relatively under-explored.

[6, 22] are among the early studies that investigate the privacy issue of querying vertically partitioned data. [9] adopts a random-kernel-based method to mine vertically partitioned data privately. However, these studies provide privacy guarantees for simpler static queries instead of model training. [11] proposes a composite model structure that can jointly learn from distributed features via a SGD-based algorithm and its DP-enabled version, yet without offering theoretical privacy guarantees. Our work establishes (ϵ, δ)-differential privacy guarantee result for federated learning over distributed features. Experimental results further suggest that our ADMM sharing method converges in fewer epochs than gradient methods in the case of high dimensional features. This is critical to preserving privacy in machine learning since the privacy loss increases as the number of epochs increases [17].

2 Empirical Risk Minimization over Distributed Features

Consider N samples, each with d features distributed on M parties, which do not wish to share data with each other. The entire dataset $\mathcal{D} \in \mathbb{R}^N \times \mathbb{R}^d$ can be viewed as M vertical partitions $\mathcal{D}_1, \ldots, \mathcal{D}_M$, where $\mathcal{D}_m \in \mathbb{R}^N \times \mathbb{R}^{d_m}$ denotes the data possessed by the mth party and d_m is the dimension of features on party m. Clearly, $d = \sum_{m=1}^M d_m$. Let $\mathcal{D}^i$ denote the ith row of $\mathcal{D}$, and $\mathcal{D}_m^i$ be the ith row of $\mathcal{D}_m$ ($i = 1, \cdots, N$). Let $Y_i \in \{-1, 1\}^N$ be the label of sample i.

Let $x = (x_1^\top, \cdots, x_m^\top, \cdots, x_M^\top)^\top$ represent the model parameters, where $x_m \in \mathbb{R}^{d_m}$ are the local parameters associated with the mth party. The objective is to find a model $f(\mathcal{D}^i; x)$ with parameters x to minimize the regularized empirical risk, i.e.,

$$\underset{x \in X}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^N l_i(f(\mathcal{D}^i; x), Y_i) + \lambda R(x),$$

where $X \subset \mathbb{R}^d$ is a closed convex set and the regularizer $R(\cdot)$ prevents overfitting.

Similar to recent literature on distributed machine learning [10, 23], ADMM [4, 3], and privacy-preserving machine learning [15, 24], we assume the loss has a form $\sum_{i=1}^N l_i(f(\mathcal{D}^i; x), Y_i) = \sum_{i=1}^N l_i(\mathcal{D}^i x, Y_i) = l\left(\sum_{m=1}^M \mathcal{D}_m^i x_m\right)$, where we have abused the notation of l and in the second equality absorbed the label Y_i into the loss l, which can be a non-convex function. This framework incorporates a wide range of commonly used models including support vector machines, Lasso, logistic regression, boosting, etc.

Therefore, the risk minimization over distributed features, or vertically partitioned datasets $\mathcal{D}_1, \ldots, \mathcal{D}_M$, can be written in the following compact form:

$$\underset{x}{\text{minimize}} \quad l\left(\sum_{m=1}^{M} \mathcal{D}_m x_m\right) + \lambda \sum_{m=1}^{M} R_m(x_m), \tag{1}$$

$$\text{subject to} \quad x_m \in X_m, m = 1, \ldots, M, \tag{2}$$

where $X_m \subset \mathbb{R}^{d_m}$ is a closed convex set for all m.

We have further assumed the regularizer is separable such that $R(x) = \sum_{m=1}^{M} R_m(x_m)$. This assumption is consistent with our algorithm design philosophy—under vertically partitioned data, we require each party focus on training and regularizing its local model x_m, without sharing any local model parameters or raw features to other parties at all.

3 Differentially Private ADMM Sharing Algorithm

Differential privacy [17, 25] is a notion that ensures a strong guarantee for data privacy. The intuition is to keep the query results from a dataset relatively close if one of the entries in the dataset changes, by adding some well designed random noise into the query, so that little information on the raw data can be inferred from the query. Formally speaking, a randomized algorithm $\mathcal{M}$ is $(\varepsilon, \delta)-$differentially private if for all $S \subseteq \text{range}(\mathcal{M})$, and for all x and y, such that $|x - y|_1 \leq 1$, we have $\Pr(\mathcal{M}(x) \in S) \leq \exp(\varepsilon)\Pr(\mathcal{M}(y) \in S) + \delta$. We present an differentially private variant of ADMM sharing algorithm [26, 27] to solve Problem (1). Our algorithm requires each party only share a single value to other parties in each iteration, thus requiring the minimum message passing.

