 CMC, vol.0, no.0, pp.1-16, 2019

A Distributed ADMM Approach for Collaborative Regression Learning in Edge Computing

Yangyang Li[1], Xue Wang[2], Weiwei Fang[2,*], Feng Xue[2], Hao Jin[1], Yi Zhang[1] and Xianwei Li[3]

Abstract: With the recent proliferation of Internet-of-Things (IoT), enormous amount of data are produced by wireless sensors and connected devices at the edge of network. Conventional cloud computing raises serious concerns on communication latency, bandwidth cost, and data privacy. To address these issues, edge computing has been introduced as a new paradigm that allows computation and analysis to be performed in close proximity with data sources. In this paper, we study how to conduct regression analysis when the training samples are kept private at source devices. Specifically, we consider the lasso regression model that has been widely adopted for prediction and forecasting based on information gathered from sensors. By adopting the Alternating Direction Method of Multipliers (ADMM), we decompose the original regression problem into a set of subproblems, each of which can be solved by an IoT device using its local data information. During the iterative solving process, the participating device only needs to provide some intermediate results to the edge server for lasso training. Extensive experiments based on two datasets are conducted to demonstrate the efficacy and efficiency of our proposed scheme.

Keywords: Internet-of-Things(IoT), edge computing, ADMM, lasso.

1 Introduction

In recent years, the fast penetration of Internet-of-Things (IoT) devices with various embedded sensors have significantly changed the way of information gathering, processing and sharing. Generally, it is impractical to run computation intensive applications at the IoT devices, since these devices are often constrained by on-board resources and battery capacity. This motivates development of IoT cloud platforms allowing offloading computation and analysis tasks to a resourceful centralized cloud [Truong and Dustdar (2015)].

[1] Innovation Center & Mobile Internet Development and Research Center, China Academy of Electronics and Information Technology, Beijing, 100041, China.
[2] School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044, China.
[3] Global Information and Telecommunication Institute, Waseda University, Tokyo 169-0051, Japan.
[*] Corresponding Author: Weiwei Fang. Email: wwfang@bjtu.edu.cn.

Nevertheless, the cloud-based solution introduces unpredictably long latency for data communication through the wide area network, and incurs huge additional bandwidth occupation that may be beyond the capabilities of today's Internet [Ha, Pillai, Lewis et al. (2013)]. Furthermore, it would also bring about a number of privacy threats and security challenges [Zhou, Cao, Dong et al. (2017)]. Therefore, it is more preferable to move computation and analysis to a close proximity of the IoT devices, i.e., to the edge of the network. It is envisioned that edge computing would be a promising supplement to cloud computing [Shi, Cao, Zhang et al. (2016)], and make as much of an impact on human society as the latter.

By edge computing, it is feasible to conduct collaborative machine learning [Portelli and Anagnostopoulos (2017)] in real-time on site, obtaining useful information from data collected by a variety of IoT devices. For instance, the roadside base-station can use regression analysis to forecast short-term traffic flow by analyzing data originated from proximate GPS-enabled vehicles, video cameras and roadway sensors [Zhou, Cao, Dong et al. (2017); Shi, Cao, Zhang et al. (2016); Xi, Sheng, Sun et al. (2018)]. Another good example is equipment maintenance, which uses multi-sensor information (e.g., temperature, sound and vibration) to construct classifiers for fault detection and diagnosis [Kwon, Hodkiewicz, Fan et al. (2016)]. In such systems, edge analytics is usually performed in a centralized fashion, i.e., each involved device sends its own data samples to a dedicated edge server for training and building learning models. However, this centralized solution suffers from three drawbacks. Firstly, many machine learning algorithms require to solve a particular convex optimization problem. According to previous studies [Dhar, Yi, Ramakrishnan et al. (2015); Boyd, Parikh, Chu et al. (2011)], the traditional centralized solver does not scale well with increasing volume of data. Secondly, not all edge servers are as resourceful as cloud servers to run sophisticated tools for single-node in-memory analytics [Dhar, Yi, Ramakrishnan et al. (2015); Ismail, Goortani, Karim et al. (2015)]. Thirdly, IoT-generated data may contain private and sensitive information (e.g., healthy state of wearable users) that should not be directly exposed to the edge server or other devices [Zhou, Cao, Dong et al. (2017); Gong, Fang and Guo (2016)]. To tackle these challenges, it is desirable that the learning solution for edge computing can jointly take scalability, performance and privacy issues into consideration.

