Differential Privacy Distributed Logistic Regression with Objective Function Perturbation

Haibo Yang[1], Bin Zou[1*], Yulong Ji[1] and Yingxiong Fu[1]

[1*]School of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan 430062, China.

*Corresponding author(s). E-mail(s): zoubin0502@hubu.edu.cn;

Abstract

Distributed learning is a very effective divide-and-conquer strategy for dealing with big data. As distributed learning algorithms become more and more mature, network security issues including the risk of privacy disclosure of personal sensitive data, have attracted high attention and vigilance. Differential privacy is an important method that maximizes the accuracy of a data query while minimizing the chance of identifying its records when querying from this data. The known differential privacy distributed learning algorithms are based on variable perturbation and the variable perturbation method may be non-convergence and the experimental results usually have large deviations. Therefore, in this article we consider differential privacy distributed learning algorithm based on objective function perturbation. We first propose a new distributed logistic regression algorithm based on objective function perturbation (DLR-OFP). We prove that the proposed DLR-OFP satisfies differential privacy, and obtain a fast convergence rate by introducing a new acceleration factor for the gradient descent method. The numerical experiments based on benchmark data show that the proposed DLR-OFP algorithm has fast convergence rate and good privacy protection ability.

Keywords: distributed algorithms, approximation of functions, differential privacy, logistic regression, objective function perturbation

1 Introduction

With the development of machine learning, privacy protection has become an inexorable requirement of information technology development under the era of big data, and many researchers have paid close attention to the protection of data privacy during data release [13]-[16] and query processing [17]-[18]. Dwork [4] first introduced the concept of ϵ-differential privacy. Different from the previously known privacy definitions proposed in [1]-[3], ϵ-differential privacy defines a rigorous attack model without background-independent knowledge, and it makes a quantitative representation of the degree of privacy leakage, which can be realized to protect individual privacy information while learning the overall law of a given data. A good algorithm that satisfies differential privacy is usually need to satisfy both robustness, running speed and accuracy. Due to the complexity of designing such algorithms, research on differential privacy-based machine learning algorithms has been focused recently. Chaudhuri et al. [20] first proposed the privacy algorithm based on output perturbation. Chaudhuri et al. [21] introduced the privacy ERM algorithm based on objective function perturbation. Kifer et al. [22] proposed an ERM-based differential privacy algorithm for the high-dimensional setting. Abadi et al. [24] presented several methods of target perturbation causal analysis. But for the case of big data, these algorithms above are usually very time-consuming and the algorithmic hardware requirements are very high since these algorithms above are batch learning. Thus Li et al. [25], Aldeen et al. [26] considered the privacy preserving method based on distributed learning since distributed learning is a promising way of dealing with the high demand of large-scale and it reduces the algorithmic computational burden. Han et al. [27] considered the idea of variable perturbation and proposed a distributed projection gradient descent algorithm. Ji et al. [36] proposed the distributed Newton-Raphson algorithm.

In particular, with the Alternating Direction Method of Multipliers (ADMM), learning problem can be divided into several sub-problems solved by agents independently and locally, and only intermediate parameters need to be shared. For example, Zhang et al. [28] considered the methods of dual variable perturbation and primal variable perturbation to provide dynamic differential privacy. Zhang [29] proposed an improved distributed ADMM privacy protection algorithm with variable perturbation, which greatly improves the running speed of the algorithm introduced in [28]. Wang et al.[38] considered the centralized ADMM method of distributed learning with variable perturbation by improving the ADMM method used in [28] and [29]. Huang [35] studied the centralized ADMM method of distributed learning with variable perturbation by using the first order approximation of Lagrange function to solve the ADMM. Wang and Zhang [10] considered the decentralized ADMM method of distributed logistic regression by using variable perturbation such that the distributed learning model can control the optimized target consistency. These methods proposed in [10], [28], [29], [35], [38] are based on variable

perturbation, and the method based on variable perturbation is usually non-convergence, and the final algorithmic results usually have large deviation [39]. For the centralized ADMM method, there exists the risk that the information of all distributed computer system will be compromised if the Fusion Center (see Fig.1 of [38]) is attacked. In addition, although the decentralized ADMM method is considered in [10], the added noise in [10] is dependent on sensitivity, and the change of noise will be not obvious as the value of sensitivity is smaller, which will result in the bad property of protection privacy for the case of smaller sensitivity. Then a problem is posed:

How to design a distributed differential privacy protection algorithm based on the ADMM method such that it not only has good privacy protection ability, but also has fast convergence rate ?

To solve the problem above, we consider the decentralized distributed ADMM algorithm with objective function perturbation and propose a new distributed logistic regression algorithm based on objective function perturbation (DLR-OFP), which is not dependent on sensitivity. We prove that the proposed DLR-OFP algorithm satisfies ϵ-difference privacy. We introduce an acceleration factor for the gradient descent method of DLR-OFP algorithm, estimate the convergence rate and the change bound of the introduced acceleration factor. We also present some numerical researches on the performance of the proposed algorithm. The main contributions of this article are summarized as follows:

- The proposed DLR-OFP is proved to satisfy ϵ-differential privacy.
- We introduce a new acceleration factor for the gradient descent method of the proposed DLR-OFP and establish a fast convergence rate.
- The performance of the proposed algorithm are validated by experiments for benchmark repository. As far as our knowledge, these researches are the first works on distributed ADMM differential privacy protection.

This article is arranged as follows. Section 2 presents some notions and definitions. In Section 3, the DLR-OFP algorithm is proposed. Section 4 presents the algorithmic performance of the proposed DLR-OFP. Section 5 gives the numerical experimental results. Section 6 gives some discussions on the proposed algorithm. Section 7 summarizes this article.

2 Preliminaries

In this section, we present the definition of ϵ-differential privacy, the decentralized distributed ADMM logistic regression, and some corresponding notations and definitions.

2.1 ϵ-differential privacy

ϵ-differential privacy is a suitable concept that provides a strong guarantee by removing or adding a single database item such that an adversary did not distinguish an individual data point. ϵ-differential privacy has become the

mainstream method of privacy security analysis [5]-[9], which is defined as follows.

Definition 1 ([4], [40]) Given a randomized algorithm $\mathcal{G}$, T_{all} is the set composed of all possible outputs of $\mathcal{G}$. Algorithm $\mathcal{G}$ is ϵ-differentially private if for any two datasets D_1, D_2 that differ on one element and all subset T of T_{all},

$$\frac{\mathrm{P}[\mathcal{G}(D_1) \in T]}{\mathrm{P}[\mathcal{G}(D_2) \in T]} \leq e^{\epsilon}, \tag{1}$$

where $\mathrm{P}[\mathcal{G}(D_1) \in T]$ represents the probability that the output $\mathcal{G}(D_1)$ of Algorithm $\mathcal{G}$ on D_1 belongs to T, $\epsilon \geq 0$ is the privacy budget. Inequality (1) can be equivalently stated as,

$$\frac{\mathrm{P}[\mathcal{G}(D_1) = t]}{\mathrm{P}[\mathcal{G}(D_2) = t]} \leq e^{\epsilon}, \ \forall t \in T_{all},$$

where we define $\frac{0}{0}$ to be 1.

More generally, ϵ-differential privacy can be defined by requiring inequality (1) to hold on D_1 and D_2 that are neighboring.

2.2 The distributed ADMM logistic regression

Let $D_{all} = \{D_1 \ldots, D_J\}$ be a data set stored in J separate computers that can communicate with each other. Let N_j be the neighborhood set of the j-th computer, and E be the adjacency matrix of the distributed system of J computers. If the i-th computer and the j-th computer can communicate with each other, we set $E_{ij} = 1$, otherwise $E_{ij} = 0$. The local data stored in the j-th ($1 \leq j \leq J$) computer is expressed as $D_j = \{x_i^j, y_i^j\}_{i=1}^{n_j}$ with $n = \sum_{j=1}^{J} n_j$, where $x_i^j \in \mathbb{R}^p$, $y_i^j \in \{-1, 1\}$, n_j are the input variable, the corresponding label and the size of D_j, respectively. Wang and Zhang [10] used the local variable $\{\omega_j\}_{j=1}^{J}$ instead of the global variable ω (the definition of ω, please see Equation (2) of [10]) in order to enable each computer to compute independently of the global variables, where $\omega_j \in \mathbb{R}^p$ is the coefficient vector of the j-th computer. The distributed logistic regression model introduced in [10] is defined as follows:

$$\begin{aligned} &\min_{\{\omega_j\}_{j=1}^{J}} \sum_{j=1}^{J} \{\frac{1}{n} \sum_{i=1}^{n_j} \log(1 + exp(-y_i^j \omega_j^T x_i^j)) + \frac{\lambda}{2J} \left\|\omega_j\right\|_2^2\}, \\ &\text{s.t.} \ \ \omega_j = \omega_{j'}, 1 \leq j \leq J, j' \in N_j, \end{aligned} \tag{2}$$

where λ is a regularization parameter, ω_j^T is the transpose of ω_j and $\left\|\omega_j\right\|_2$ is the ℓ_2 norm of ω_j. To turn the model (2) into the solution form of the ADMM algorithm, new local auxiliary variables $\{Z_j\}_{j=1}^{J}$ in the regularization term of

the model (2) are introduced. Then the model (2) is changed to be

$$\min_{\{\omega_j\}_{j=1}^J,\{Z_j\}_{j=1}^J} \frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\log(1+exp(-y_i^j\omega_j^T x_i^j))+\frac{\lambda}{2J}\sum_{j=1}^{J}\|Z_j\|_2^2, \quad (3)$$
$$\text{s.t.} \quad Z_i-\omega_j=0, \forall i,j=1,\ldots,J.$$

To calculate the ADMM algorithm, Wang et al. [38] considered centralized ADMM algorithm and then they used a Fusion Center to receive and process the coefficient vector $\omega_j (1 \le j \le J)$ calculated by each computer in each iteration process. Fusion Center is responsible for calculating the average value $\overline{\omega}$ of ω and sending $\overline{\omega}$ to each computer for the next iteration. Different from [38], Wang and Zhang [10] used the decentralized ADMM algorithm such that their algorithm can effectively avoid the risk that the information of distributed computer system will be compromised if the Fusion Center is attacked. Thus the iteration termination condition of this algorithm in [10] has changed. Namely, the final constraint condition $Z_i-\omega_j=0$ of the model (3) is satisfied for any i and j $(1 \le i,j \le J)$. Since the undirected graph is assumed to be connected in [10], then the constraint condition of the model (3) can be changed to be $Z_j=\omega_j$ for any j.