In particular, Problem (1) is equivalent to

$$\underset{x}{\text{minimize}} \quad l\,(z) + \lambda \sum_{m=1}^{M} R_m(x_m), \tag{3}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} \mathcal{D}_m x_m - z = 0, \quad x_m \in X_M, m = 1, \ldots, M, \tag{4}$$

where z is an auxiliary variable. The corresponding augmented Lagrangian is given by

$$\mathcal{L}(\{x\}, z; y) = l(z) + \lambda \sum_{m=1}^{M} R_m(x_m) + \langle y, \sum_{m=1}^{M} \mathcal{D}_m x_m - z\rangle + \frac{\rho}{2}\| \sum_{m=1}^{M} \mathcal{D}_m x_m - z\|^2, \tag{5}$$

where y is the dual variable and ρ is the penalty factor. In the t^{th} iteration of the algorithm, variables are updated according to

$$x_m^{t+1} := \underset{x_m \in X_m}{\text{argmin}} \quad \lambda R_m(x_m) + \langle y^t, \mathcal{D}_m x_m\rangle + \frac{\rho}{2}\Big\| \sum_{k=1,\ k\neq m}^{M} \mathcal{D}_k \tilde{x}_k^t + \mathcal{D}_m x_m - z^t \Big\|^2, \tag{6}$$

$$\tilde{x}_m^{t+1} := x_m^{t+1} + \mathcal{N}(0, \sigma_{m,t+1}^2), \quad m = 1, \cdots, M, \tag{7}$$

$$z^{t+1} := \underset{z}{\text{argmin}} \quad l(z) - \langle y^t, z\rangle + \frac{\rho}{2}\Big\| \sum_{m=1}^{M} \mathcal{D}_m \tilde{x}_m^{t+1} - z \Big\|^2, \tag{8}$$

$$y^{t+1} := y^t + \rho\Big(\sum_{m=1}^{M} \mathcal{D}_m \tilde{x}_m^{t+1} - z^{t+1}\Big), \tag{9}$$

where $\mathcal{N}(\mu, \sigma^2)$ represents Gaussian noise. Formally, in a distributed and fully parallel manner, the algorithm is described in Algorithm 1. Note that each party m needs the value $\sum_{k\neq m} \mathcal{D}_k \tilde{x}_k^t - z^t$ to complete the update, and Lines 3, 4 and 12 in Algorithm 1 present a trick to reduce communication overhead. On each local party , (6) is computed where a proper x_m is derived to simultaneously minimize the regularizer and bring the global prediction close to z^t, given the local predictions from other parties. When $R_m(\cdot)$ is l_2 norm, (6) becomes a trivial quadratic program which can be

Algorithm 1 The ADMM Sharing Algorithm

—–*Each party m performs in parallel:*
for t in $1, \ldots, T$ **do**
  Pull $\sum_k \mathcal{D}_k \tilde{x}_k^t - z^t$ and y^t from central node
  Obtain $\sum_{k \neq m} \mathcal{D}_k \tilde{x}_k^t - z^t$ by subtracting the locally cached $\mathcal{D}_m \tilde{x}_m^t$ from the pulled value $\sum_k \mathcal{D}_k \tilde{x}_k^t - z^t$
  Compute $\tilde{x}_m^{t+1}$ according to (6) and (7)
  Push $\mathcal{D}_m \tilde{x}_m^{t+1}$ to the central node
—–*Central node:*
for t in $1, \ldots, T$ **do**
  Collect $\mathcal{D}_m \tilde{x}_m^{t+1}$ for all $m = 1, \ldots, M$
  Compute z^{t+1} according to (8)
  Compute y^{t+1} according to (9)
  Distribute $\sum_k \mathcal{D}_k \tilde{x}_k^{t+1} - z^{t+1}$ and y^{t+1} to all the parties.

efficiently solved. We perturb the shared value $\mathcal{D}_m x_m^{t+1}$ in Algorithm 1 with a carefully designed random noise in 7 to provide differential privacy. On the central node, the global prediction z is found in (8) by minimizing the loss $l(\cdot)$ while bringing z close to the aggregated local predictions from all local parties. Therefore, the computational complexity of (8) is independent of the number of features, thus making the proposed algorithm scalable to a large number of features, as compared to SGD or Frank-Wolfe algorithms.

4 Analysis

We demonstrate that Algorithm 1 guarantees (ε, δ) differential privacy with outputs $\{\mathcal{D}_m \tilde{x}_m^{t+1}\}_{t=0,1,\cdots,T-1}$ for some carefully selected $\sigma_{m,t+1}$. We introduce a set of assumptions widely used by the literature.