In this paper, we are particularly interested in lasso (i.e., least absolute shrinkage and selection operator [Tibshirani (1996)]), a classic linear regression technique that combines regularization and variable selection together for prediction and forecasting. This technique has already been used in a lot of IoT applications, e.g., battery availability prediction for IoT devices [Longo, Mateos and Zunino (2018)], and internal temperature forecast for smart buildings [Spencer, Alfandi and Al-Obeidat (2018)]. Specifically, we develop a distributed, collaborative learning solution that utilizes sampling data from multiple IoT devices for training lasso regression models. Based on the Alternating Direction Method of Multiplies (ADMM) [Boyd, Parikh, Chu et al. (2011)], the proposed scheme naturally decomposes the target optimization problem of lasso into small sub-problems that can be solved by each participating device using its local data. Unlike centralized solutions [Lon-

go, Mateos and Zunino (2018); Spencer, Alfandi and Al-Obeidat (2018)], in our scheme the edge server only needs to collect locally trained intermediate parameters from IoT devices, and performs a simple aggregate operation to obtain the objective lasso model. The edge server and IoT devices work in such a collaborative way for multiple iterations until the lasso model converges. We have conducted extensive experiments based on two datasets with various system configurations. The experimental results show that our scheme quickly converges to near-optimal performance in a few tens of iterations. As compared to other benchmark solutions, it performs well in terms of efficiency and scalability, while obtaining a resulting lasso model with modest accuracy.

The rest of this paper is organized as follows. A brief review of existing work is presented in Section 2. Section 3 describes the system model and derives the problem formulation. In Section 4, we elaborate and discuss the proposed ADMM-based algorithm. Section 5 illustrates and discusses simulation results. Finally, we conclude this paper in Section 6.

2 Related work

Traditional machine learning algorithms [Tibshirani (1996); Spencer, Alfandi and Al-Obeidat (2018)] and tools [Boyd, Parikh, Chu et al. (2011)] are implemented using a fully centralized architecture, which requires a dedicated server with powerful computation capability and huge amount of memory. However, they fail to scale well with increasing size of data in the big data era. To address this challenge, various approaches have leveraged distributed data-parallel platforms to develop distributed machine learning libraries, such as Apache Mahout and Spark MLlib [Dhar, Yi, Ramakrishnan et al. (2015)]. These platforms and libraries can significantly speed up the large-scale data analytics by coordinating the operations of multiple servers [Richter, Khoshgoftaar, Landset et al. (2015)]. Nevertheless, they are not suitable to be applied for model learning in edge computing, due to resource constraints [Shi, Cao, Zhang et al. (2016)] and privacy concerns [Zhou, Cao, Dong et al. (2017)].

Considering that convex optimization is at the core of most machine learning algorithms, recent years have seen a number of distributed learning algorithms based on iterative methods [Dhar, Yi, Ramakrishnan et al. (2015)], which use successive approximations to come closer to the optimal solutions in each iteration. Among them, Stochastic Gradient Descent (SGD) is the most influential technique for solving linear prediction problems, e.g., logistic regression. Zinkevich et al. [Zinkevich, Weimer, Smola et al. (2010)] propose the first parallel SGD algorithm that brings very little overhead on both I/O and communication. Meeds et al. [Meeds, Hendriks, Al Faraby et al. (2015)] develop a SGD-based javascript library that enables ubiquitous compute devices to run training algorithms in web browsing environments. With similar motivation to ours, McMahan et al. [McMahan, Moore, Ramage et al. (2017)] study the SGD-based distributed model training by iteratively aggregating locally trained parameters from edge devices. Although very efficient and easy to implement, SGD algorithms generally have a slow convergence rate due to their stochastic nature [Dhar, Yi, Ramakrishnan et al. (2015)]. How to accelerate the convergence of SGD still remains as a challenging issue [Allen-Zhu (2017)].

