FUDAN UNIVERSITY

DATA620008: OPTIMIZATION THEORY

Final Project:
One-shot Distributed Lasso Regression

Author

Zhe Li 20210980134

February 16, 2021

Contents

One-shot Distributed Lasso Regression

Zhe Li
Fudan University
School of Data Science

Date: February 16, 2021

Abstract

In this work, we develop a distributed lasso regression approximation (**DLR**) method that is able to solve the lasso regression problem on a distributed system. To handle the problem, we distribute the dataset on different worker servers. An traditional optimization algorithm (PGM, ADMM, Subgradient descent) is first conducted on each worker to get the local estimator. After that, the informations on each worker are then broadcasted to master to obtain a combined estimator by taking a weighted average of local estimators. The proposed distributed algorithm has three merits. First, the communication cost is low since it requires only one round of communication. Second, no further iterative algorithm is needed on master and hence it does not suffer from problems as initialization and non-robustness. Third, the computational complexity is much lower compared to get the global estimator using whole dataset. The code of the project can be found in (www.github.com/lkerlz/DLR). Theoretical properties are provided with respect to the estimation consistency. Lastly, the advantages of the proposed methodology are illustrated by experiments on a variety of synthetic and empirical datasets.

Keywords: Lasso; Distributed system; Optimization algorithm

1 INTRODUCTION

Data Analysis with huge datasets has become more and more popular in today's world. Consequently, the dataset must be stored and processed on many connected computer nodes, which thereafter are referred to as a distributed system (Zhu et al., 2019). Furthermore, A distributed system is a system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user.

In this work, we consider a **"master-and-worker"**-type distributed system with strong workers. As one will see, the most widely used systems, Hadoop (Hadoop, 2011) and Spark (Spark, 2018), belong to this category. In this systems architecture, the workers is mainly responsible for the computation task and do not communicate with each other directly. In constract, the master should communicate with each worker and collect all the information from workers to get the final result. Since the communication cost between the

master and workers is expensive, we should try to consider "one-shot" (OS) approach first, which requires only one round of communication. Specifically, the local worker computes the estimators in parallel and then communicate to the master to obtain an average global estimator (Chang et al., 2017; Fan et al., 2019; Liu and Ihler, 2014; Zhang et al., 2012). However "one-shot" algorithms may not achieve the best efficiency in statistical estimation in most occasions since there is not enough communications between master and workers (Smith et al., 2018). Therefore, developing algorithms that are highly efficient computationally, communicationally and statistically for distributed systems has become a problem of great interest.

The aforementioned approach is also studied for the regression problem using ℓ_1 shrinkage estimation (i.e. Lasso Regression). (Battey et al., 2015) use the MapReduce structure to solve the lasso problem and further considered distributed testing and estimation methods in a unified likelihood framework, in which a refitted estimation is used to obtain an oracle convergence rate. For the second approach, both (Wang et al., 2017) and (Jordan et al., 2018) have developed iterative algorithms to solve the sparse estimation problem, and they theoretically proved that the error bounds match the centralized estimator. Beyond the ℓ_1 shrinkage estimation, (Chen and Xie, 2014) studied a penalized likelihood estimator with more general penalty function forms in a high-dimensional setting. However, to the best of our knowledge, all of the above methods assume independent and identical samples stored by each worker, which is questionable in practice because the distributed dataset might experience great heterogeneity from worker to worker.

In this work, we aim to develop a novel methodology to address the Lasso regression. Specifically, we assume the data possessed by different workers are allowed to be heterogeneous but share the same regression relationship. The proposed method borrows the idea of the distributed least squares approximation, i.e. DLSA (Zhu et al., 2019) and can be used to handle a large class of parametric regression models on a distributed system. Specifically, let $Y \in \mathbb{R}$ be the response of interest, let X be the associated predictor, and let $\theta \in \mathbb{R}^p$ be the corresponding regression coefficient. The objective is to estimate the regression parameter θ using lasso regression on a distributed system that has one master and many strong workers. Under this setting, we propose a distributed Lasso regression (DLR) method. The key idea is as follows:

- First, we estimate the parameter θ on each worker separately by using local data on distributed workers. This can be done efficiently by using standard optimization methods (e.g., Subgradient descent, proximal gradient descent, ADMM). By assuming that the sample size of each worker is sufficiently large, the resulting estimator and its asymptotic covariance estimate should be consistent but not statistically efficient, as compared with the global estimates.
- Next, each worker passes the local estimator of θ and its asymptotic covariance estimate (a $p \times p$ matrix) to the master.
- Once the master receives all the local estimators from the workers, a weighted least squares-type objective function can be constructed. This can be viewed as a local quadratic approximation of the global log-likelihood functions. As one can expect, the resulting estimator shares the same asymptotic

covariance with the full-size MLE method (i.e., the global estimator) under appropriate regularity conditions.