Assumption 1 1. *The feasible set $\{x, y\}$ and the dual variable z are bounded; their l_2 norms have an upper bound b_1.*

2. *The regularizer $R_m(\cdot)$ is doubly differentiable with $|R''_m(\cdot)| \leq c_1$, c_1 being a finite constant.*

3. *Each row of $\mathcal{D}_m$ is normalized and has an l_2 norm of 1.*

Note that Assumption 1.1 is adopted in [28] and [29]. Assumption 1.2 comes from [21] and Assumption 1.3 comes from [21] and [28]. As a typical method in differential privacy analysis, we first study the l_2 sensitivity of $\mathcal{D}_m x_m^{t+1}$, which is defined by:

Definition 1 *The l_2-norm sensitivity of $\mathcal{D}_m x_m^{t+1}$ is defined by:*

$$\Delta_{m,2} = \max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}'_m x_{m,\mathcal{D}'_m}^{t+1}\|.$$

where $\mathcal{D}_m$ and $\mathcal{D}'_m$ are two neighbouring datasets differing in only one feature column, and $x_{m,\mathcal{D}_m}^{t+1}$ is the x_m^{t+1} derived from the first line of equation (6) under dataset $\mathcal{D}_m$.

We have Lemma 1 state the upper bound of the l_2-norm sensitivity of $\mathcal{D}_m x_m^{t+1}$.

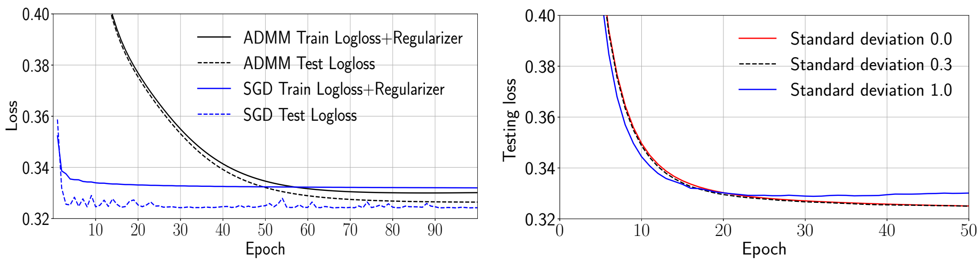
Lemma 1 *Assume that Assumption 1 hold.*

Then the l_2-norm sensitivity of $\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1}$ is upper bounded by $\mathbb{C} = \frac{3}{d_m \rho}\left[\lambda c_1 + (1 + M\rho) b_1\right]$.

We have Theorem 1 for differential privacy guarantee in each iteration.

Theorem 1 *Assume assumptions 1.1-1.3 hold and $\mathbb{C}$ is the upper bound of $\Delta_{m,2}$. Let $\varepsilon \in (0, 1]$ be an arbitrary constant and let $\mathcal{D}_m \xi_m^{t+1}$ be sampled from zero-mean Gaussian distribution with variance $\sigma_{m,t+1}^2$, where*

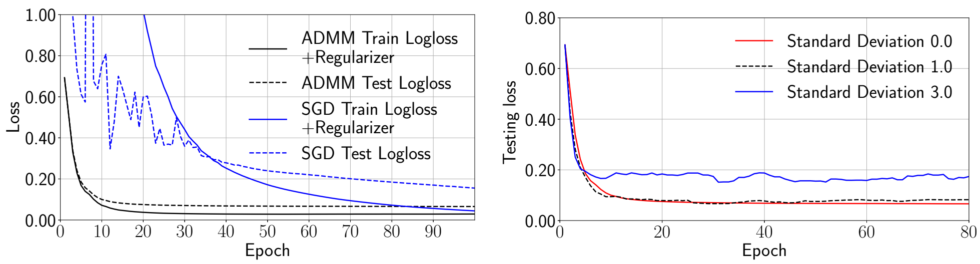
$$\sigma_{m,t+1} = \frac{\sqrt{2ln(1.25/\delta)}\mathbb{C}}{\varepsilon}.$$



(a) Loss vs. epoch

(b) Test log loss under different noise levels

Figure 1: Performance over the *a9a* data set with 32561 training samples, 16281 testing samples and 123 features.



(a) Loss vs. epoch

(b) Test log loss under different noise levels

Figure 2: Performance over the *gisette* data set with 6000 training samples, 1000 testing samples and 5000 features.