Recent research progresses on ADMM [Boyd, Parikh, Chu et al. (2011)] make it a competitive technique for solving distributed optimization and statistical learning problems. ADMM integrates the fast convergence characteristics of the multipliers method with the decomposability of the dual ascent approach, and can quickly converge to modest accuracy. This technique can be used to solve many supervised learning algorithms on regression and classification [Dhar, Yi, Ramakrishnan et al. (2015)]. For example, Zhang et al. [Zhang, Lee and Shin (2012)] propose a distributed linear classification algorithm to solve the support vector machine problem. Gong et al. [Gong, Fang and Guo (2016)] design a privacy-preserving scheme for training a logistic regression model based on distributed data from multiple users. However, the private aggregation mechanism used in Gong et al. [Gong, Fang and Guo (2016)] are proved to be inefficient and not suitable for resource-constrained devices [Joye and Libert (2013)]. The work most relevant to ours is Bazerque et al. [Bazerque, Mateos and Giannakis (2010)], in which a consensus-based distributed algorithm is developed for in-network lasso regression. This algorithm is designed for networked systems with no central coordination, e.g., wireless sensor network. A device needs to communicate with its one-hop neighbors frequently for updating intermediate parameters, resulting in heavy communication overhead and low convergence rate in large networks.

3 System model and problem formulation

3.1 System model

We consider edge systems consisting of an edge server, and N homogeneous IoT devices performing a common sensing task. The IoT device continuously generates sensory data, and transforms the raw data in a certain time duration into a feature vector. Each feature vector consists of more than one predictor variables, and corresponds to an response variable. The edge server is responsible for performing data analysis and modelling the relationship between feature vectors and response variables. The resulting model is learned and built in a collaborative fashion at the edge server based on data samples from all participating devices.

The prediction model considered in this work is a lasso regression model [Tibshirani (1996)], which is a classic linear regression technique widely used for prediction and forecasting. Here we briefly introduce its basics. Given input training data set $\{(\boldsymbol{x_i}, y_i), i = 1, \ldots, N\}$, where $\boldsymbol{x_i} \in \mathbb{R}^n$ denotes a feature vector and $y_i \in \mathbb{R}$ denotes the corresponding response variable, the lasso model solves the following optimization problem:

$$\min \quad \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i + b - y_i)^2 + \lambda \left\|\boldsymbol{w}\right\|_1, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector, $b \in \mathbb{R}$ is the intercept, and $\lambda > 0$ is the regularization parameter. With the trained lasso model $(\boldsymbol{w}, b)$ and a given feature vector $\boldsymbol{x} \in \mathbb{R}^n$, we can estimate the value of response variable $\hat{y}$ as follows:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b. \tag{2}$$

Although very effective in practice, current lasso implementations [Longo, Mateos and Zunino (2018); Kwon, Hodkiewicz, Fan et al. (2016)] generally require that the participating devices contribute their native data to the edge server for model training. This would cause privacy leakage problems [Zhou, Cao, Dong et al. (2017)], as the native data could reveal private or sensitive information about device users. We assume that standard network security mechanisms [Zhou, Cao, Dong et al. (2017)], such as encryption and authentication, are applied to protect data storage and network communication of IoT devices from outsider attacks. Nevertheless, the edge server may not be trustworthy, and can still be a potential source of information leakage.

3.2 Problem formulation

Based on the basic model introduced in (1), we investigate the problem of collaborative lasso learning in edge computing systems. Specifically, it is assumed that each IoT device $i \in \{1, \ldots, N\}$ generate a set of data samples $D_i = \{(\boldsymbol{x}_{ij}, y_{ij}), j = 1, \ldots, M_i\}$, where $\boldsymbol{x}_{ij} \in \mathbb{R}^n$ denotes a feature vector, $y_{ij} \in \mathbb{R}$ denotes the response variable of $\boldsymbol{x}_{ij}$, and M_i denotes how many data samples are contributed by i. Then, we hope to find a distributed solution to address the following minimization problem:

$$\min \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M_i} (\boldsymbol{w}^T \boldsymbol{x}_{ij} + b - y_{ij})^2 + \lambda \left\| \boldsymbol{w} \right\|_1, \tag{3}$$

where $\boldsymbol{w} \in \mathbb{R}^n$, and $b \in \mathbb{R}$.