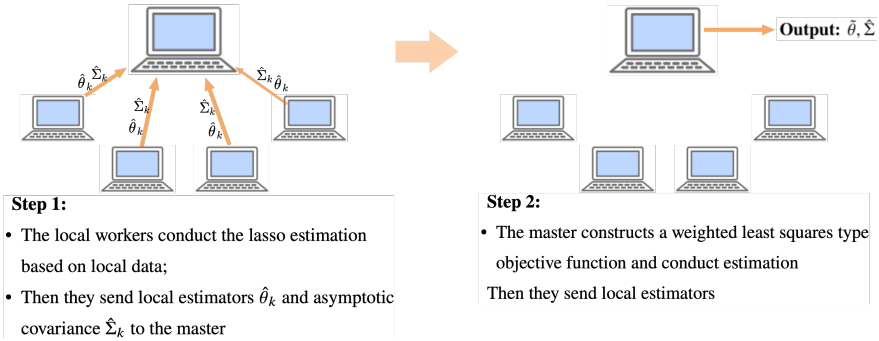

Figure 1: Illustration of the DLR method.

The major steps of the DLR method are further illustrated in Figure 1 and the remainder of this article is organized as follows. Section 2 introduces the model setting and the distributed lasso regression algorithm. We develop the theoretical properties of the estimation in Section 3. In Section 4 and 5, we study the performance of our algorithm via simulation and real data analysis. Section 6 concludes the article with a discussion.

2 DISTRIBUTED LASSO REGRESSION

2.1 Lasso Regression Model

LASSO is actually an abbreviation for "Least absolute shrinkage and selection operator", which was firstly proposed by Tibshirani (1996). Consider there are in total N observations, which are indexed as $i = 1, \ldots, N$. The ith observation is denoted as $Z_i = (X_i^\top, y_i)^\top \in \mathbb{R}^{p+1}$, where $y_i \in \mathbb{R}$ is the response of interest and $X_i \in \mathbb{R}^p$ is the corresponding covariate vector. If we denote the parameter $\beta = (\beta_1, \beta_2, \ldots, \beta_p)^\top \in \mathbb{R}^p$, then the goal of Lasso is to get the estimation of β (i.e. $\hat{\beta}^L$), which satisfies

$$\hat{\beta}^L = \arg\min \frac{1}{2N} \sum_{i=1}^{N} (y_i - X_i\beta)^2, \quad \text{s.t} \sum_{j=1}^{p} |\beta_j| \le t \tag{2.1}$$

Therefore, the lasso regression model could be written as

$$y = X\beta + \lambda \|\beta\|_1 \tag{2.2}$$

where $y \in \mathbb{R}^N$, $X \in \mathbb{R}^{N\times p}$ and $\beta \in \mathbb{R}^p$. The shrinkage parameter λ controls the value range of the variables, which means the ℓ_1-norm is a penalty term. Thus, we can provide the variable selection by changeing the value of λ in lasso regression model.

There are many algorithms to solve the lasso regression, such as **LARS** (Efron et al., 2004), **Coordinate Descent** (Wu et al., 2008). However, in this article, we use three optimization algorithms to solve the lasso regression.

2.1.1 Proximal Gradient Descent

Proximal gradient methods are a generalized form of projection used to solve non-differentiable convex optimization problems. we can rewrite equation (2.2) as an unconstrained optimization with the composite model:

$$\min_{\beta} f(\beta) = g(\beta) + h(\beta) \tag{2.3}$$

where $g(\beta) = \frac{1}{2N}\|y - X^\top\beta\|_2^2$ and $h(\beta) = \lambda\|\beta\|_1$. Since $h(\beta)$ is closed and convex function, then we can define the proximal mapping as follows:

Definition 2.1. Suppose h is a closed and convex function. The proximal mapping (prox-operator) of h is defined as

$$\text{prox}_h(x) = \underset{u}{\text{argmin}} \left(h(u) + \frac{1}{2}\|u - x\|^2 \right) \tag{2.4}$$

Following the definition 2.1, the proximal mapping for $h(]beta) = \|\beta\|_1$ can be written as

$$\text{prox}_h(\beta) = \text{sign}(\beta)\max\{|\beta| - \lambda, 0\} \tag{2.5}$$

Thus, the Proximal Gradient Descent of Lasso regression is summarized in Algorithm 1.