Then each iteration guarantees (ε, δ)-differential privacy. Specifically, for any neighboring datasets $\mathcal{D}_m$ and $\mathcal{D}'_m$, for any output $\mathcal{D}_m \tilde{x}^{t+1}_{m,\mathcal{D}_m}$ and $\mathcal{D}'_m \tilde{x}^{t+1}_{m,\mathcal{D}'_m}$, the following inequality always holds:

$$P(\mathcal{D}_m \tilde{x}^{t+1}_{m,\mathcal{D}_m} | \mathcal{D}_m) \leq e^{\varepsilon} P(\mathcal{D}'_m \tilde{x}^{t+1}_{m,\mathcal{D}'_m} | \mathcal{D}'_m) + \delta.$$

With an application of the composition theory in [17], we come to a result stating the overall privacy guarantee for the training procedure.

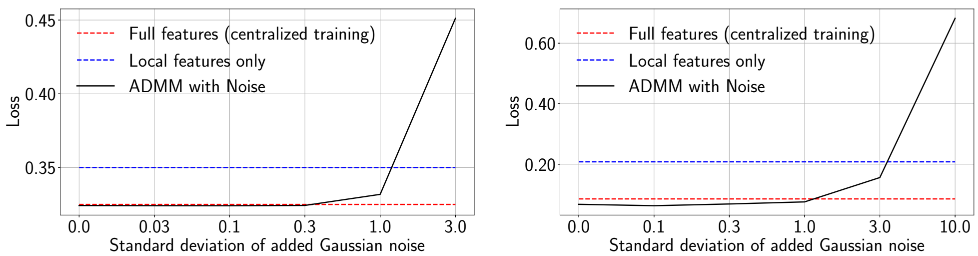
Corollary 1 *For any $\delta' > 0$, Algorithm 1 satisfies $(\varepsilon', T\delta + \delta')-$differential privacy within T epochs of updates, where*

$$\varepsilon' = \sqrt{2T ln(1/\delta')}\varepsilon + T\varepsilon(e^{\varepsilon} - 1). \tag{10}$$

Without surprise, the overall differential privacy guarantee may drop dramatically if the number of epochs T grows to a large value, since the number of exposed results grows linearly in T. However, as we will show in the experiments, the ADMM-sharing algorithm converges fast, taking much fewer epochs to converge than SGD when the number of features is relatively large. Therefore, it is of great advantage to use ADMM sharing for wide features as compared to SGD or Frank-Wolfe algorithms. When T is confined to less than 20, the risk of privacy loss is also confined.

5 Experiments

We test our algorithm by training l_2-norm regularized logistic regression on two popular public datasets, namely, *a9a* from UCI [30] and *giette* [31]. We get the datasets from [32] so that we follow the same preprocessing procedure listed there. *a9a* dataset is 4 MB and contains 32561 training samples, 16281 testing samples and 123 features. We divide the dataset into two parts, with the first part containing the first 66 features and the second part remaining 57 features. The first part is regarded as the local party who wishes to improve its prediction model with the help of data from the other party. On the other hand, *gisette* dataset is 297 MB and contains 6000 training samples, 1000 testing samples and 5000 features. Similarly, we divide the features into 3 parts, the first 2000 features being the first part regarded as the local data, the next 2000 features being the second part, and the remaining 1000 as the third part. Note that *a9a* is small in terms of the number of features and *gisette* has a relatively higher dimensional feature space.



(a) *a9a* data set

(b) *gisette* data set

Figure 3: Test performance for ADMM under different levels of added noise.

A prototype system is implemented in *Python* to verify our proposed algorithm. Specifically, we use optimization library from *scipy* to handle the optimization subproblems. We apply L-BFGS-B algorithm to do the x update in (6) and entry-wise optimization for z in (8). We run the experiment on a machine equipped with Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz and 128 GB of memory.

We compare our algorithm against an SGD based algorithm proposed in [11]. We keep track of the training objective value (log loss plus the l_2 regularizer), the testing log loss for each epoch for different datasets and parameter settings. We also test our algorithm with different levels of Gaussian noise added. In the training procedure, we initialize the elements in x, y and z with 0 while we initialize the parameter for the SGD-based algorithm with random numbers.

Fig. 1 and Fig. 2 show a typical trace of the training objective and testing log loss against epochs for *a9a* and *gisette*, respectively. On *a9a*, the ADMM algorithm is slightly slower than the SGD based algorithm, while they reach the same testing log loss in the end. On *gisette*, the SGD based algorithm converges slowly while the ADMM algorithm is efficient and robust. The testing log loss from the ADMM algorithm quickly converges to 0.08 after a few epochs, but the SGD based algorithm converges to only 0.1 with much more epochs. This shows that the ADMM algorithm is superior when the number of features is large. In fact, for each epoch, the x update is a trivial quadratic program and can be efficiently solved numerically. The z update contains optimization over computationally expensive functions, but for each sample, it is always an optimization over a single scalar so that it can be solved efficiently via scalar optimization and scales with the number of features.