4 Distributed lasso learning via ADMM

According to the problem formulation and our analysis presented above, it is inappropriate to take the centralized approaches [Kwon, Hodkiewicz, Fan et al. (2016); Spencer, Alfandi and Al-Obeidat (2018)] as a solution in edge computing scenarios. A desirable solution should take the requirements on scalability, performance and privacy into consideration. This motivates us to develop an efficient and scalable scheme that enables collaborative lasso learning in a distributed manner.

4.1 A briefing on ADMM

The proposed scheme is based on ADMM, which follows a decomposition-coordination process. The target optimization problem is firstly decomposed into a set of small sub-problems, and then the solutions to these sub-problems are coordinated to obtain the global optimal result. Specifically, ADMM solves optimization problems taking the following forms:

$$\begin{aligned} \min \quad & f(x)+g(y) \\ \text{s.t.} \quad & Ax+By=C, \\ & x\in X, y\in Y, \end{aligned} \tag{4}$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, f and g are two convex functions, $A \in \mathbb{R}^{l\times n}$ and $B \in \mathbb{R}^{l\times m}$ are relation matrices, $C \in \mathbb{R}^l$ is a relation vector, and X and Y are non-empty convex subsets of R^n and R^m, respectively.

The augmented Lagrangian of problem (4) is formed by adding a l_2 norm penalty to the objective function:

$$\begin{aligned} L_\rho(x,y,z) =& f(x)+g(y)+z^T(Ax+By-C) \\ &+\frac{\rho}{2}\|Ax+By-C\|_2^2, \end{aligned} \tag{5}$$

where $z \in \mathbb{R}^l$ is the dual variable, and ρ is a positive penalty parameter.

Then, the problem (4) is solved in a iterative fashion, by updating x, y, z sequentially and alternatively. Specifically, in the t-th iteration, the updates of variables are as follows:

$$\begin{aligned} x^{t+1} &= \arg\min_x L_\rho(x,y^t,z^t), \\ y^{t+1} &= \arg\min_y L_\rho(x^{t+1},y,z^t), \\ z^{t+1} &= z^t+\rho(Ax^{t+1}+By^{t+1}-C). \end{aligned} \tag{6}$$

The proofs on optimality and convergence of ADMM have been given in [Bertsekas and Tsitsiklis (1989)]. Besides, it is revealed by empirical studies that this technique often achieves an acceptable solution with modest accuracy after dozens of iterations [Boyd, Parikh, Chu et al. (2011)].

4.2 Algorithm design

However, ADMM cannot be applied to problem (3) directly, as the coupling of variables makes it impossible to separate the objective function over each set of variables. In this case, a set of auxiliary variables $\{(\boldsymbol{w}_i, b_i), i = 1,\ldots,N\}$ are introduced to reformulate problem (3) as:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\boldsymbol{w}_i^T\boldsymbol{x}_{ij}+b_i-y_{ij})^2+\lambda\|\boldsymbol{w}\|_1 \\ \text{s.t.} \quad & \boldsymbol{w}_i=\boldsymbol{w}, b_i=b, i=1,\ldots,N. \end{aligned} \tag{7}$$