3 DLR-OFP algorithm

Different from [10, 25, 27, 30, 36], in this article we consider objective function perturbation method inspired by [37]. Namely, by modifying the model (3), the proposed DLR-OFP can be stated as

$$\min_{\{\omega_j\}_{j=1}^J,\{Z_j\}_{j=1}^J} \sum_{j=1}^{J}\Big[\frac{1}{n}\sum_{i=1}^{n_j}\log(1+exp(-y_i^j\omega_j^T x_i^j))+\frac{b_j^T\omega_j}{n_j}\Big]+\frac{\lambda}{2J}\sum_{j=1}^{J}\|Z_j\|_2^2,$$
$$\text{s.t.,} \quad Z_i-\omega_j=0, \forall i,j=1,\ldots,J, \quad (4)$$

where $b_j (1 \le j \le J)$ is the noise that added to the j-th computer. Inspired by [20, 22], in this paper we assume that b_j is proportional to $h(b_j)$ and the density function of $h(b_j)$ is $exp(-\frac{n\epsilon_j}{2n_j}\|b_j\|_2)$, where ϵ_j is the privacy budget of the j-th computer, $\|b_j\|$ is generated according to the distribution $\Gamma(p, 2n_j/n\epsilon_j)$ and the direction of b_j uniformly at random.

To solve the model (4), we convert the quadratic augmented Lagrangian function of the model (4) to be the following form:

$$\begin{aligned} L(Z,\omega,V) &:= L(\{Z_j\}_{j=1}^J,\{\omega_j\}_{j=1}^J,\{V_j\}_{j=1}^J) \\ &= \sum_{j=1}^{J}\Big[\frac{1}{n}\sum_{i=1}^{n_j}\log(1+exp(-y_i^j\omega_j^T x_i^j)) + \frac{b_j^T\omega_j}{n_j}\Big] + \frac{\lambda}{2J}\sum_{j=1}^{J}\|Z_j\|_2^2 \\ &+ \frac{c}{2}\sum_{j=1}^{J}\sum_{i=1}^{J}E_{ij}\big(\parallel Z_i-\omega_j+V_{ij}/c\|_2^2-\|V_{ij}/c\|_2^2\big), \end{aligned} \tag{5}$$

where V_j is the Lagrange multiplier of the j-th computer, V_{ji} is the Lagrange multiplier passed from the i-th computer to the j-th computer and $c>0$ is a pre-selected parameter [40].

To solve (5), we should update Z,ω,V, respectively. Let $k=1,...,K$ be the output loop mark number (see Algorithm 1) for the solution of (5), thus we can divide it into the following three steps.

Step 1: Update Z. The Lagrangian function equation (5) can be divided into J different sub-problems by the decomposability (5). For any $1\le j\le J$, we denote

$$Z_j(k+1)=\arg\min_{Z_j}\frac{\lambda}{2J}\|\omega_j\|_2^2+\frac{c}{2}\sum_{i=1}^{J}E_{ij}\Big(\parallel Z_j-\omega_i(k)+V_{ji}/c\|_2^2\Big), \tag{6}$$

where $\omega_i(k)$ be the coefficient vector of the i-th computer at the k-th iteration. The solution to the optimization problem (6) can be stated as follows

$$Z_j(k+1)=\frac{c\sum\limits_{i=1}^{J}E_{ij}(\omega_i(k)-V_{ji}(k)/c)}{c\sum\limits_{i=1}^{J}E_{ij}+\lambda/c}, \tag{7}$$

where $V_{ji}(k)$ be the Lagrange multiplier passed from the i-th computer to the j-th computer at the k-th iteration.

Step 2: Update the variable ω. The update of ω_j can also be split into the following J different sub-problems,

$$\begin{aligned} \omega_j(k+1) &= \arg\min_{\omega_j} L_j \\ &= \arg\min_{\omega_j}[\frac{1}{n}\sum_{i=1}^{n_j}\log(1+exp(-y_i^j\omega_j^T x_i^j)) + \frac{b_i^T\omega_j}{n_j}] \\ &+ \frac{c}{2}\sum_{i=1}^{J}E_{ij}(\parallel Z_i(k+1)-\omega_j+V_{ij}/c\|_2^2). \end{aligned} \tag{8}$$

Equation (8) is solved by the help of the Nesterov acceleration method proposed by Nesterov in [12]. Let m be the internal cycle mark number for the solution equation (8). For $m = 1, \ldots, M$, we introduce an acceleration factor φ_m in order to solve Equation (8). That is, set $\varphi_0 = 0$, and for any $m \geq 1$, set

$$\varphi_m = \frac{1 + \sqrt{1 + 4\varphi_{m-1}^2}}{2}. \tag{9}$$

Let $\gamma_m = (1 - \varphi_m)/\varphi_{m+1}$ be the control factor satisfying $-1 < \gamma_m < 0$. Define $\nu_m = \omega_j^{m-1}(k+1) - \alpha(\partial L_j/\partial \omega_j)$, where $\omega_j^{m-1}(k+1)$ is the coefficient vector of the j-th computer at the $m-1$-th internal cycle and the $k+1$-th outer cycle, α be the step length of the gradient descent and

$$\begin{aligned}\frac{\partial L_j}{\partial \omega_j} &= \frac{1}{n}\sum_{i=1}^{n_j} \frac{-y_i^j x_i^j}{1 + exp(-y_i^j \omega_j^T x_i^j)} + \frac{b_i}{n_j} \\ &- c\sum_{i=1}^{J} E_{ij}\big[Z_i(k+1) - \omega_j^{m-1}(k+1) + V_{ij}(k)/c\big].\end{aligned}$$

Then we have

$$\omega_j^m(k+1) = (1 - \gamma_{m-1})\nu_m + \gamma_{m-1}\nu_{m-1}. \tag{10}$$

Step 3: Update the Lagrange multiplier V,

$$V_{ji}(k+1) = V_{ji}(k) + c(Z_j(k+1) - \omega_i(k+1)), i = 1, \ldots, J. \tag{11}$$

Now the DLR-OFP algorithm can be presented as follows:

In Algorithm 1, δ_1 and δ_2 are two technical parameters, which will be discussed in Section 6. In Figure 1 we present the flow chart of Algorithm 1, and in Figures 2(a)-2(b), we give an example of J computers connection and the adjacency matrix for $J = 5$.

To have a better understanding Algorithm 1, we present some remarks.

Remark 1 *(i) To calculate* ADMM *algorithm, Wang et al. [38] used the Fusion Center to receive and process the coefficient vector ω calculated by each computer in each iteration process. In [35], Huang et al. considered the centralized* ADMM *algorithm and used the aggregator to receive and process ω from all distributed computers. Different from [38] and [35], Algorithm 1 is based on the decentralized* ADMM *in order to avoid effectively the risk that all distributed computer information will be compromised if the Fusion Center is attacked.*

(ii) These algorithms introduced in [10] [28], [29], [35] and [38] are based on variable perturbation. While Algorithm 1 is based on objective function perturbation since the variable perturbation method may be non-convergence, and the final experimental results usually have large deviation.

Algorithm 1 : LDAMSS

Input: Dataset $D_{all} = \{D_1, \ldots, D_J\}$, pre-selected parameter value c, gradient drop step length α, maximum number M of inner cycle and maximum number K of outer cycle, two significance constants δ_1 and δ_2.

Output: $\omega^* = \frac{1}{J}\sum_{j=1}^{J}\omega_j(K)$.

1: For all $j \in J$, letting $\omega(0) = (0, \ldots, 0)^T$, $Z(0) = (0, \ldots, 0)^T$, $V(0) = (0, \ldots, 0)^T$, $n = \sum_{j=1}^{J} n_j$

2: Each computer run independently Generate noise vectors $b_j \sim exp(-\frac{n\epsilon_j}{2n_j}\|b_j\|_2)$

3: While $k \leq K$

4: Using the arithmetic equation (7) to update $Z_j(k+1)$ and transmit the results to the neighbor computer

5: While$m \leq M$

6: Using the arithmetic equation (8) to update $\omega_j^m(k+1)$ and transmit the results to the neighbor computer

7: If $\|\omega_j^m(k+1) - \omega_j^{m-1}(k+1)\| < \delta_1$

8: stop iterating

9: Letting $\omega_j(k+1) = \omega_j^M(k+1)$ and transmit the results to the neighbor computer

10: Using the arithmetic equation (11) to update $V_{ji}(k+1)$ and transmit the results to the neighbor computer

11: If $\|\omega_j(k+1) - Z_j(k+1)\| < \delta_2$

12: stop the loop.