Algorithm 1 Proximal gradient descent of Lasso regression.

Input: Dataset $Z = (X^\top, y)^\top \in \mathbb{R}^{(p+1)\times N}$;the shrinkage parameter λ; the step size α (using constant step size); tolerance ϵ; initial point β_0

Output: Estimation of β: $\hat{\beta}$

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: update $x_{k+1} = \text{prox}_{\alpha h}(\beta) = \text{sign}(\beta)\max\{|\beta| - \alpha\lambda, 0\}$ (2.5)
3: **if** $\|x_{k+1} - x_k\|_2 < \epsilon$ **then**
4: stop
5: **end if**
6: **end for**

2.1.2 Subgradient Descent

The subgradient method is a simple algorithm for minimizing a nondifferentiable convex function (Boyd et al., 2003). we can calculate the subgradient of the loss function (2.6) directly using the definition 2.2.

$$\mathcal{L}(\beta) = \frac{1}{2N}\left\|y - X^\top\beta\right\|_2^2 + \lambda\|\beta\|_1 \tag{2.6}$$

Definition 2.2. Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function. The subdifferential $\partial f(x)$ of f at x is the set of all subgradients:

$$\partial f(x) = \left\{ g \mid g^T (y - x) \leq f(y) - f(x), \forall y \in \mathrm{dom}(f) \right\} \tag{2.7}$$

Since the $\mathrm{sign}(\beta) \in \partial \|\beta\|_1$, the descent direction of the subgradient descent algorithm can be calculated as equation (2.8) and the algorithm is summarized in Algorithm 2.

$$d^k = -\left(\frac{1}{N} X(X^\top \beta - y) + \lambda \cdot \mathrm{sign}(\beta) \right) \tag{2.8}$$

Algorithm 2 Subgradient descent of Lasso regression.

Input: Dataset $Z = (X^\top, y)^\top \in \mathbb{R}^{(p+1)\times N}$;the shrinkage parameter λ; the step size α (using constant step size); tolerance ϵ; initial point β_0

Output: Estimation of β: $\hat{\beta}$

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: update $d^{k+1} = -\left(\frac{1}{N} X(X^\top \beta_k - y) + \lambda \cdot \mathrm{sign}(\beta) \right)$ (2.8)
3: update $x_{k+1} = x_k - \alpha \cdot d^{k+1}$
4: **if** $\|x_{k+1} - x_k\|_2 < \epsilon$ **then**
5: stop
6: **end if**
7: **end for**

2.1.3 Alternating Direction Method of Multipliers

ADMM is used to solve the linearly constrained convex problem. We can regard the lasso regression as

$$\min_{\beta, z} \frac{1}{2N} \|X^\top \beta - y\|_2^2 + \mu \|z\|_1, \quad \text{s.t.} \quad \beta = z \tag{2.9}$$

To solve the problem (2.9), the Lagrange multiplier u is introduced to obtain the augmented Lagrange function (2.10). Therefore, we can use the dual ascent method to solve it.

$$L_\rho(\beta, z, u) = \frac{1}{2N} \|X^\top \beta - y\|_2^2 + \lambda \|z\|_1 + u^\top (\beta - z) + \frac{\rho}{2} \|\beta - z\|_2^2 \tag{2.10}$$

By using the dual ascent method, we can finally get the update functions of β, z and u:

- **Update** β:

$$\begin{aligned} \beta^{k+1} &= \arg\min_{\beta} \left(\frac{1}{2N} \|X^\top \beta - y\|_2^2 + \frac{\sigma}{2} \left\| \beta - z^k + u^k / \sigma \right\|_2^2 \right) \\ \Rightarrow \quad \beta^{k+1} &= \left(X^\top X + \sigma N I \right)^{-1} \left(X^\top y + \sigma N z^k - N u^k \right) \end{aligned} \tag{2.11}$$

- **Update** z:

$$\begin{aligned} z^{k+1} &= \arg\min_{z} \left(\mu \|z\|_1 + \frac{\sigma}{2} \left\| \beta^{k+1} - z + u^k / \sigma \right\|_2^2 \right) \\ \Rightarrow \quad z^{k+1} &= \mathrm{prox}_{(\mu/\sigma)\|\cdot\|_1} \left(\beta^{k+1} + u^k / \sigma \right) \end{aligned} \tag{2.12}$$

- **Update** u:

$$u^{k+1} = u^k + \gamma\sigma\left(\beta^{k+1} - z^{k+1}\right) \tag{2.13}$$

The ADMM algorithm is summarized in Algorithm 3.