Moreover, Corollary 1 implies that the total differential privacy guarantee will be stronger if the number of epochs required for convergence is less. The fast convergence rate of the ADMM sharing algorithm also makes it more appealing to achieve differential privacy guarantees than SGD, especially in the case of wide features (*gisette*).

Fig. 3 shows the testing loss for ADMM with different levels of Gaussian noise added. The other two baselines are the logistic regression model trained over all the features (in a centralized way) and that trained over only the local features in the first party. The baselines are trained with the built-in logistic regression function from *sklearn* library. We can see that there is a significant performance boost if we employ more features to help training the model on Party 1. Interestingly, in Fig. 3(b), the ADMM sharing has even better performance than the baseline trained with all features with *sklearn*. It further shows that the ADMM sharing is better at datasets with a large number of features.

Moreover, after applying moderate random perturbations, the proposed algorithm can still converge in a relatively small number of epochs, as Fig. 1(b) and Fig. 2(b) suggest, although too much noise may ruin the model. Therefore, ADMM sharing algorithm under moderate perturbation can improve the local model and the privacy cost is well contained as the algorithm converges in a few epochs.

6 Conclusion

We study learning over distributed features (vertically partitioned data) where none of the parties shall share the local data. We propose the parallel ADMM sharing algorithm to solve this challenging problem where only intermediate values are shared, without even sharing model parameters. To further protect the data privacy, we apply the differential privacy technique in the training procedure to derive a privacy guarantee within T epochs. We implement a prototype system and evaluate the proposed algorithm on two representative datasets in risk minimization. The result shows that the ADMM sharing algorithm converges efficiently, especially on dataset with large number of features. Furthermore, the differentially private ADMM algorithm yields better prediction accuracy than model trained from only local features while ensuring a certain level of differential privacy guarantee.

References

[1] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[2] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, pages 1223–1231, 2013.

[3] Xueru Zhang, Mohammad Mahdi Khalili, and Mingyan Liu. Improving the privacy and accuracy of admm-based distributed algorithms. In *International Conference on Machine Learning*, pages 5791–5800, 2018.

[4] Tao Zhang and Quanyan Zhu. A dual perturbation approach for differential private admm-based distributed empirical risk minimization. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 129–137. ACM, 2016.

[5] Zonghao Huang, Rui Hu, Yanmin Gong, and Eric Chan-Tin. Dp-admm: Admm-based distributed learning with differential privacy. *arXiv preprint arXiv:1808.10101*, 2018.

[6] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

[7] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2003.

[8] Jian Lou and Yiu-ming Cheung. Uplink communication efficient differentially private sparse optimization with feature-wise distributed data. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. Privacy via the johnson-lindenstrauss transform. *Journal of Privacy and Confidentiality*, 5, 2013.

[10] Bicheng Ying, Kun Yuan, and Ali H Sayed. Supervised learning under distributed features. *IEEE Transactions on Signal Processing*, 67(4):977–992, 2018.

[11] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *Proceedings of KDD '19*. ACM, 2019.

[12] Christina Heinze-Deml, Brian McWilliams, and Nicolai Meinshausen. Preserving differential privacy between features in distributed estimation. *stat*, 7:e189, 2018.

[13] Wenrui Dai, Shuang Wang, Hongkai Xiong, and Xiaoqian Jiang. Privacy preserving federated big data analysis. In *Guide to Big Data Applications*, pages 49–82. Springer, 2018.

[14] Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. A distributed frank-wolfe algorithm for communication-efficient sparse learning. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 478–486. SIAM, 2015.

[15] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[16] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[17] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[18] Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, pages 1876–1884, 2010.

[19] Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics*, pages 933–941, 2012.

[20] Chunlei Zhang, Muaz Ahmad, and Yongqiang Wang. Admm based privacy-preserving decentralized optimization. *IEEE Transactions on Information Forensics and Security*, 14(3):565–580, 2019.

[21] Tao Zhang and Quanyan Zhu. Dynamic differential privacy for admm-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187, 2017.

[22] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Annual International Cryptology Conference*, pages 528–544. Springer, 2004.

[23] Yi Zhou, Yaoliang Yu, Wei Dai, Yingbin Liang, and Eric Xing. On convergence of model parallel proximal gradient algorithm for stale synchronous parallel system. In *Artificial Intelligence and Statistics*, pages 713–722, 2016.

[24] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *International Conference on Machine Learning*, pages 555–563, 2016.