(iii) Although Algorithm 1 and the algorithms proposed in [10], [28] and [29] are based on the decentralized ADMM *method, the difference between Algorithm 1 and these algorithms proposed in [10], [28] and [29] are obvious: First, the added noise of the algorithms proposed in [10], [28] and [29] is dependent on sensitivity while in Algorithm 1 we assume that the noise does not depend on the sensitivity (see Definition 2 of [20]) since the method that the added noise is dependent on sensitivity may result in the bad protection privacy for the case of smaller sensitivity (the change of the added noise will be not obvious as the value of sensitivity is smaller). Second, in Algorithm 1, we introduce an acceleration factor φ_m in the iteration process of* ADMM *method inspired by the idea from the Nesterov acceleration method [12] to improve the convergence speed of the gradient descent method.*

4 Estimating the algorithmic performance

In this section, we present some theoretical studies on the performance of the proposed DLR-OFP algorithm. We first prove that the DLR-OFP algorithm satisfies differential privacy, and estimate the convergence rate of the proposed DLR-OFP algorithm. We also establish the bound on the change of weight vector ω_j obtained by the j-th computer for the gradient

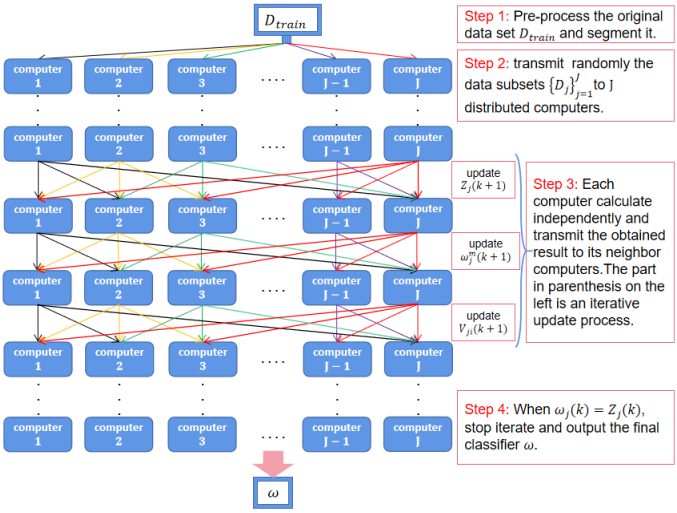
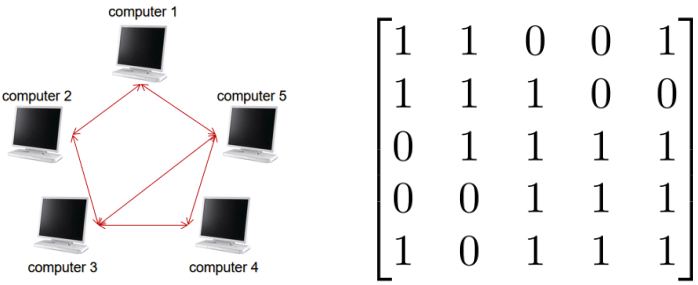


Fig. 1 Flow chart of Algorithm 1



(a) (b)

Fig. 2 An example for $J = 5$ (a): connection instance of J computers (b): the adjacent matrix of $J = 5$

descent method with acceleration factor φ_m. The main tools are the following definitions of β-smooth, p-strongly convex.

Definition 2 [41] Suppose f is differentiable, and its gradient $\bigtriangledown f$ exists at each point in the domain. Let x_1, x_2 be any two constants in the domain of function f. The function f is called to be ξ-strongly convex if f satisfies

$$f(x_1) \geq f(x_2) + \bigtriangledown f(x_2)^T (x_1 - x_2) + \frac{\xi}{2} \parallel x_1 - x_2 \parallel_2^2 .$$

In addition, let $\beta \geq 0$, the function f is called to be β-smooth if the inequality

$$\parallel \bigtriangledown f(x_1) - \bigtriangledown f(x_2) \parallel_2 \leq \beta \parallel x_1 - x_2 \parallel_2$$

holds true.

Theorem 1 We assume that $\|x\|_2 \leq 1$ for any input $x \in D_{all}$. Let G be the set of all possible outputs during the algorithm iteration for a given D_{all}, $D = \{D_j\}_{j=1}^{J}$ and $D' = \{D_j{}'\}_{j=1}^{J}$ be two data sets of D_{all}, where D_j and $D_j{}'$ be two neighboring data sets. Then the proposed DLR-OFP algorithm $\mathcal{A}$ satisfies ϵ-differential privacy with $\epsilon = \sum_{j=1}^{J} \epsilon_j$. Namely,

$$\frac{P(\{\mathcal{A}(k)\}_{k=1}^{K} \in G \| D)}{P(\{\mathcal{A}(k)\}_{k=1}^{K} \in G \| D')} \leq e^{\epsilon},$$

where $\epsilon \geq 0$ is the privacy budget, $\mathcal{A}(k)$ is the k-th time outer cycle output of algorithm $\mathcal{A}$ and K is defined in Algorithm 1.

The proof of Theorem 1 is shown in Appendix A. In Algorithm 1, we introduce the Nesterov acceleration method in the process of gradient descent. Inspired by the idea from [12], we can easily obtain the following convergence rate of the proposed DLR-OFP algorithm for the case of single computer by making use of the similar proof method of [12].

Theorem 2 *Let ω_0 be the initial coefficient vector of gradient descent, ω^* be the coefficient vector output after gradient descent training, and $\omega_{T'}$ be the coefficient vector of the T'-th gradient descent. If the objective function f is a β-smooth convex function and the step length η satisfies $\eta = 1/\beta$, then the acceleration method used in Algorithm 1 has the following convergence rate*

$$f(\omega_{T'}) - f(\omega^*) \leq \frac{2\beta \|\omega_0 - \omega^*\|^2}{T'^2}.$$

Further, if f is also ξ-strongly convex, then we have

$$f(\omega_{T'}) - f(\omega^*) \leq \frac{\xi + \beta}{2} e^{-\frac{T'}{\sqrt{Q}}} \left\| \omega_0 - \omega^* \right\|^2,$$

where $Q = \beta/\xi$, β, ξ are defined in Definition 2.

For the acceleration factor φ_m defined in (9), we have

Theorem 3 Let m be the cycle label of gradient descent, and u be a positive constant satisfying $\frac{1}{2} \leq u \leq \frac{2}{5}$. If $m \geq 3$, then the following bound holds true.

$$\frac{m}{2} + u \log_2 m \leq \varphi_m \leq \frac{m}{2} + \frac{5 \log_2 m}{2}. \tag{12}$$

The proof of Theorem 3 is proved in Appendix B. For any two neighboring data sets D_1, D_2 , let $L = L(Z, \omega, V)$, $L(Z, \omega, V)$ is defined in (5). If L satisfies $\|\partial L/\partial \omega\| \leq c_1$, then we have

$$\| \frac{\partial L_j}{\partial \omega_j^*} - \frac{\partial L_j}{\partial \omega_j} \| \leq 2c_1,$$

where $\partial L_j/\partial\omega_j$ is defined in (10), ω_j^* and ω_j be the corresponding weight vectors generated by the gradient descent method with D_1 and D_2 in Algorithm 1. Thus, we have the following bound on the difference of the weight vectors of D_1 and D_2 is valid.

Theorem 4 For any neighboring data sets D_1 and D_2, let ω_j^* and ω_j be the corresponding weight vectors generated by the gradient descent method with D_1 and D_2 in Algorithm 1, respectively. If $4 \leq m \leq M$, we have

$$\max_{D_1,D_2} \|\omega_j^* - \omega_j\| \leq c_1 M^2/2.$$

The proof of Theorem 4 is shown in Appendix B. To give a better showing these results obtained in Theorems 1-4, we give the following remarks.

Remark 2 *Wang et al. [10] proved that the distributed variable perturbation ADMM algorithm introduced in [10] satisfies ϵ-difference privacy and analyzed the convergence rate of their proposed three-step* ADMM *algorithm. In [35], Huang et al. proved that their distributed variable perturbation ADMM algorithm with time-varying noise satisfied (ϵ, δ)-difference privacy. In [38], Wang et al. not only considered the centralized ADMM distributed logistic regression algorithm and proved the convergence property of their proposed algorithm, but also given the bound of the accumulated difference, and obtained the bound on the final solved classifier ω_j.*

Different from [10], [35] and [38], in Theorems 1-4, we not only consider the decentralized ADMM distributed algorithm with objective function perturbation, prove the proposed DLR-OFP algorithm satisfies difference privacy (see Theorem 1), but also we establish the fast convergence rate of Nesterov acceleration method used in gradient descent (see Theorem 2), obtain the bound of the acceleration factor φ_m (see Theorem 3) and the bound the change of sensitivity during gradient descent (see Theorem 4). In particular, By Theorem 3, we can find that the obtained convergence rate of the gradient descent method is $1/T^2$, which improves the known convergence rate $1/T$ established in [10] and [38]. As far as our knowledge, these results presented in Theorems 1-4 are the first works on the distributed ADMM with objective function perturbation.

5 Experiments results

We give the numerical researches on the performance of the proposed DLR-OFP algorithm from the kaggle platform including 9 data sets: telescope [1], wine [2], rain [3], phishing [4], sensor [5], trojan [6], hedge [7], android [8], anti [9], are

[1] https://www.kaggle.com/abhinand05/magic-gamma-telescope-dataset
[2] https://www.kaggle.com/hufe09/winequality
[3] https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
[4] https://www.kaggle.com/eswarchandt/phishing-website-detector
[5] https://www.kaggle.com/nphantawee/pump-sensor-data
[6] https://www.kaggle.com/subhajournal/trojan-detection/code
[7] https://www.kaggle.com/datasicencelab/hedge-fund-x-financial-modeling-challenge
[8] https://www.kaggle.com/saurabhshahane/android-permission-dataset
[9] https://www.kaggle.com/rafay12/anti-freeze-protein-classification

the public datasets for differential privacy ([10], [35], [38]). For each data set, we remove non-numerical features and randomly separate it into a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$ according to the ratio of 7:3. The information of these data sets are summarized in Table 1.

Table 1 9 Public Datasets

Dataset	attributes	data size	positive cases	negative cases
telescope	10	19020	6688	12332
wine	11	6497	4898	1599
rain	18	142194	63757	65536
phishing	30	11054	6157	4897
sensor	51	220313	205836	14477
trojan	80	177482	90683	86799
hedge	88	10000	4994	5006
android	177	29999	20000	9999
anti	840	10823	1330	9493

Now we briefly describe our experimental procedures as follows:

(i) We first pre-processed 9 data sets as follows: For every data set, we first combine the multiple CSV files into a single file, and we removed non-numeric columns, empty columns and columns with data numbers from the given data set, set all defaults and values marked NA to be 0, and change the classification label to be 1 or -1. We normalized all the data sets except phishing and android data by using the Z-score standardization method in order to prevent possible numerical overflow problem in the numerical calculations.

(ii) We set the adjacency matrix E of distributed computer to be 1^{J*J}. Since $\epsilon = \sum_{j=1}^{J} \epsilon_j$ (see Theorem 1), for simplicity, we assume that $\epsilon_1 = \cdots = \epsilon_J = \epsilon^*$, where $\epsilon_j (1 \leq j \leq J)$ is the privacy budget of the j-th computer.