By enforcing equality constraints, the new problem (7) is obviously equivalent to the original problem (3). Particularly, $\{(\boldsymbol{w}, b)\}$ can be regarded as the global model parameters at the edge server, while $\{(\boldsymbol{w}_i, b_i), i = 1, \ldots, N\}$ can be regarded as the local model parameters at each IoT device i. Now we are able to separate the objective function over $\{(\boldsymbol{w}, b)\}$ and $\{(\boldsymbol{w}_i, b_i), i = 1, \ldots, N\}$. The augmented Lagrangian of problem (7) can be obtained as:

$$\begin{aligned} L_\rho(\xi, \psi, \zeta) =& \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\boldsymbol{w}_i^T\boldsymbol{x}_{ij} + b_i - y_{ij})^2 + \lambda\|\boldsymbol{w}\|_1 \\ &+ \sum_{i=1}^{N}((\boldsymbol{w}_i - \boldsymbol{w})^T\boldsymbol{\zeta}_{i,w} + \zeta_{i,b}(b_i - b)) \\ &+ \sum_{i=1}^{N}\frac{\rho}{2}((\boldsymbol{w}_i - \boldsymbol{w})^T(\boldsymbol{w}_i - \boldsymbol{w}) + (b_i - b)^2), \end{aligned} \tag{8}$$

where for simplicity we define $\xi = \{(\boldsymbol{w}, b)\}$, $\psi = \{(\boldsymbol{w}_i, b_i), i = 1, \ldots, N\}$, and $\zeta = \{(\boldsymbol{\zeta_{i,w}}, \zeta_{i,b}), i = 1, \ldots, N\}$ as the dual variables associated with the equality constraints in (7). The problem is then solved by updating ξ, ψ, and ζ sequentially. In each iteration t, the updates are performed as follows:

1. ξ-update: The update of ξ is performed by solving the following subproblem:

$$\begin{aligned} \min_{\xi} \quad & \lambda\|\boldsymbol{w}\|_1 + \frac{\rho N}{2}\boldsymbol{w}^T\boldsymbol{w} - \rho\boldsymbol{w}^T\sum_{i=1}^{N}\boldsymbol{w}_i^t \\ & - \boldsymbol{w}^T\sum_{i=1}^{N}\boldsymbol{\zeta}_{i,w}^t + \frac{\rho N}{2}b^2 - \rho b\sum_{i=1}^{N}b_i^t - b\sum_{i=1}^{N}\zeta_{i,b}^t. \end{aligned} \tag{9}$$

By denoting $\overline{\boldsymbol{A}}$ as the mean of a vector $\boldsymbol{A}$, we can rewrite (9) as:

$$\begin{aligned} \min_{\xi} \quad & \lambda\|\boldsymbol{w}\|_1 + \frac{1}{2}\rho N\boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{w} - 2\overline{\boldsymbol{w}}^t - \frac{2\overline{\boldsymbol{\zeta}}_w^t}{\rho}) \\ & + \frac{1}{2}\rho Nb(b - 2\overline{b}^t - \frac{2\overline{\zeta}_b^t}{\rho}). \end{aligned} \tag{10}$$

Since problem (10) is convex but non-differentiable, we use the subgradient calculus technique [Nesterov (2013)] in convex analysis to obtain a closed-form solution. The solution is given by

$$\boldsymbol{w}^{t+1} = \begin{cases} \overline{\boldsymbol{w}}^t + \frac{\overline{\boldsymbol{\zeta}}_w^t}{\rho} - \frac{\lambda}{\rho N}, & \overline{\boldsymbol{w}}^t + \frac{\overline{\boldsymbol{\zeta}}_w^t}{\rho} > \frac{\lambda}{\rho N} \\ 0, & \overline{\boldsymbol{w}}^t + \frac{\overline{\boldsymbol{\zeta}}_w^t}{\rho} \in [-\frac{\lambda}{\rho N}, \frac{\lambda}{\rho N}] \\ \overline{\boldsymbol{w}}^t + \frac{\overline{\boldsymbol{\zeta}}_w^t}{\rho} + \frac{\lambda}{\rho N}, & \overline{\boldsymbol{w}}^t + \frac{\overline{\boldsymbol{\zeta}}_w^t}{\rho} < -\frac{\lambda}{\rho N} \end{cases} \tag{11}$$

$$b^{t+1} = \overline{b}^t + \frac{\overline{\zeta}_b^t}{\rho} \tag{12}$$

2. ψ**-update:** The update of ψ is performed by solving the following subproblem:

$$\begin{aligned} \min_{\psi} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\boldsymbol{w}_i^T\boldsymbol{x}_{ij} + b_i - y_{ij})^2 \\ & + \sum_{i=1}^{N}\frac{\rho}{2}\boldsymbol{w}_i^T(\boldsymbol{w}_i - 2\boldsymbol{w}^{t+1} + \frac{2\boldsymbol{\zeta}_{i,w}^t}{\rho}) \\ & + \sum_{i=1}^{N}\frac{\rho}{2}b_i(b_i - 2b^{t+1} + \frac{2\zeta_{i,b}^t}{\rho}) \end{aligned} \tag{13}$$

This problem is decomposable into N subproblems over all IoT devices, among which the device i only needs to solve its local subproblem as follows:

$$\begin{aligned} \min_{\psi_i} \quad & \frac{1}{2}\sum_{j=1}^{M_i}(\boldsymbol{w}_i^T\boldsymbol{x}_{ij} + b_i - y_{ij})^2 \\ & + \frac{\rho}{2}\boldsymbol{w}_i^T(\boldsymbol{w}_i - 2\boldsymbol{w}^{t+1} + \frac{2\boldsymbol{\zeta}_{i,w}^t}{\rho}) \\ & + \frac{\rho}{2}b_i(b_i - 2b^{t+1} + \frac{2\zeta_{i,b}^t}{\rho}) \end{aligned} \tag{14}$$

The new subproblem (14) is a typical nonlinear programming problem, which is difficult to solve due to its complexity. Even standard tools like YALMIP can be used as solvers, they are still too heavyweight to run on IoT devices. Thus, we propose to to solve subproblem (14) by using the conjugate gradient method [Nesterov (2013)] in two sequential steps. Firstly, we consider the objective function of (14) as a function h of $\boldsymbol{w}_i$, and let $b_i = b_i^t$ to find the optimal $\boldsymbol{w}_i^*$ that minimizes h. Then, we consider the objective function of (14) as a function h' of b_i, and let $\boldsymbol{w}_i = \boldsymbol{w}_i^*$ to find the optimal b_i^* that minimizes h'. After these two steps, we obtain the solution as $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^*$, and $b_i^{t+1} = b_i^*$.

Due to space limitations, we only introduce the algorithm for $\boldsymbol{w}_i^*$ in Algorithm 1, while b_i^* can be obtained in a similar way. In Algorithm 1, k denotes the iteration time, p denotes

the conjugate direction, ε denotes the convergence criteria. Particularly, we design a simple heuristic to search for the optimal step size σ from a given set S, with the help of two auxiliary parameters s_1 and s_2 [Hager and Zhang (2006)]. That's because the objective function $h(\boldsymbol{w}_i)$ does not contain a symmetric and positive-definite matrix, which is required by standard conjugate gradient for determining the value of σ [Nesterov (2013)]. Algorithm 1 presents the parameter values used in our experiments in Section 3. Note that the choice of suitable values for these parameters depends on the scale of variable values in $h(\boldsymbol{w}_i)$.

Algorithm 1 Conjugate gradient algorithm for obtaining $\boldsymbol{w}_i^*$

Initialize $k \leftarrow 0, \varepsilon \leftarrow 10^{-5}$, $\boldsymbol{w}_i^0 \leftarrow 0, p^0 \leftarrow -\bigtriangledown h(\boldsymbol{w}_i^0)$, $S \leftarrow \{-1, 1\}$, $s_1 \leftarrow 1 \times 10^{-2}$, and $s_2 \leftarrow 2 \times 10^{-2}$.
repeat
  $\sigma^k \leftarrow s_1$
  for each $\delta \in S$ **do**
    if $h(\boldsymbol{w}_i + \delta s_2 p^k) < h(\boldsymbol{w}_i + \sigma^k p^k)$ **then**
      $\sigma^k \leftarrow \delta$
    end if
  end for
  $\boldsymbol{w}_i^{k+1} \leftarrow \boldsymbol{w}_i^k + \sigma^k p^k$
  $\beta^k \leftarrow \frac{\|\bigtriangledown h(\boldsymbol{w}_i^{\boldsymbol{k+1}})\|^2}{\|\bigtriangledown h(\boldsymbol{w}_i^{\boldsymbol{k}})\|^2}$
  $p^{k+1} = -\bigtriangledown h(\boldsymbol{w}_i^{\boldsymbol{k+1}}) + \beta^k p^k$
  $k \leftarrow k + 1$
until Convergence: $\|\bigtriangledown h(\boldsymbol{w}_i^k)\| \leq \varepsilon$
$\boldsymbol{w}_i^* \leftarrow \boldsymbol{w}_i^k$