[25] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. Security and privacy in cloud computing: A survey. In *2010 Sixth International Conference on Semantics, Knowledge and Grids*, pages 105–112. IEEE, 2010.

[26] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[27] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[28] Anand D Sarwate and Kamalika Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE signal processing magazine*, 30(5):86–94, 2013.

[29] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

[30] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[31] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[32] Libsvm data: Classification (binary class). `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`. Accessed: 2019-05-23.

7 Supplementary Materials

7.1 Proof of Lemma 1

From the optimality condition of the x update procedure in (7), we can get

$$
\begin{aligned}
\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} &= -\mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}\left[\lambda R_m'(x_{m,\mathcal{D}_m}^{t+1}) + \mathcal{D}_m^\top y^t + \rho\mathcal{D}_m^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right],\\
\mathcal{D}_m' x_{m,\mathcal{D}_m'}^{t+1} &= -\mathcal{D}_m'(\rho\mathcal{D}_m'^\top\mathcal{D}_m')^{-1}\left[\lambda R_m'(x_{m,\mathcal{D}_m'}^{t+1}) + \mathcal{D}_m'^\top y^t + \rho\mathcal{D}_m'^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right].
\end{aligned}
$$

Therefore we have

$$
\begin{aligned}
&\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}_m' x_{m,\mathcal{D}_m'}^{t+1}\\
&= -\mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}\left[\lambda R_m'(x_{m,\mathcal{D}_m}^{t+1}) + \mathcal{D}_m^\top y^t\mathcal{D}_m + \rho\mathcal{D}_m^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right]\\
&\quad + \mathcal{D}_m'(\rho\mathcal{D}_m'^\top\mathcal{D}_m')^{-1}\left[\lambda R_m'(x_{m,\mathcal{D}_m'}^{t+1}) + \mathcal{D}_m'^\top y^t + \rho\mathcal{D}_m'^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right]\\
&= \mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}\\
&\qquad\times\left[\lambda(R_m'(x_{m,\mathcal{D}_m'}^{t+1}) - R_m'(x_{m,\mathcal{D}_m}^{t+1})) + (\mathcal{D}_m' - \mathcal{D}_m)^\top y^t + \rho(\mathcal{D}_m' - \mathcal{D}_m)^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right]\\
&\quad + [\mathcal{D}_m'(\rho\mathcal{D}_m'^\top\mathcal{D}_m')^{-1} - \mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}]\\
&\qquad\times\left(\lambda R_m'(x_{m,\mathcal{D}_m'}^{t+1}) + \mathcal{D}_m'^\top y^t + \rho\mathcal{D}_m'^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right).
\end{aligned}
$$

Denote

$$
\begin{aligned}
\Phi_1 &= \mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}\\
&\quad\times\left[\lambda(R_m'(x_{m,\mathcal{D}_m'}^{t+1}) - R_m'(x_{m,\mathcal{D}_m}^{t+1})) + (\mathcal{D}_m' - \mathcal{D}_m)^\top y^t + \rho(\mathcal{D}_m' - \mathcal{D}_m)^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right],\\
\Phi_2 &= [\mathcal{D}_m'(\rho\mathcal{D}_m'^\top\mathcal{D}_m')^{-1} - \mathcal{D}_m(\rho\mathcal{D}_m^\top\mathcal{D}_m)^{-1}]\\
&\quad\times\left(\lambda R_m'(x_{m,\mathcal{D}_m'}^{t+1}) + \mathcal{D}_m'^\top y^t + \rho\mathcal{D}_m'^\top(\sum_{\substack{k=1\\k\neq m}}^{M}\mathcal{D}_k\tilde{x}_k - z)\right).
\end{aligned}
$$

As a result:

$$
\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}_m' x_{m,\mathcal{D}_m'}^{t+1} = \Phi_1 + \Phi_2. \tag{11}
$$

In the following, we will analyze the components in (11) term by term. The object is to prove $\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|x^{t+1}_{m,\mathcal{D}_m} - x^{t+1}_{m,\mathcal{D}'_m}\|$ is bounded. To see this, notice that

$$\begin{aligned}
&\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\mathcal{D}_m x^{t+1}_{m,\mathcal{D}_m} - \mathcal{D}'_m x^{t+1}_{m,\mathcal{D}'_m}\| \\
\leq &\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_1\| + \max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_2\|.
\end{aligned}$$

For $\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_2\|$, from assumption 1.3, we have

$$\begin{aligned}
&\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_2\| \\
\leq &\left\| \frac{2}{d_m \rho} \left(\lambda R'_m(x^{t+1}_{m,\mathcal{D}'_m}) + \mathcal{D}'^{\top}_m y^t + \rho \mathcal{D}'^{\top}_m (\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z) \right) \right\|.
\end{aligned}$$