(iii) For a given training set $\mathcal{D}_{train}$, we train our algorithm (Algorithm 1), DLP algorithm [10], PPD algorithm [38] and B-ADMM algorithm [31], respectively and obtain the corresponding classifiers. We test them on the same test set $\mathcal{D}_{test}$. We combine $\mathcal{D}_{train}$ with $\mathcal{D}_{test}$, and redivide into another new training set $\mathcal{D}'_{train}$ and another new test set $\mathcal{D}'_{test}$ randomly. The sizes of $\mathcal{D}'_{train}$ and $\mathcal{D}'_{test}$ are same as that of $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, respectively. We repeat the above procedures 10 times and compute the corresponding accuracy, the mean-square error (MSE) and the total time. The regulation parameter λ of DLR-OFP algorithm is chosen by 5-fold cross validation.

5.1 Comparisons with the algorithms introduced in [10], [31] and [38]

In this section, we compare the proposed algorithm with three distributed differential privacy algorithms proposed in [10], [31] and [38]. In Tables 2-4, we present the experimental results on accuracy, MSE and the total time of

Table 2 Accuracy (%) of DLR-OFP, DLP, PPD and B-ADMM for $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$

Dataset	Accuracy (%)			
	B-ADMM	DLP	PPD	DLR-OFP
telescope	65.25±1.13	70.86±1.21	72.97±0.58	**77.45±0.17**
wine	98.65±**0.50**	99.27±0.55	90.70±0.98	**99.62**±0.60
rain	86.57±0.41	84.18±**0.28**	77.31±0.56	**88.87**±0.31
phishing	82.60±3.87	81.02±4.52	75.62±0.53	**84.67±0.36**
sensor	81.07±1.63	86.36±1.61	83.56±1.23	**88.62±1.07**
trojan	71.94±2.27	73.33±1.94	68.01±**1.75**	**76.41**±2.16
hedge	71.86±0.87	82.79±0.71	59.76±0.81	**88.62±0.64**
android	70.24±1.67	**72.42**±1.62	66.67±**1.16**	71.61±1.28
anti	65.66±2.88	67.21±2.18	69.39±**0.45**	**86.75**±0.91

Table 3 MSE of DLR-OFP, DLP, PPD and B-ADMM for $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$

Dataset	MSE			
	B-ADMM	DLP	PPD	DLR-OFP
telescope	0.3250±0.0028	0.3179±**0.0027**	0.5794±0.0032	**0.2964**±0.0046
wine	0.8941±**0.0026**	0.9052±0.0061	0.9020±0.0042	**0.8943**±0.0035
rain	0.8486±0.0297	0.8335±0.0315	0.7854±0.0440	**0.6895±0.0276**
phishing	0.5958±0.0291	0.6031±0.0280	**0.5513**±0.0256	0.5780±**0.0116**
sensor	0.5942±0.0147	0.6788±**0.0115**	0.8513±0.0170	**0.4159**±0.0301
trojan	0.5139±0.0307	0.4300±**0.0156**	0.4600±0.0224	**0.2738**±0.0188
hedge	0.4612±0.0143	0.4365±0.0344	0.5332±0.0304	**0.3813±0.0058**
android	0.6651±**0.0061**	0.6651±0.0087	0.6651±0.0069	**0.6649**±0.0062
anti	0.6914±0.0138	0.6924±0.0123	0.6757±0.0088	**0.6418±0.0080**

Table 4 Total time (s) of DLR-OFP, DLP, PPD and B-ADMM for $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$

Dataset	Total time (s)			
	B-ADMM	DLP	PPD	DLR-OFP
telescope	616.0	342.1	385.1	**314.0**
wine	372.0	202.0	198.3	**102.2**
rain	1821.0	1335.8	1153.7	**791.7**
phishing	462.7	224.5	324.5	**186.6**
sensor	2796.5	2001.5	1716.0	**803.7**
trojan	3217.0	3031.0	1522.6	**1289.7**
hedge	649.1	402.4	338.4	**302.8**
android	592.4	730.1	479.3	**297.4**
anti	975.2	772.6	759.7	**442.0**
Sum of time	11501.9	9042.0	6877.6	**4530.1**

the proposed DLR-OFP algorithm, DLP [10], PPD [38] and B-ADMM algorithm with output disturbance [31] for the case of $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$. Here J is the number of distributed computers, c is the constant of the Lagrange function (5), α is the step of gradient descent, K is the maximum number of the outer cycle, M is the maximum number of the inner cycle. Meanwhile, inspired by references [10] and [35], we select the privacy budget $\epsilon = \sum_{j=1}^{J} \epsilon_j = 1.6$, $\epsilon_1 = \cdots = \epsilon_J = \epsilon^* = 0.16$. All the experimental results are based on a virtual machine with ubuntu16 in our experiment by using VMWare (2-Nuclear Processor, 8G RAM) and the distributed computer system is simulated by using the Hadoop pseudo-distributed setup.

From Tables 2-4, we can find that for $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$, all the means of accuracies of DLR-OFP algorithm are better than that of other three algorithms except android data set. The means

Table 5 Wilcoxon tests of DLR-OFP, B-ADMM, DLP and PPD for $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$.

Metrics	Comparison	R_-	R_+	Hypothesis($\alpha = 0.05$)	Selected
Accuracy	B-ADMM vs DLR-OFP	0	45	Rejected	DLR-OFP
	DLP vs DLR-OFP	2	43	Rejected	DLR-OFP
	PPD vs DLR-OFP	0	45	Rejected	DLR-OFP
MSE	B-ADMM vs DLR-OFP	39.5	5.5	No Rejected	B-ADMM
	DLP vs DLR-OFP	40	5	Rejected	DLR-OFP
	PPD vs DLR-OFP	42	3	Rejected	DLR-OFP

of MSE of DLR-OFP algorithm are better than that of other three algorithms except at most two data sets (phishing and anti). In addition, the total time of DLR-OFP algorithm is also less than that of other three algorithms.

By the means of accuracy presented in Table 2, we use Wilcoxon signed-rank test [23] to check whether there is a significant difference between the proposed DLR-OFP algorithm and B-ADMM, DLP, PPD in Table 5, respectively.

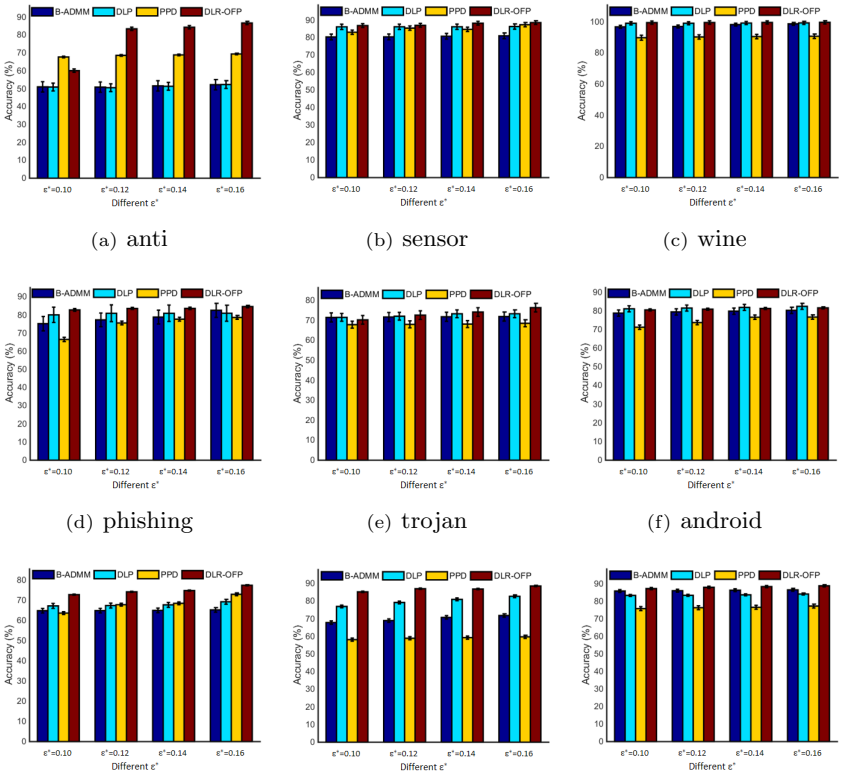
By Table 5, we can find that in terms of accuracy, there is significant difference between DLR-OFP algorithm and B-ADMM, DLP, PPD, respectively. In terms of MSE, there is also significant difference between DLR-OFP algorithm and DLP, PPD, respectively.

In addition, to have a better showing the performance of the proposed DLR-OFP algorithm, we also compare the experimental results of DLR-OFP algorithm with B-ADMM, DLP, PPD for different ϵ with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$ in Figures 1-3. All the experimental results are based on 10-times repeated experiments.

By Figures 3-5, we can find that for different ϵ^*, all the accuracies of DLR-OFP algorithm are better than that of other three algorithms, all the MSEs of DLR-OFP algorithm are better than that of other three algorithms. In addition, the total time of DLR-OFP algorithm is less than that of other three algorithms. In particular, Figure 3 shows that with the increase of ϵ^*, the accuracy of DLR-OFP algorithm has a tendency of increase. Figure 4 shows that with the increase of ϵ^*, the MSE of DLR-OFP algorithm has a tendency of decrease. Figure 5 shows that with the increase of ϵ^*, the total time of the four algorithms does not change significantly. In Section 6, we will give some discussions on the choices of J, c, α, K, M, δ_1, δ_2 for DLR-OFP algorithm.