Algorithm 3 ADMM of Lasso regression.

Input: Dataset $Z = (X^\top, y)^\top \in \mathbb{R}^{(p+1)\times N}$;the shrinkage parameter λ; the step size α (using constant step size);lagrange parameter σ; tolerance ϵ; initial point β_0

Output: Estimation of β: $\hat{\beta}$

Initialize z and u
for $k = 0, 1, 2, \ldots$ **do**
  update $\beta^{k+1} = (X^\top X + \sigma N I)^{-1}\left(X^\top y + \sigma N z^k - N u^k\right)$ (2.11)
  update $z^{k+1} = \text{prox}_{(\mu/\sigma)\|\cdot\|_1}\left(\beta^{k+1} + u^k/\sigma\right)$ (2.12)
  update $u^{k+1} = u^k + \gamma\sigma\left(\beta^{k+1} - z^{k+1}\right)$ (2.13)
  if $\|x_{k+1} - x_k\|_2 < \epsilon$ **then**
    stop
  end if
end for

2.2 Local Regression on Workers

Suppose the observations are distributed across K local workers. Define $\mathcal{S} = \{1, \ldots, N\}$ to be all sample observations. Decompose $S = \cup_{k=1}^{K}\mathcal{S}_k$, where $\mathcal{S}_k$ collects the observations distributed to the kth worker. Obviously, we should have $\mathcal{S}_{k_1} \cap \mathcal{S}_{k_2} = \emptyset$ for any $k_1 \neq k_2$. Define $n = N/K$ as the average sample size for each worker. Then, we assume $|\mathcal{S}_k| = n_k$ and that all n_k diverge in the same order $O(n)$. Specifically, $c_1 \leq \min_k n_k/n \leq \max_k n_k/n \leq c_2$ for some positive constants c_1 and c_2.

In order to get the local estimator of β on each worker, we can use the sub-dataset $\mathcal{S}_k$ to construct the local loss function as

$$\mathcal{L}_k(\beta) = \frac{1}{2n_k}\left\|y_k - X_k^\top\beta\right\|_2^2 + \lambda\|\beta\|_1 \tag{2.14}$$

By solving the problem (2.14) using the algorithms in section 2.1, we can finally get the lasso estimator $\hat{\beta}_k$ on each worker. Besides, we need to calculate the covariant matrix $\hat{\Sigma}_k = X_k^\top X_k$. In summary, for each worker, we need to get the local lasso estimator $\hat{\beta}_k$ and the covariant matrix $\hat{\Sigma}_k = X_k^\top X_k$.

2.3 Communication Mechanism

After getting the local estimators on workers, we then broadcast the results to master to further complete global estimation on master. In the communication step, we only broadcast the $\hat{\beta}_k$, $\hat{\Sigma}_k$ and n_k from each

worker to master. Suppose the dimension of the estimator $\hat{\beta}_k$ is p, then the total communication information is $\mathcal{O}\left(Kp(p+1)+K\right)$. This is why we call the algorithm "One-Shot" algorithm.

Algorithm 4 Distributed Lasso Regression Algorithm.

Input: Dataset $Z = (X^\top, y)^\top \in \mathbb{R}^{(p+1)\times N}$;the shrinkage parameter λ; the step size α (using constant step size);number of workers K; tolerance ϵ; initial point β_0

Output: Estimation of β: $\tilde{\beta}$

Step 1 Lasso Regression on Worker

Step 1.1 Calculate the covariant matrix $\hat{\Sigma}_k = X^\top X$.

Step 1.2 Conduct proximal gradient descent algorithm (1), subgradient descent algorithm. (2) or ADMM algorithm (3) to obtain $\hat{\beta}_k$.

Step 2 Broadcast Informations to Master Server

Step 2.1 Broadcast the n_k,$\hat{\beta}_k$ and $\hat{\Sigma}_k$ to master server.

Step 3 Combination on Master Server

Step 3.1 Use (2.17) to obtain the global estimator $\tilde{\beta}$.

2.4 Combination on Master Server

There is not complex algorithm on the master server, the only thing that master server need to do is combining all the information from workers to get the global estimator: $\tilde{\beta}$.