By mean value theorem, we have

$$\begin{aligned}
&\left\| \frac{2}{d_m \rho} \left(\lambda \mathcal{D}'^{\top}_m R''_m(x_*) + \mathcal{D}'^{\top}_m y^t + \rho \mathcal{D}'^{\top}_m (\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z) \right) \right\| \\
\leq &\frac{2}{d_m \rho} \left[\lambda \|R''_m(\cdot)\| + \|y^t\| + \rho \|(\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z)\| \right].
\end{aligned}$$

For $\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_1\|$, we have

$$\begin{aligned}
&\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \leq 1}} \|\Phi_1\| \leq \big\| \mathcal{D}_m (\rho \mathcal{D}^{\top}_m \mathcal{D}_m)^{-1} \\
&\times \left[\lambda (R'_m(x^{t+1}_{m,\mathcal{D}'_m}) - R'_m(x^{t+1}_{m,\mathcal{D}_m})) + (\mathcal{D}'_m - \mathcal{D}_m)^{\top} y^t + \rho (\mathcal{D}'_m - \mathcal{D}_m)^{\top} (\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z) \right] \bigg\| \\
&\leq \rho^{-1} \|(\mathcal{D}^{\top}_m \mathcal{D}_m)^{-1}\| \left[\lambda \|R''_m(\cdot)\| + \|y^t\| + \rho \|(\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z)^{\top}\| \right] \\
&= \frac{1}{d_m \rho} \left[\lambda \|R''_m(\cdot)\| + \|y^t\| + \rho \|(\sum_{\substack{k=1 \\ k \neq m}}^{M} \mathcal{D}_k \tilde{x}_k - z)\| \right].
\end{aligned}$$

Thus by assumption 1.1-1.2

$$\begin{aligned}
&\max_{\substack{\mathcal{D}_m, D'_m \\ \|\mathcal{D}_m - D'_m\| \le 1}} \quad \|\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}'_m x_{m,\mathcal{D}'_m}^{t+1}\| \\
&\le \frac{3}{d_m \rho} \left[\lambda c_1 + \|y^t\| + \rho \| (\sum_{\substack{k=1 \\ k \ne m}}^{M} \mathcal{D}_k \tilde{x}_k - z)^\top \| \right] \\
&\le \frac{3}{d_m \rho} \left[\lambda c_1 + \|y^t\| + \rho \|z\| + \rho \sum_{\substack{k=1 \\ k \ne m}}^{M} \|\tilde{x}_k\| \right] \\
&\le \frac{3}{d_m \rho} \left[\lambda c_1 + (1 + M\rho) b_1 \right]
\end{aligned}$$

is bounded. □

7.2 Proof of Theorem 1

Proof: The privacy loss from $D_m \tilde{x}_m^{t+1}$ is calculated by:

$$\left| \ln \frac{P(\mathcal{D}_m \tilde{x}_m^{t+1} | \mathcal{D}_m)}{P(\mathcal{D}'_m \tilde{x}_m^{t+1} | \mathcal{D}'_m)} \right| = \left| \ln \frac{P(\mathcal{D}_m \tilde{x}_{m,\mathcal{D}_m}^{t+1} + \mathcal{D}_m \xi_m^{t+1})}{P(\mathcal{D}'_m \tilde{x}_{m,\mathcal{D}'_m}^{t+1} + \mathcal{D}'_m \xi_m^{\prime,t+1})} \right| = \left| \ln \frac{P(\mathcal{D}_m \xi_m^{t+1})}{P(\mathcal{D}'_m \xi_m^{\prime,t+1})} \right|.$$

Since $\mathcal{D}_m \xi_m^{t+1}$ and $\mathcal{D}'_m \xi_m^{\prime,t+1}$ are sampled from $\mathcal{N}(0, \sigma_{m,t+1}^2)$, combine with lemma 1, we have

$$\begin{aligned}
&\left| \ln \frac{P(\mathcal{D}_m \xi_m^{t+1})}{P(\mathcal{D}'_m \xi_m^{\prime,t+1})} \right| \\
&= \left| \frac{2\xi_m^{t+1} \|\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}'_m x_{m,\mathcal{D}'_m}^{t+1}\| + \|\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} - \mathcal{D}'_m x_{m,\mathcal{D}'_m}^{t+1}\|^2}{2\sigma_{m,t+1}^2} \right| \\
&\le \left| \frac{2\mathcal{D}_m \xi_m^{t+1} \mathbb{C} + \mathbb{C}^2}{2 \frac{\mathbb{C}^2 \cdot 2 \ln(1.25/\sigma)}{\varepsilon^2}} \right| \\
&= \left| \frac{(2\mathcal{D}_m \xi_m^{t+1} + \mathbb{C}) \varepsilon^2}{4\mathbb{C} \ln(1.25/\sigma)} \right|.
\end{aligned}$$