5.2 Comparisons with FM algorithm introduced in [37]

The proposed DLR-OFP algorithm is a distributed differential privacy algorithm based on objective function perturbation and difference privacy. In [37], Xu et al. proposed FM differential privacy algorithm by considering the achieving differential privacy in vertically partitioned multiparty based on objective function perturbation and (ϵ, δ)-difference privacy. Since ϵ-difference privacy is also (ϵ, δ)-difference privacy with $\delta = 0$. In this section we compare DLR-OFP algorithm based on $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$ with



(a) anti (b) sensor (c) wine

(d) phishing (e) trojan (f) android

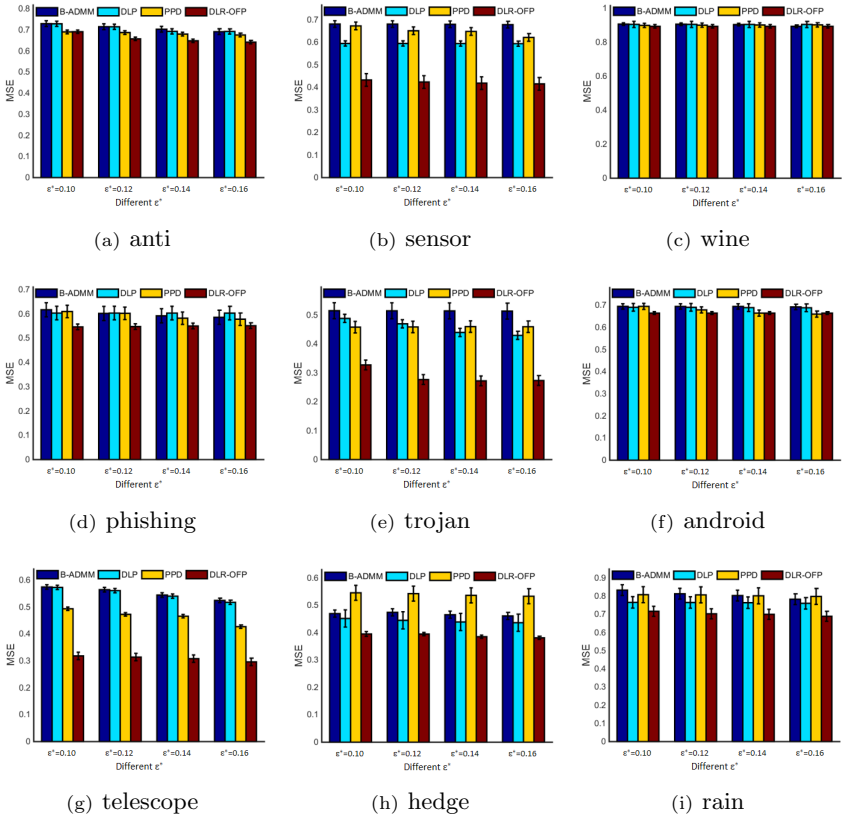
(g) telescope (h) hedge (i) rain

Fig. 3 Accuracy (%) of DLR-OFP, B-ADMM, DLP and PPD for different ϵ^* with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$

FM algorithm based on $\delta = 0$, $\epsilon = 1.6$ since in this paper the privacy budgets ϵ and ϵ^* satisfy $\epsilon = \sum_{j=1}^{J} \epsilon_j$ and $\epsilon_1 = \cdots = \epsilon_J = \epsilon^*$. FM algorithm introduced in [37] is not distributed algorithm, we replicated this algorithm on PyCharm Community. We present the experimental results on accuracy, MSE and the total time in Table 6, which are based on 10-times repeated experiments.

In Table 7, we use Wilcoxon signed-rank test [23] to check whether there is a significant difference between DLR-OFP algorithm and FM by the means of Accuracy and MSE presented in Table 6, respectively.

By Table 7, we can find that in terms of Accuracy and MSE, there is no significant difference between DLR-OFP algorithm and FM algorithm introduced in [37]. But the sum of the total time of DLR-OFP algorithm is less than that of FM algorithm.



(a) anti (b) sensor (c) wine

(d) phishing (e) trojan (f) android

(g) telescope (h) hedge (i) rain

Fig. 4 MSE of four algorithms for different ϵ^* with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$

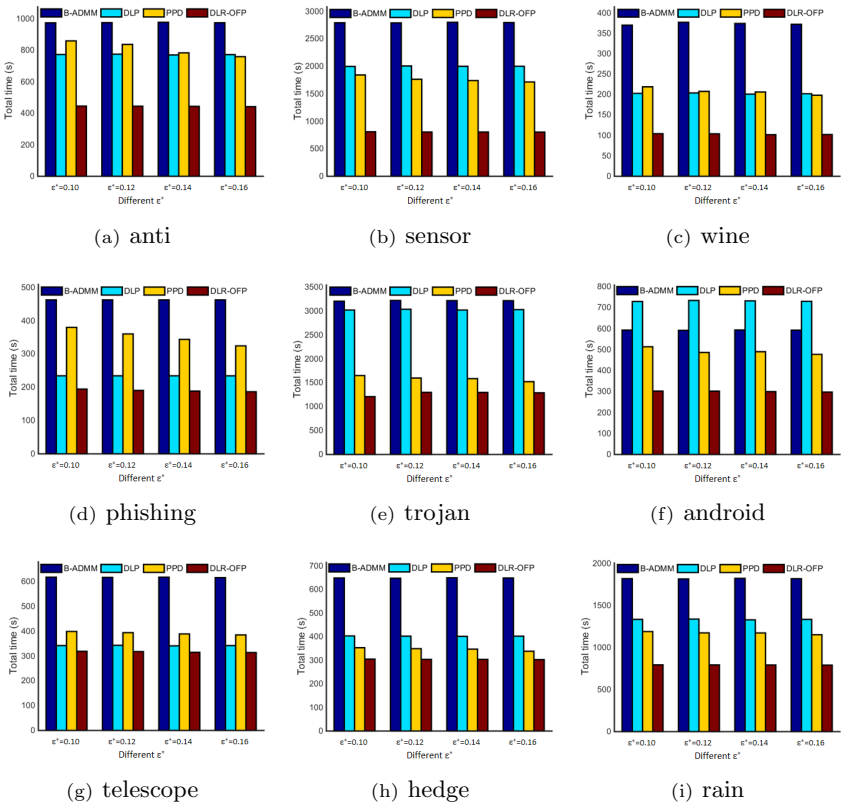
6 Discussions

In this section, we give some useful discussions on the choices of J, c, α, K, M, δ_1, δ_2 for DLR-OFP algorithm. All the experimental results are based on 10-times repeated experiments.

6.1 Choice of J

To select the number of distributed computers, we use some data sets to conduct pre-experiment on distributed ADMM algorithm without disturbance. In Table 8, we present the experimental results of distributed ADMM algorithm for different J with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$, where J is chosen from $(5, 10, 20, 50, 100)$.

By Table 8, we can find that as the number of distributed computers increases, the total time increases significantly. To have a trade-off between accuracy, MSE and the total time, we set $J = 10$ for the proposed algorithm.



(a) anti (b) sensor (c) wine

(d) phishing (e) trojan (f) android

(g) telescope (h) hedge (i) rain

Fig. 5 Total time of four algorithms for different ϵ^* with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$

6.2 Choice of c

In Algorithm 1, c is a constant of Lagrange function. For the choice of c, we use some data sets to conduct pre-experiment on DLR-OFP algorithm. In Table 9, we present the experimental results of distributed ADMM algorithm for different c with $J = 10$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$, where c is chosen from $(0.01, 0.1, 0.5, 1)$. Since the accuracy value of DLR-OFP have a tendency of decrease while the total time of DLR-OFP has a tendency of increase as c increase. To have a trade-off between accuracy and the total time, we set $c = 0.3$ for the proposed algorithm.

6.3 Choice of α

For the step size α of gradient descent, we present the experimental results of distributed ADMM algorithm for various α with $J = 10$, $c = 0.1$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$ in Table 10, where α is chosen from $(0.1, 0.3, 0.5, 1)$. Since the accuracy value of DLR-OFP have a tendency of decrease while the total time of DLR-OFP has a tendency of increase as α increase. To have a

Table 6 Experimental results of FM with $\delta = 0$, $\epsilon = 1.6$ and DLR-OFP algorithm with $c = 0.1$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\epsilon^* = 0.16$

	Accuracy(%)		MSE		Total time(s)	
telescope	**81.60**±0.97	77.45±**0.17**	0.3516±0.0559	**0.2964±0.0046**	**173.5**	314.0
wine	96.91±2.02	**99.62±0.60**	**0.7539**±0.1009	0.8943±**0.0035**	**88.7**	102.2
rain	87.95±1.56	**88.87±0.31**	**0.2243**±0.0278	0.6895±**0.0276**	899.2	**791.7**
phishing	**86.43**±0.61	84.67±**0.36**	**0.5570**±0.0793	0.5780±**0.0116**	380.5	**186.6**
sensor	88.09±1.95	**88.62±1.07**	0.9332±0.1401	**0.4159±0.0301**	1047.6	**803.7**
trojan	65.56±**1.42**	**76.41**±2.16	0.5109±0.0671	**0.2738±0.0188**	1864.0	**1289.7**
hedge	79.99±3.27	**88.62±0.64**	0.4994±0.0665	**0.3813±0.0058**	**241.3**	302.8
android	**74.33**±1.37	71.61±**1.28**	0.6667±0.0861	**0.6649±0.0062**	**284.8**	297.4
anti	**87.82**±4.35	86.75±**0.91**	**0.2434**±0.0314	0.6414±**0.0080**	**300.9**	442.0
Sum	/	/	/	/	5280.5	**4536.4**

Table 7 Wilcoxon Rank test for FM algorithm and DLR-OFP algorithm

Metrics	Comparison	R_-	R_+	Hypothesis($\alpha = 0.05$)	Selected
Accuracy	DLR-OFP vs FM	23	22	No Rejected	DLR-OFP
MSE	DLR-OFP vs FM	20	25	No Rejected	DLR-OFP

Table 8 Experimental results of distributed ADMM for different J with $c = 0.1$, $\alpha = 0.3$,$K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$

Accuracy	J=5	J=10	J=20	J=50	J=100
rain	90.15±0.25	88.87±0.31	88.51±0.30	85.77±0.89	84.30±1.85
anti	89.17±0.95	86.75±0.91	85.29±0.97	81.13±1.35	74.86±4.36
MSE	J=5	J=10	J=20	J=50	J=100
rain	0.6538±0.0187	0.6895±0.0276	0.6956±0.0270	0.7404±0.0381	0.7607±0.0726
anti	0.6282±0.0082	0.6414±0.0080	0.6498±0.0103	0.6639±0.0173	0.7022±0.0427
Total time	J=5	J=10	J=20	J=50	J=100
rain	1116	992	1363	2637	4949
anti	631	628	788	1071	2956

trade-off between accuracy and the total time, we set $\alpha = 0.3$ for the proposed algorithm.