It's easy to see that the global loss function $\tilde{L}(\beta)$ can be decomposed as (2.15) using Taylor's expansion techniques.

$$\begin{aligned}\mathcal{L}(\beta) &= N^{-1}\sum_{k=1}^{K}\sum_{i\in\mathcal{S}_k}\mathcal{L}\left(\beta;Z_i\right) = N^{-1}\sum_{k=1}^{K}\sum_{i\in\mathcal{S}_k}\left\{\mathcal{L}\left(\beta;Z_i\right)-\mathcal{L}\left(\widehat{\beta}_k;Z_i\right)\right\}+C_1\\ &\approx N^{-1}\sum_{k=1}^{K}\sum_{i\in\mathcal{S}_k}\left(\beta-\widehat{\beta}_k\right)^\top\ddot{\mathcal{L}}\left(\widehat{\beta}_k;Z_i\right)\left(\beta-\widehat{\beta}_k\right)+C_2\end{aligned} \tag{2.15}$$

where the last equation uses the fact that $\dot{\mathcal{L}}_k\left(\widehat{\beta}_k\right)=0$, and C_1, C_2 are some constants. Then, minimizing the $\mathcal{L}(\beta)$ is equal to minimize the $\widetilde{\mathcal{L}}(\beta)$ in (2.16).

$$\begin{aligned}\widetilde{\mathcal{L}}(\beta) &= N^{-1}\sum_{k}\left(\beta-\widehat{\beta}_k\right)^\top\left\{\sum_{i\in\mathcal{S}_k}\ddot{\mathcal{L}}\left(\widehat{\beta}_k;Z_i\right)\right\}\left(\beta-\widehat{\beta}_k\right)\\ &\overset{\text{def}}{=}\sum_{k}\left(\beta-\widehat{\beta}_k\right)^\top\alpha_k\widehat{\Sigma}_k^{-1}\left(\beta-\widehat{\beta}_k\right)\end{aligned} \tag{2.16}$$

where $\alpha_k = n_k/N$. Finally, we can get the global estimator $\tilde{\beta}$ by (2.17)

$$\tilde{\beta} = \arg\min_{\beta}\widetilde{\mathcal{L}}(\beta) = \left(\sum_{k}\alpha_k\widehat{\Sigma}_k^{-1}\right)^{-1}\left(\sum_{k}\alpha_k\widehat{\Sigma}_k^{-1}\widehat{\beta}_k\right) \tag{2.17}$$

The whole steps of DLR algorithm is summarized in Algorithm 4.

3 THEORETICAL PROPERTIES

Suppose the parameter of interest is given by $\beta \in \mathbb{R}^p$. Let $\mathcal{L}(\beta; Z)$ in (2.17) be a plausible twice-differentiable loss function and the whose global minimizer $\widehat{\beta} = \arg\min \mathcal{L}(\beta; Z)$. It is assumed that $\tilde{\beta}$ admits the following asymptotic rule

$$\sqrt{N}\left(\widehat{\beta} - \beta_0\right) \to_d N(0, \Sigma) \tag{3.1}$$

for some positive definite matrix $\Sigma \in \mathbb{R}^{p\times p}$ as $N \to \infty$. Correspondingly, define the local loss function in the kth worker as $\mathcal{L}_k(\beta) = n_k^{-1} \sum_{i\in\mathcal{S}_k} \mathcal{L}(\beta; Z_i)$, whose minimizer is $\widehat{\beta}_k = \arg\min_\beta \mathcal{L}_k(\beta)$. We assume that

$$\sqrt{n_k}\left(\widehat{\beta}_k - \beta_0\right) \to_d N(0, \Sigma_k) \tag{3.2}$$

as $n_k \to \infty$ for a positive definite matrix Σ_k. Besides, given Given other conditions in Zhu et al. (2019), we can establish the asymptotic properties of global estimator $\tilde{\beta}$ using Algorithm 4 in the following Proposition 3.1 and Theorem 3.1.

Proposition 3.1. *Assume Condtions (3.1), (3.2) and other conditions in Zhu et al. (2019). Then we have*

$$\sqrt{N}\left(\tilde{\beta} - \beta_0\right) = V(\beta_0) + B(\beta_0) \tag{3.3}$$

with $\operatorname{cov}\{V(\beta_0)\} = \Sigma$ *and* $B(\beta_0) = O_p(K/\sqrt{N})$, *where* $\Sigma = \left(\sum_{k=1}^K \alpha_k \Sigma_k^{-1}\right)^{-1}$.