In order to make $\left| \frac{(2\mathcal{D}_m \xi_m^{t+1} + \mathbb{C}) \varepsilon^2}{4\mathbb{C} \ln(1.25/\sigma)} \right| \le \varepsilon$, we need to make sure

$$\left| \mathcal{D}_m \xi_m^{t+1} \right| \le \frac{2\mathbb{C} \ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2}.$$

In the following, we need to proof

$$P(\left| \mathcal{D}_m \xi_m^{t+1} \right| \ge \frac{2\mathbb{C} \ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2}) \le \delta \tag{12}$$

holds. However, we will proof a stronger result that lead to (12). Which is

$$P(\mathcal{D}_m \xi_m^{t+1} \ge \frac{2\mathbb{C} \ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2}) \le \frac{\delta}{2}.$$

Since the tail bound of normal distribution $\mathcal{N}(0, \sigma_{m,t+1}^2)$ is:

$$P(\mathcal{D}_m \xi_m^{t+1} > r) \le \frac{\sigma_{m,t+1}}{r\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma_{m,t+1}^2}}.$$

Let $r = \frac{2\mathbb{C}\ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2}$, we then have

$$\begin{aligned}&P(\mathcal{D}_m\xi_m^{t+1} \geq \frac{2\mathbb{C}\ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2})\\ &\leq \frac{\mathbb{C}\sqrt{2\ln(1.25/\sigma)}}{r\sqrt{2\pi}\varepsilon}\exp\left[-\frac{[4\ln(1.25/\sigma)-\varepsilon]^2}{8\ln(1.25/\sigma)}\right].\end{aligned}$$

When δ is small and let $\varepsilon \leq 1$, we then have

$$\frac{\sqrt{2\ln(1.25/\sigma)}2}{(4\ln(1.25/\sigma)-\varepsilon)\sqrt{2\pi}} \leq \frac{\sqrt{2\ln(1.25/\sigma)}2}{(4\ln(1.25/\sigma)-1)\sqrt{2\pi}} < \frac{1}{\sqrt{2\pi}}. \tag{13}$$

As a result, we can proof that

$$-\frac{[4\ln(1.25/\sigma)-\varepsilon]^2}{8\ln(1.25/\sigma)} < \ln(\sqrt{2\pi}\frac{\delta}{2})$$

by equation (13). Thus we have

$$P(\mathcal{D}_m\xi_m^{t+1} \geq \frac{2\mathbb{C}\ln(1.25/\sigma)}{\varepsilon} - \frac{\mathbb{C}}{2}) < \frac{1}{\sqrt{2\pi}}\exp(\ln(\sqrt{2\pi}\frac{\delta}{2}) = \frac{\delta}{2}.$$

Thus we proved (12) holds. Define

$$\begin{aligned}\mathbb{A}_1 &= \{\mathcal{D}_m\xi_m^{t+1} : |\mathcal{D}_m\xi_m^{t+1}| \leq \frac{1}{\sqrt{2\pi}}\exp(\ln(\sqrt{2\pi}\frac{\delta}{2}\},\\ \mathbb{A}_2 &= \{\mathcal{D}_m\xi_m^{t+1} : |\mathcal{D}_m\xi_m^{t+1}| > \frac{1}{\sqrt{2\pi}}\exp(\ln(\sqrt{2\pi}\frac{\delta}{2}\}.\end{aligned}$$

Thus we obtain the desired result:

$$\begin{aligned}&P(\mathcal{D}'_m\tilde{x}_m^{t+1}|\mathcal{D}_m)\\ &= P(\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} + \mathcal{D}_m\xi_m^{t+1} : \mathcal{D}_m\xi_m^{t+1} \in \mathbb{A}_1)\\ &+P(\mathcal{D}_m x_{m,\mathcal{D}_m}^{t+1} + \mathcal{D}_m\xi_m^{t+1} : \mathcal{D}_m\xi_m^{t+1} \in \mathbb{A}_2)\\ &< e^{\varepsilon}P(\mathcal{D}_m x_{m,\mathcal{D}'_m}^{t+1} + \mathcal{D}_m\xi_m^{\prime,t+1}) + \delta = e^{\varepsilon}P(\mathcal{D}_m\tilde{x}_m^{t+1}|\mathcal{D}'_m) + \delta.\end{aligned}$$

□