6.4 Choice of K

In Table 11, we present the experimental results of distributed ADMM algorithm for different number K of external cycle with $J = 10$, $c = 0.1$, $\alpha = 0.3$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$, where K is chosen from $(50, 80, 100)$.

Since the accuracy value of DLR-OFP have a tendency of increase while the total time of DLR-OFP has a tendency of increase as α increase. To have a trade-off between accuracy and the total time, we set $K = 80$ for the proposed algorithm.

Table 9 Experimental results for $J = 10$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$

Accuracy	c=0.01	c=0.1	c=0.5	c=1
rain	88.38±0.61	88.87±0.31	87.70±0.58	87.66±0.41
anti	84.85±0.36	86.75±0.91	84.84±1.02	87.69±0.95
MSE	c=0.01	c=0.1	c=0.5	c=1
rain	0.6824±0.173	0.6895±0.0276	0.6589±0.0311	0.6590±0.0287
anti	0.6293±0.0064	0.6414±0.0080	0.6290±0.0109	0.6445±0.0152
Total time	c=0.01	c=0.1	c=0.5	c=1
rain	794.3	791.7	795	821.6
anti	448.7	442.0	458.3	471.2

Table 10 Experimental results of distributed ADMM algorithm for different α with $J = 10$, $c = 0.1$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$

Accuracy	α=0.1	α=0.3	α=0.5	α=1
rain	89.36±0.29	88.87±0.31	88.12±0.43	87.78±0.69
anti	87.31±0.86	86.75±0.91	85.72±1.21	85.39±1.33
MSE	α=0.1	α=0.3	α=0.5	α=1
rain	0.6939±0.0145	0.6895±0.0276	0.6802±0.0292	0.6746±0.0372
anti	0.6392±0.0084	0.6414±0.0080	0.6413±0.0149	0.6353±0.0175
Total time	α=0.1	α=0.3	α=0.5	α=1
rain	1066.2	791.7	815.4	861.1
anti	613.5	442.0	436.8	497.5

6.5 Choice of M

To select the number of internal cycles, we use some data sets to conduct pre-experiment on DLR-OFP algorithm. In Table 12, we present the experimental results of distributed ADMM algorithm for different M of internal cycles with $J = 10$, $\alpha = 0.3$, $K = 80$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$, where M is chosen from $(50, 100, 200)$. Since the accuracy value of DLR-OFP have a tendency of increase while the total time of DLR-OFP has a tendency of increase as α increase. To have a trade-off between accuracy and the total time, we set $M = 100$ for the proposed algorithm.

6.6 Choices of δ_1, δ_2

For the two constants δ_1 and δ_2, we present the experimental results of distributed ADMM algorithm for various δ_1, δ_2 with $J = 10$, $c = 0.1$, $K = 100$, $M = 200$ in Table 13 and 14, where δ_1, δ_2 are chosen from $(10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$.

Table 11 Experimental results of distributed ADMM algorithm for different K with $J = 10$, $c = 0.1$, $\alpha = 0.3$, $M = 100$, $\delta_1 = \delta_2 = 10^{-3}$

Accuracy	K=50	K=80	K=100
rain	81.35±1.25	88.87±0.31	88.79±0.40
anti	79.67±3.57	86.75±0.91	86.64±1.41
MSE	K=50	K=80	K=100
rain	0.6183±0.0405	0.6895±0.0276	0.6911±0.0290
anti	0.6285±0.0115	0.6414±0.0080	0.6426± 0.072
Total time	K=50	K=80	K=100
rain	689.4	791.7	792.5
anti	394.8	442.0	454.3

Table 12 Experimental results of distributed ADMM algorithm for different M with $J = 10$, $c = 0.1$, $\alpha = 0.3$,$K = 80$, $\delta_1 = \delta_2 = 10^{-3}$

Accuracy	M=50	M=100	M=200
rain	79.98±1.73	88.87±0.31	88.98±0.28
anti	83.25±1.46	86.75±0.91	86.73±0.83
MSE	M=50	M=100	M=200
rain	0.5985±0.0456	0.6895±0.0276	0.6936±0.0266
anti	0.6285±0.0197	0.6414±0.0080	0.6436±0.0083
Total time	M=50	M=100	M=200
rain	733	791.7	840.5
anti	406.7	442.0	476.1

Tables 13-14 show that the accuracy value of DLR-OFP have a tendency of increase while the total time of DLR-OFP has a tendency of significant increase as δ_1 and δ_2 decrease. To have a trade-off between accuracy and the total time, and inspired by references [10], we set $\delta_1 = \delta_2 = 10^{-3}$ for the proposed algorithm.

7 Conclusions

Different to the previous algorithms with distributed privacy protection based on variable perturbation, in this paper we focused on the distributed machine learning algorithms with privacy protection based on objective function perturbation. We used differential privacy to limit the risk of potential information exposure to individual records. In this article, we proposed a new distributed objective function perturbation logistic regression algorithm. Theoretically, we analyzed the theoretical advantages of our algorithm compared

Table 13 Experimental results of distributed ADMM algorithm for different δ_1 with $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 100$, $M = 200$, $\delta_2 = 10^{-3}$

Accuracy	$\delta_1 = 10^{-2}$	$\delta_1 = 10^{-3}$	$\delta_1 = 10^{-4}$	$\delta_1 = 10^{-5}$
rain	86.26±1.91	88.96±0.86	89.59±1.05	90.32±0.69
anti	84.58±2.10	86.86±1.09	87.19±0.75	87.49±0.77
MSE	$\delta_1 = 10^{-2}$	$\delta_1 = 10^{-3}$	$\delta_1 = 10^{-4}$	$\delta_1 = 10^{-5}$
rain	0.6939±0.0287	0.6875±0.0264	0.6792±0.0313	0.6584±0.0208
anti	0.6629±0.0225	0.6326±0.0109	0.6257±0.0094	0.6195±0.0071
Total time	$\delta_1 = 10^{-2}$	$\delta_1 = 10^{-3}$	$\delta_1 = 10^{-4}$	$\delta_1 = 10^{-5}$
rain	796.1	860.4	951.7	1378.2
anti	448.3	479.7	550.8	727.5

Table 14 Experimental results of distributed ADMM algorithm for different δ_2 with $J = 10$, $c = 0.1$, $\alpha = 0.3$, $K = 100$, $M = 200$, $\delta_1 = 10^{-3}$

Accuracy	$\delta_2 = 10^{-2}$	$\delta_2 = 10^{-3}$	$\delta_2 = 10^{-4}$	$\delta_2 = 10^{-5}$
rain	86.31±1.19	88.96±0.86	89.81±0.76	90.57±0.82
anti	84.40±1.68	86.86±1.09	87.11±0.93	87.83±0.67
MSE	$\delta_2 = 10^{-2}$	$\delta_2 = 10^{-3}$	$\delta_2 = 10^{-4}$	$\delta_2 = 10^{-5}$
rain	0.7027±0.0347	0.6875±0.0264	0.6806±0.0215	0.6754±0.0129
anti	0.6571±0.0200	0.6326±0.0109	0.6246±0.0116	0.6148±0.0063
Total time	$\delta_2 = 10^{-2}$	$\delta_2 = 10^{-3}$	$\delta_2 = 10^{-4}$	$\delta_2 = 10^{-5}$
rain	812.8	860.4	976.4	1187.5
anti	454.8	479.7	520.1	625.3

to these existing algorithms: We not only proved that our algorithm satisfies ϵ-differential privacy, obtained the fast convergence rate $O(1/T^2)$, which improved the corresponding results on convergence rate $O(1/T)$ obtained in [10] and [38], we but also given the estimation on the bound of the acceleration factor φ_m related to gradient descent and weight vector ω. To our knowledge, these theoretical studies are the first works of distributed privacy protection based on objective function perturbation. Experimental results demonstrated that our algorithm can efficiently process distributed storage data and protect its privacy: the proposed algorithm not only has better learning performance (e.g., accuracy, MSE), but also has less total time compared to the previously known privacy protection algorithms.

On the base of current work, there are severe problems under us consideration such as studying the learning performance of our proposed DLR-OFP algorithm for the case of multi-class classification, RBF kernels and so on.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This work is supported in part by Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province (No.CICIP2018002) and National Key Research and Development Program of China (NO.2020YFA0714200).

Appendix A

In this section, we prove our main results presented in Section 4.