The proof of Proposition 3.1 is given in Zhu et al. (2019). Consequently, if the local sample size is sufficiently large, the bias should be sufficiently small, and thus, the estimation efficiency will be the same as the global estimator.

Theorem 3.1. *(*Global Asymptotic Normality*) If further assume* $n/N^{1/2} \to \infty$. *Then, we have* $\sqrt{N}\left(\tilde{\beta} - \beta_0\right) \to_d N(0, \Sigma)$, *which achieves the same asymptotic normality as the global estimator* $\widehat{\beta}$.

It can be concluded that we should require the local sample size to be of order larger than $\sqrt{N}$, which is easy to satisfy in practice.

4 NUMERICAL STUDIES

4.1 Simulation Models and Performance Measurements

In order to demonstrate the performance of our DLR algorithm, we conduct experiments using synthetic datasets under three scenarios. The generating mechanism of the simulation model is

$$y = X\beta_0 + \epsilon \tag{4.1}$$

The random experiments are repeated for $R = 500$ times for a reliable evaluation. For the rth replication, denote $\widehat{\beta}^{(r)}$ and $\tilde{\beta}^{(r)}$ as the global lasso estimator and global DLR estimator, respectively. To measure the estimation efficiency, we calculate the root mean square error (RMSE) for the jth estimator as $\text{RMSE}_{\tilde{\beta},j} = \left\{ R^{-1} \sum_r \left\| \tilde{\beta}_j^{(r)} - \beta_{0j} \right\|^2 \right\}^{1/2}$. The RMSE for the global estimator $\widehat{\beta}$ can be defined similarly. In order to measure the sparse discovery accuracy, we calculate the average model size as $\text{MS} = R^{-1} \sum_r \left| \widehat{\mathcal{M}}^{(r)} \right|$, where $\text{MS} = R^{-1} \sum_r \left| \widehat{\mathcal{M}}^{(r)} \right|$ is the set of selected variables in the rth replication.

4.2 Simulation Results

Scenario 1 (Shrinkage Parameter) First, we investigate the role of shrinkage parameter λ on the numerical performances. Particularly, we let $N = 10000$ and $K = 5$. In addition, the true value of β is given as $\beta_0 = (3, 0, 2, 1.5, 0, 4, 1, 0)^\top$.The performances are evaluated for λ from 0 to 10000 for the three algorithms in Section 2.1. The global estimator $\widehat{\beta}$ is also obtained using coordinate descent method on the whole dataset for comparison.