Lemma 1 [10] *Let* $L(Z,\omega,V) = L(\{Z_j\}_{j=1}^J,\{\omega_j\}_{j=1}^J,\{V_j\}_{j=1}^J)$ *be the quadratic augmented Lagrangian function. Let* $c > \max\{2L_f/M', L_f M', \beta/M\}$ *be the constant in* $L(Z,\omega,V)$*. The sequences generated by the* ADMM *algorithm proposed in Section* 3 *converge to the local minimum point of* $L(Z,\omega,V)$*. Where* c *is the constant of lagrange function defined in Algorithm 1,* M *represents the minimum strictly positive eigenvalue of matrix* A^TA*,* M' *represents the minimum strictly positive eigenvalue of matrix* B^TB*, and* A*,* B *defined in equation 5 of [10].*

Proof of Theorem 1: *By Lemma 1, we come to prove that our algorithm* $\mathcal{A}$ *satisfies* ϵ*-differential privacy. For* D_{all}*, since* G *is the set of all possible outputs during the algorithm iteration.* $D = \{D_j\}_{j=1}^J$*,* $D' = \{D_j{}'\}_{j=1}^J$ *are two datasets in* D_{all}*, For any* j*,* D_j *and* $D_j{}'$ *are a pair of neighboring data sets differ on one element.*

$$D_j = \{(x_1,y_1),\ldots,(x_{n_j-1},y_{n_j-1}),(a_j,y)\},$$
$$D_j{}' = \{(x_1,y_1),\ldots,(x_{n_j-1},y_{n_j-1}),(a_j',y')\}.$$

Let $\epsilon = \sum_{j=1}^J \epsilon_j$*, we should prove*

$$\frac{P(\mathcal{A}(\|)_{k=1}^K \in G\|D)}{P(\mathcal{A}(k)_{k=1}^K \in G\|D')} \le e^{\epsilon}.$$

It follows that our algorithm $\mathcal{A}$ *provides* ϵ*-differential privacy for the variables* $Z(k)$ *and* $\omega(k)$*:*

$$\frac{P(\{Z(k),\omega(k)\}_{k=1}^K \in G\|D)}{P(\{Z(k),\omega(k)\}_{k=1}^K \in G\|D')} \le e^{\epsilon}.$$

The left side of the above inequality can be written to be

$$\begin{aligned}
&\frac{P(\{Z(k),\omega(k)\}_{k=1}^{K} \in G \| D)}{P(\{Z(k),\omega(k)\}_{k=1}^{K} \in G \| D')} \\
&= \frac{P(\{Z(0),\omega(0)\} \in G(0) \| D)}{P(\{Z(0),\omega(0)\} \in G(0) \| D')} \\
&\quad \cdot \prod_{k=1}^{K} \frac{P(\{Z(k),\omega(k)\} \in G(k) \| \{Z(r),\omega(r)\}_{r=0}^{k-1}, D)}{P(\{Z(k),\omega(k)\} \in G(k) \| \{Z(r),\omega(r)\}_{r=0}^{k-1}, D')} \\
&\overset{(a)}{=} \prod_{j=1}^{J} \prod_{k=1}^{K} \frac{P(\{Z_j(k),\omega_j(k)\} \in G_j(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j)}{P(\{Z_j(k),\omega_j(k)\} \in G_j(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j')} \\
&\overset{(b)}{=} \prod_{j=1}^{J} \prod_{k=1}^{K} \frac{P(Z_j(k) = Z_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j)}{P(Z_j(k) = Z_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j')} \\
&\quad \cdot \frac{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j)}{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j')}.
\end{aligned}$$

Equation (a) is because that at the beginning of the iteration, we have $\omega(0) = (0,\ldots,0)^T$ *and* $Z(0) = (0,\ldots,0)^T$*. Hence, for any two datasets* D *and* D'*,* $\frac{P(\{Z(0),\omega(0)\} \in G(0) \| D)}{P(\{Z(0),\omega(0)\} \in G(0) \| D')}$ *is constant* 1*. We consider the first term of Equation (b),*

$$\frac{P(Z_j(k) = Z_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j)}{P(Z_j(k) = Z_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, D_j')}.$$

According to Equation (7), we have that when the algorithm updates variable $Z_j(k)$ *to* $Z_j(k+1)$*, the update of* $Z_j(k+1)$ *has nothing to do with the data set, but only depends on* $\omega_j(k)$ *and* $V_{ji}(k)$ *in the previous* k*-th iteration. So the value of the above formula term is always* 1*.*

Now we consider the second term of Equation (b),

$$\frac{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j)}{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j')}.$$

According to the rules for updating $\omega_j(k)$*, we can rewrite the relationship between* b_j *and* ω_j *as follows:*

$$\begin{aligned}
b_j &= n_j\{(1/\alpha - ct)[(1-\gamma_{m-1})\omega_j^{m-1}(k+1) + \gamma_{m-1}\omega_j^{m-2}(k+1)] \\
&+ \frac{1}{n}\sum_{i=1}^{n_j} \frac{y_i^j x_i^j}{1 + exp(-y_i^j \omega_j^T x_i^j)} + c\sum_{j=1}^{J} E_{ij}(Z_i(k+1) + V_{ji}(k)/c) - \omega_j^m(k+1)/\alpha\},
\end{aligned}$$

where the constant t *is the degree of the* j*-th computer in the connected undirected graph.*

Given $\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1}$ *and* $Z_j^*(k)$*. For any output* ω_j^* *from our algorithm, there is an unique* b_j *that maps our input dataset to our output vector. The*

uniqueness holds because the loss and regularization functions are differentiable everywhere.

Let $g_k(\cdot, D_j)$: $R^p \to R^p$ *be the mapping from* b_j *to* $\omega_j(k)$ *of the dataset* D_j. *Let* ω_j^* *be the solution of the minimization optimization problem for the two neighboring dataset. Hence, we have*

$$\begin{aligned}
&\frac{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r), \omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j)}{P(\omega_j(k) = \omega_j^*(k) \| \{Z_j(r), \omega_j(r)\}_{r=0}^{k-1}, Z_j^*(k), D_j{}')} \\
&\overset{(c)}{=} \frac{P(b_j = g_k^{-1}(\omega_j^*(k), D_j))}{P(b_j = g_k^{-1}(\omega_j^*(k), D_j{}'))} \cdot \frac{\|det(J(g_k^{-1}(\omega_j^*(k), D_j)))\|}{\|det(J(g_k^{-1}(\omega_j^*(k), D_j{}')))\|} \\
&\overset{(d)}{=} \frac{P(b_j = g_k^{-1}(\omega_j^*(k), D_j))}{P(b_j = g_k^{-1}(\omega_j^*(k), D_j{}'))}.
\end{aligned}$$

Equation (c) is follows that in the formula, $g_k^{-1}(\omega_j^*(k), D_j)$ *is the mapping from* $\omega_j^*(k)$ *to* $b_j(k)$, $J(g_k^{-1}(\omega_j^*(k), D_j))$ *is its Jacobian matrix. Equation (d) is because that* $g_k(b_j, \cdot)$ *is a linear function, so* $\frac{\|\det(J(g_k^{-1}(\omega_j^*(k), D_j)))\|}{\|\det(J(g_k^{-1}(\omega_j^*(k), D_j{}')))\|} = 1$.

For a quantity b_j *that maps to dataset* D_j, *there exists a quantity* $b_j{}'$ *of dataset* $D_j{}'$ *such that the following statement is true:*

$$b_j - b_j{}' = \frac{n_j}{n}\Big[\frac{1}{1 + exp(y\omega_j^{*T} a)} - \frac{1}{1 + exp(y'\omega_j^{*T} a')}\Big].$$

Because $\|a\|_2 \leq 1$ *and* $\|a'\|_2 \leq 1$ *satisfying that* $\frac{1}{1+exp(y\omega_j^{*T} a)} \leq 1$, $\frac{1}{1+exp(y'\omega_j^{*T} a')} \leq 1$. *For any* ω_j^*, *we have* $\left\|b_j - b_j{}'\right\|_2 \leq 2n_j/n$. *By the triangle inequality we have* $\|b_j\|_2 - 2n_j/n \leq \left\|b_j{}'\right\|_2 \leq \|b_j\|_2 + 2n_j/n$. *Thus, we have*

$$\frac{P(b_j = g_k^{-1}(\omega_j^*(k), D_j))}{P(b_j = g_k^{-1}(\omega_j^*(k), D_j{}'))} = exp(-\frac{n\epsilon_j}{2n_j}(\|b_j\|_2 - \left\|b_j{}'\right\|_2)) \quad \leq e^{\epsilon_j}.$$

And since the second iteration, the disturbance vector has been involved in the subsequent calculation as a constant vector and no longer changes. Combining

the above conclusions, we get

$$\begin{aligned}\frac{P(\{Z(k),\omega(k)\}_{k=0}^{K}\in G\|D)}{P(\{Z(k),\omega(k)\}_{k=0}^{K}\in G\|D')} &= \prod_{j=1}^{J}\prod_{k=1}^{K}\frac{P(Z_j(k)=Z_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},D_j)}{P(Z_j(k)=Z_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},D_j{}')}\\ &\cdot\frac{P(\omega_j(k)=\omega_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},Z_j^*(k),D_j)}{P(\omega_j(k)=\omega_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},Z_j^*(k),D_j{}')}\\ &= \prod_{j=1}^{J}\frac{P(\omega_j(k)=\omega_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},Z_j^*(k),D_j)}{P(\omega_j(k)=\omega_j^*(k)\|\{Z_j(r),\omega_j(r)\}_{r=0}^{k-1},Z_j^*(k),D_j{}')}\\ &\le \prod_{j=1}^{J}e^{\epsilon_j}=e^{\sum_{j=1}^{J}\epsilon_j}=e^{\epsilon},\end{aligned}$$

which implies that Algorithm $\mathcal{A}$ satisfies ϵ-differential privacy. Then we finished the proof of Theorem 1.

Appendix B

Proof of Theorem 3: *Let m be the cycle label of gradient descent. For $m\ge 3$, we have*

$$\varphi_m=\frac{1+\sqrt{1+4\varphi_{m-1}^2}}{2}=\frac{1}{2}+\varphi_{m-1}(1+\frac{1}{4\varphi_{m-1}^2})^{\frac{1}{2}}.$$

We use Taylor expansion of the function $(1+x)^{\frac{1}{2}}$ to estimate the variable φ_m. We have

$$\frac{1}{2}+\varphi_{m-1}\big[1+\frac{1}{8\varphi_{m-1}^2}-\frac{1}{128\varphi_{m-1}^4}\big]\le\varphi_m\le\frac{1}{2}+\varphi_{m-1}\big[1+\frac{1}{8\varphi_{m-1}^2}\big].$$

Then for $m\ge 3$, we have $\frac{1}{8\varphi_{m-1}^2}-\frac{1}{128\varphi_{m-1}^4}\ge\frac{1}{9\varphi_{m-1}^2}$. Thus we have

$$\frac{1}{2}+\varphi_{m-1}+\frac{1}{9\varphi_{m-1}}\le\varphi_m\le\frac{1}{2}+\varphi_{m-1}+\frac{1}{8\varphi_{m-1}}. \tag{B.1}$$

Let u be a positive constant satisfying $\frac{1}{2}\le u\le\frac{2}{5}$, we prove Theorem 3 by mathematical induction. If $m=3$, by plugging in the actual value of the variable φ_3, inequality (12) (see Theorem 3) holds for $m=3$. For $m\ge 4$, we assume that inequality (12) holds for $m-1$. We use then the definition (9) of φ_m and inequality (B.1) to prove that inequality (12) holds for m.