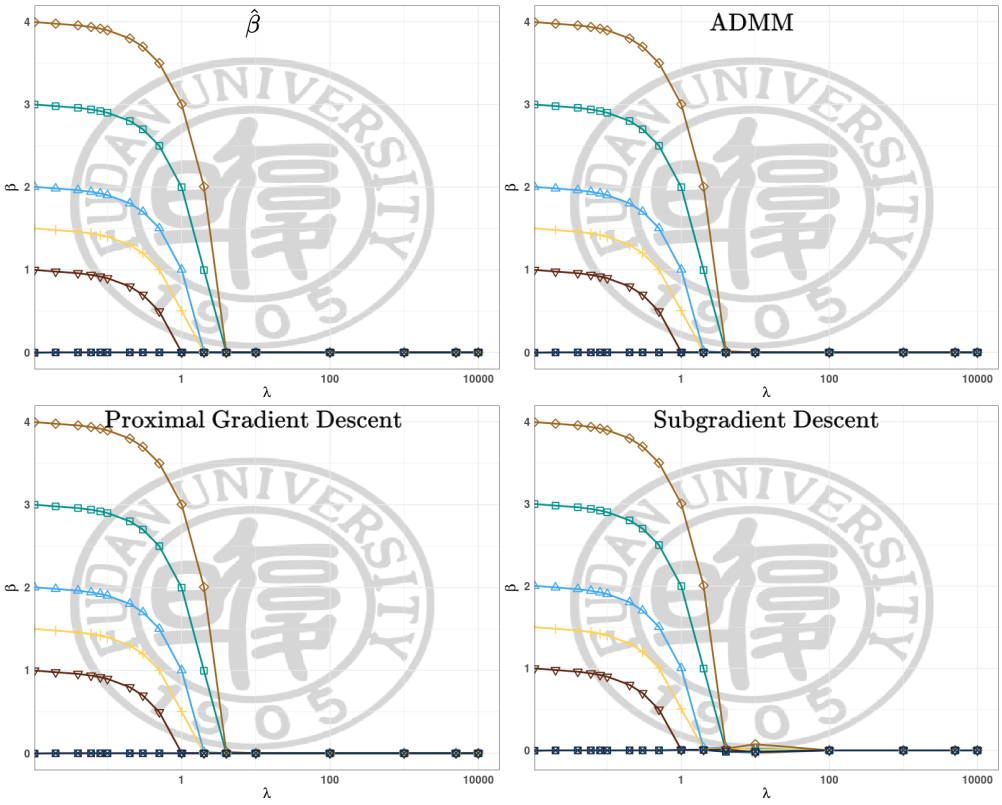

Figure 2: The estimation of β using four different algorithms

As shown in Figure 2, all estimation values of β converges to zero as λ grows. Besides, we can see the global estimator $\tilde{\beta}$ obtained by the distributed algorithm (i.e., Algorithm 4) is very closed to the global estimator $\widehat{\beta}$ on the whole dataset, which means our algorithm is accurate and effective.

Scenario 2 (Sample Size) In this scenario, we observe how the RMSE changes with respect to the sample size. Accordingly we fix $K = 5$, $\lambda = 0.001$ and vary the sample size from 500 to 10000. In the meanwhile, we set the $\beta_0 = (3, 2, 1.5, 4, 1)^\top$ and the method we used on worker is **Proximal Gradient Descent**. The result is summarized in Figure 3.

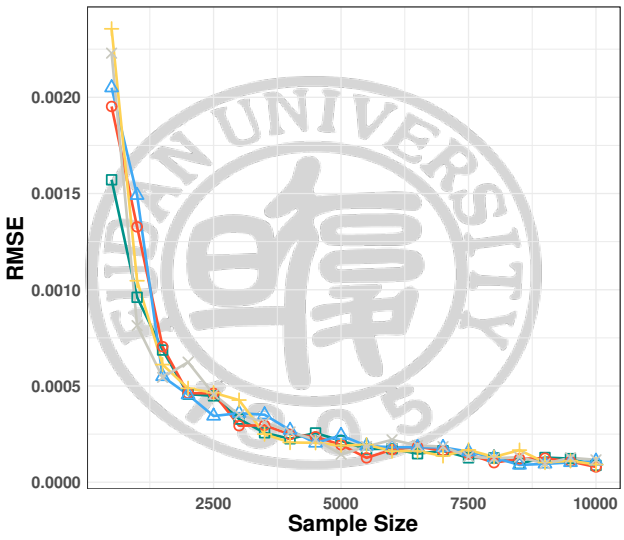

Figure 3: RMSE of each dimension in β under different N

As shown in Figure 3, the RMSE converges to zero as the sample size grows, which corroborates with our theoretical findings in Theorem 3.1. Besides, we can see the converge rate is very fast, when the sample size equals to 8000 (i.e. each worker has 1600 points), the RMSE is less than 10^{-4}.

Scenario 3 (Worker Number) In this setting, we verify the worker number effect on the finite sample performances. First, we fix $N = 1000$, $\lambda = 0.001$. Then, we set the $\beta_0 = (3, 2, 1.5, 4, 1)^\top$ and vary K from 5 to 100. As the same in Scenario 2, we use **Proximal Gradient Descent** on each worker to get $\hat{\beta}_k$. The result is summarized in Figure 4.

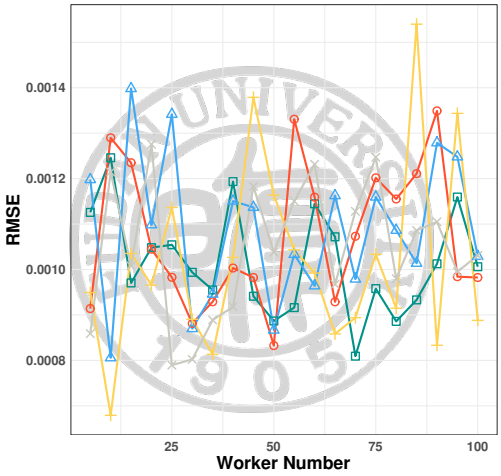

Figure 4: RMSE of each dimension in β under different K

As shown in Figure 4, worker number has no effect on the RMSE. This is because each worker contributes the weight $\alpha_k \hat{\Sigma}_k^{-1}$, which makes the global estimator $\tilde{\beta}$ consistent even if the sample size on each worker is small. Therefore, even if there are many workers in the distributed system, our algorithm could still get the consistent global estimator $\tilde{\beta}$. However, the effect of worker number needs to explore in the further study.