According to the recursion, by using $\varphi_m=(1+\sqrt{1+4\varphi_{m-1}^2})/2$ and the estimate on the right hand side of inequality (B.1), we conclude that the right of the above inequality (12) holds. According to the recursion, by using $\varphi_m=(1+\sqrt{1+4\varphi_{m-1}^2})/2$ and the estimate on the left hand side of inequality (B.1), we conclude that the left of the above inequality (12) holds.

Proof of Theorem 4: *let* ω_j^{m*} *and* ω_j^m *be the corresponding weight vectors generated by the gradient descent method in m-th iteration with* D_1 *and* D_2 *in Algorithm 1,* $1 \le m \le M$. *Let* $P_m = \omega_j^{m*} - \omega_j^m$, $Q_m = P_m - P_{m-1}$. *We have*

$$Q_m = -(\frac{\partial L_j}{\partial \omega_j^{m*}} - \frac{\partial L_j}{\partial \omega_j^m})\Big[\sum_{j=2}^{m-1}(\prod_{i=j}^{m-1}\frac{-1+\varphi_i}{\varphi_{i+1}}) + 1\Big].$$

According to Theorem 3, we have $\prod_{i=j}^{m-1}\frac{-1+\varphi_i}{\varphi_{i+1}} \le \frac{j}{m}$, *which is because that*

$$\prod_{i=j}^{m-1}\frac{-1+\varphi_i}{\varphi_{i+1}} \le \frac{j}{j+1}\frac{j+1}{j+2}\cdots\frac{m-1}{m} = \frac{j}{m}.$$

From the above inequality, we can have $Q_m \le -mc_1$. *In fact,*

$$Q_m \le -(\frac{\partial L_j}{\partial \omega_j^{m*}} - \frac{\partial L_j}{\partial \omega_j^m})\Big[\sum_{j=2}^{m-1}\frac{j}{m} + 1\Big] \le -2c_1\Big[\sum_{j=2}^{m-1}\frac{j}{m} + 1\Big] = -mc_1.$$

From above equation, we can write the recurrence of P_m *as follows*

$$P_m = P_{m-1} + Q_m = P_{m-2} + Q_m + Q_{m-1} = \cdots = P_2 + \sum_{t=3}^{m} Q_t.$$

Use the estimation of the variable Q_m, *we have* $P_m = \omega_j^{m*} - \omega_j^m = P_2 + \sum_{t=3}^{m} Q_t \le -c_1 m^2/2$. *It follows that* $\max_{D_1,D_2}\|\omega_j^* - \omega_j\| \le c_1 M^2/2$. *Then we finished the proof of Theorem 4.*

References

[1] Latanya Sweeney. K-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 2002, 10(5): 557-570.

[2] A. Machanavajjhala, J. Gehrke, D. Kifer and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. Proceedings of IEEE 22nd International Conference on Data Engineering (ICDE'06), 2006: 24-24.

[3] Ninghui Li, Tiancheng Li, S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. 2007 IEEE 23rd International Conference on Data Engineering. IEEE, 2007: 106-115.

[4] Cynthia Dwork. Differential privacy. International Colloquium on Automata, Languages, and Programming. Springer, Berlin, Heidelberg, 2006: 1-12.

[5] Úlfar Erlingsson, Vasyl Pihur, Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. Proceedings of the ACM Conference on Computer and Communications Security. 2014: 1054-1067.

[6] Frank McSherry, Kunal Talwar. Mechanism design via differential privacy. 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07). IEEE, 2007: 94-103.

[7] Cynthia Dwork, Aaron Roth. The Algorithmic Foundations of Differential Privacy. Foundations & Trends in Theoretical Computer Science, 2014, 9: 211-407.

[8] Shuigeng Zhou, Feng Li, Yufei TAO, Xiaokui Xiao. Privacy preservation in database applications: a survey. Chinese Journal of Computers, 2009, 32(5):847-861.

[9] Ping Xiong, Tianqing Zhu, Xiaofeng Wang. A survey on differential privacy and applications. Jisuanji Xuebao/Chinese Journal of Computers, 2014, 37(1): 101-122.

[10] Puyu Wang, Hai Zhang. Distributed logistic regression with differential privacy (in Chinese). Sci. Sin. Inform., 2020, 50:1511-1528

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, 2011.

[12] Nesterov Y. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. Soviet Mathematics Doklady. 1983. Vol. 27: 372-376.

[13] Avrim Blum, Katrina Ligett, Aaron Roth. A learning theory approach to noninteractive database privacy. Journal of the ACM, 2013, 60(2): 1-25.

[14] Liyue Fan, Li Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. IEEE Transactions on knowledge and data engineering, 2013, 26(9): 2094-2106.

[15] Xiaokui Xiao, Guozhang Wang, Johannes Gehrke. Differential privacy via wavelet transforms. IEEE Transactions on knowledge and data engineering, 2010, 23(8): 1200-1214.

[16] Xiaojian Zhang, Xiaofeng Meng. Differential privacy in data publication and analysis(in Chinese). Chinese Journal of Computers, 2014, 000(004):927-949.

[17] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, Ting Yu. Differentially private spatial decompositions. 2012 IEEE 28th International Conference on Data Engineering. IEEE, 2012: 20-31.

[18] Kellaris Georgios, Papadopoulos Stavros, Xiao Xiaokui, Papadias Dimitris. Differentially private event sequences over infinite streams. Proceedings of the VLDB Endowment, 2014, 7(12): 1155-1166.

[19] Y. Xiao, L. Xiao, X. Lu, H. Zhang, S. Yu and H. V. Poor. Deep-Reinforcement-Learning-Based User Profile Perturbation for Privacy-Aware Recommendation. IEEE Internet of Things Journal, 2021, 8(6): 4560-4568.

[20] Kamalika Chaudhuri, Claire Monteleoni. Privacy-preserving logistic regression. NIPS. 2008, 8: 289-296.

[21] Kamalika Chaudhuri, Claire Monteleoni, Anand D. Sarwate. Differentially private empirical risk minimization. Journal of Machine Learning Research, 2011, 12: 1069-1109.

[22] Daniel Kifer, Adam Smith, Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. Conference on Learning Theory. JMLR Workshop and Conference Proceedings, 2012: 25.1-25.40.

[23] Wilcoxon, F. Individual comparisons by ranking methods. Biometrics Bulletin, 1945, 1(6):80-83.

[24] Martn Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang. Deep learning with differential privacy. Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016: 308-318.

[25] Hongcheng Li, Xiaoping Wu, Yan Chen. k-means clustering method preserving differential privacy in MapReduce framework(in Chinese). Journal of communication, 2016, 37(2):124-130.

[26] Yousra Abdul Alsahib S Aldeen, Mazleena Salleh, Yazan Aljeroudi. An innovative privacy preserving technique for incremental datasets on cloud computing. Journal of Biomedical Informatics, 2016:107-116.

[27] Shuo Han, Ufuk Topcu, George J. Pappas. Differentially Private Distributed Constrained Optimization. IEEE Transactions on Automatic Control, 2016, 62(1): 50-64.

[28] Tao Zhang, Quanyan Zhu. Dynamic differential privacy for ADMM-based distributed classification learning. IEEE Transactions on Information

Forensics and Security, 2016, 12(1): 172-187.

[29] Xueru Zhang, Mohammad Mahdi Khalili, Mingyan Liu. Improving the privacy and accuracy of ADMM-based distributed algorithms. International Conference on Machine Learning. PMLR, 2018: 5796-5805.

[30] Tianqing Zhu, Gang Li, Wanlei Zhou, Philip S. Yu. Differentially private data publishing and analysis: A survey. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(8): 1619-1638.

[31] Ioannidis S, Jiang Y, Amizadeh S, et al. Parallel news-article traffic forecasting with ADMM. SIGKDD Workshop on Mining and Learning from Time Series. ACM. 2016.

[32] Jie Xu, Chen Xu, Bin Zou, Yuan Yan Tang, Jiangtao Peng, Xinge You. New incremental learning algorithm with support vector machines. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 49(11): 2230-2241.

[33] Zerka F, Barakat S Walsh S Bogowicz M Leijenaar RTH Jochems A Miraglio B Townend D Lambin P. Systematic review of privacy-preserving distributed machine learning from federated databases in health care. JCO clinical cancer informatics, 2020, 4: 184-200.

[34] LP Barnes, A Ozgur. Minimax Bounds for Distributed Logistic Regression. arXiv e-prints, 2019.

[35] Zonghao Huang, Rui Hu, Yuanxiong Guo, Eric Chan-Tin, Yanmin Gong. DP-ADMM: ADMM-based distributed learning with differential privacy. IEEE Transactions on Information Forensics and Security, 2019, 15: 1002-1012.

[36] Zhanglong Ji, Xiaoqian Jiang, Shuang Wang, Li Xiong, Lucila Ohno-Machado. Differentially private distributed logistic regression using private and public data. BMC Medical Genomics Volume 7 Supplement 1, 2014.

[37] Depeng Xu, Shuhan Yuan, Xintao Wu. Achieving Differential Privacy in Vertically Partitioned Multiparty Learning. Proceedings of the 2021 IEEE International Conference on Big Data (BigData), Dec 15-18, 2021.

[38] X. Wang, H. Ishii, L. Du, P. Cheng and J. Chen. Differential Privacy-preserving Distributed Machine Learning. 2019 IEEE 58th Conference on Decision and Control (CDC), 7339-7344, 2019.

[39] Shuang Song, Kamalika Chaudhuri, Anand D. Sarwate. Stochastic gradient descent with differentially private updates. 2013 IEEE Global

Conference on Signal and Information Processing, 2013, pp. 245-248.

[40] Ninghui Li, Min Lyu, Dong Su, Weining Yang. Differential Privacy: From Theory to Practice. Morgan & Claypool, 2016.

[41] Amir Beck. The block proximal gradient method, First-Order Methods in Optimization, 2017, 25: 331-351.