5 APPLICATION TO USEDCAR DATA

5.1 Data Description

For illustration purposes, we study a real-world dataset. Specifically, the dataset is the U.S. USERCAR Dataset, which is available at tianchi.aliyun.com. It contains detailed used car trading information from a website. The task is to predict the price of the used car with a regression model. Each sample in the data corresponds to one used car record, which consists of many variables such as "fuel tpye", "seller", "price". The complete variable information is described in Table .The total sample size is 150 thousand observations.

5.2 The Spark System

To demonstrate our method, we set up a Spark-on-YARN cluster on three servers . This is a standard industrial-level architecture setup for a distributed system. The system consists of one master node and two worker nodes. Each node contains 32 virtual cores, 64 GB of RAM. The dataset is stored on the Hadoop data file system (HDFS).

5.3 The DLR Method

For the USEDCAR dataset, we perform our DLR algorithm on the Spark and finally get the results. Besides, we compare our result with the solver in **"sklearn"** package of **Python** and find that there is a big difference between them. This is due to the bad initial value, how to get a good initial value is an interesting problem and it will be discussed in the further study.

6 CONCLUDING REMARKS

In this work, we propose a distributed lasso regression (DLR) algorithm to tackle regression task in large dataset. The proposed DLR algorithm has three merits. First, the communication cost is low since it requires only one round of communication. Second, no further iterative algorithm is needed on master and hence it does not suffer from problems as initialization and non-robustness. Third, the computational complexity is much lower compared to get the global estimator using whole dataset.

To conclude the article, we provide several topics for future studies. First, better mechanisms can be designed to get a better initial point. This enables us to obtain more accurate estimation of the pseudo centers

and yields better clustering results. Next, it is interesting to discuss the effect of the worker number. Third, it is important to explore how to reduce the iterations of the algorithm on each worker.

Acknowledgment

Thanks to Prof. Jiang for his teaching and guidance over the past semester. As a student of school of big data, this course gives us a very intuitive understanding of optimization algorithm. I believe the knowledge of this course will be of great help to our future research. At the same time,I would like to thank all the teaching assistants for correcting ours homework carefully and patiently answering our various questions, which has greatly helped our study of this course.

References

Battey, H., Fan, J., Liu, H., Lu, J., and Zhu, Z. (2015), "Distributed estimation and inference with statistical guarantees," *arXiv preprint arXiv:1509.05457*.

Boyd, S., Xiao, L., and Mutapcic, A. (2003), "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004, 2004–2005.

Chang, X., Lin, S.-B., and Zhou, D.-X. (2017), "Distributed semi-supervised learning with kernel ridge regression," *The Journal of Machine Learning Research*, 18, 1493–1514.

Chen, X. and Xie, M.-g. (2014), "A split-and-conquer approach for analysis of extraordinarily large data," *Statistica Sinica*, 1655–1684.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004), "Least angle regression," *Annals of statistics*, 32, 407–499.

Fan, J., Wang, D., Wang, K., and Zhu, Z. (2019), "Distributed estimation of principal eigenspaces," *Annals of statistics*, 47, 3009.

Hadoop, A. (2011), "Apache hadoop," *URL http://hadoop. apache. org*.

Jordan, M. I., Lee, J. D., and Yang, Y. (2018), "Communication-efficient distributed statistical inference," *Journal of the American Statistical Association*.

Liu, Q. and Ihler, A. (2014), "Distributed estimation, information loss and exponential families," *arXiv preprint arXiv:1410.2653*.

Smith, V., Forte, S., Chenxin, M., Takáč, M., Jordan, M. I., and Jaggi, M. (2018), "CoCoA: A general framework for communication-efficient distributed optimization," *Journal of Machine Learning Research*, 18, 230.

Spark, A. (2018), "Apache spark," *Retrieved January*, 17, 2018.

Tibshirani, R. (1996), "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, 58, 267–288.

Wang, J., Kolar, M., Srebro, N., and Zhang, T. (2017), "Efficient distributed learning with sparsity," in *International Conference on Machine Learning*, PMLR, pp. 3636–3645.

Wu, T. T., Lange, K., et al. (2008), "Coordinate descent algorithms for lasso penalized regression," *Annals of Applied Statistics*, 2, 224–244.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. (2012), "Communication-efficient algorithms for statistical optimization," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, pp. 6792–6792.

Zhu, X., Li, F., and Wang, H. (2019), "Least Squares Approximation for a Distributed System," *arXiv preprint arXiv:1908.04904*.