3. ζ-update: After obtaining ξ^{t+1} and ψ^{t+1}, we finally update the dual variables as:

$$\boldsymbol{\zeta}_{i,w}^{t+1} = \boldsymbol{\zeta}_{i,w}^t + \rho(\boldsymbol{w}_i^{t+1} - \boldsymbol{w}^{t+1}) \tag{15}$$

$$\zeta_{i,b}^{t+1} = \zeta_{i,b}^t + \rho(b_i^{t+1} - b^{t+1}) \tag{16}$$

The entire process of algorithm execution in our scenarios is summarized in Algorithm 2. In this work, the primal residual r and the dual residual s [Boyd, Parikh, Chu et al. (2011)] are used together as convergence criterion, and they are expected to be less than their respective tolerances ϵ^{pri} and ϵ^{dual}. According to Boyd et al. [Boyd, Parikh, Chu et al. (2011)], ϵ^{pri} and ϵ^{dual} can be reasonably chosen using an absolute tolerance ε^{abs} and a relative tolerance ε^{rel}. In each iteration, a participating device submits locally trained intermediate parameters $(\boldsymbol{w}_i, b_i)$ and $(\boldsymbol{\zeta_{i,w}}, \zeta_{i,b})$ to the edge server. The edge server then computes and updates the global model parameters $(\boldsymbol{w}, b)$ by averaging the parameters

gathered from devices. The edge server and IoT devices work in such a collaborative way for multiple iterations until the lasso model converges. Note that in the above process, all training samples are kept locally at the device's side, and protected from leakage to the edge server or other devices.

Algorithm 2 Distributed lasso algorithm in edge computing

1: The edge server initializes $t \leftarrow 0$, $\overline{\boldsymbol{w}}^0 \leftarrow 0$, $\overline{b}^0 \leftarrow 0$.
2: Each IoT device i initializes $t \leftarrow 0$, $\boldsymbol{\zeta}_{i,w}^0 \leftarrow 0$, $\zeta_{i,b}^0 \leftarrow 0$.
3: **repeat**
4: Given $(\boldsymbol{w}_i^t, b_i^t)$ and $(\boldsymbol{\zeta}_{\boldsymbol{i,w}}^{\boldsymbol{t}}, \zeta_{i,b}^t)$ from each device $i \in \{1, \ldots, N\}$, the edge server computes $\overline{\boldsymbol{w}}^t$, $\overline{b}^t$, $\overline{\boldsymbol{\zeta}}_w^t$, and $\overline{\zeta}_b^t$. According to (11) and (12), the server updates $\boldsymbol{w}^{t+1}$ and b^{t+1}, and broadcasts the results to all devices.
5: Given $\boldsymbol{w}^{t+1}$ and b^{t+1}, each device i solves the subproblem (14) independently to obtain $\boldsymbol{w}_i^{t+1}$ and b_i^{t+1}.
6: Each device i updates its dual variables $\boldsymbol{\zeta}_{i,w}^{t+1}$ and $\zeta_{i,b}^{t+1}$ according to (15) and (16).
7: Each device i sends the results of $(\boldsymbol{w}_i^{t+1}, b_i^{t+1})$ and $(\boldsymbol{\zeta}_{i,w}^{t+1}, \zeta_{i,b}^{t+1})$ to the edge server.
8: $t \leftarrow t + 1$
9: **until** Convergence: $\|r^t\|_2 \leq \varepsilon^{pri}$ and $\|s^t\|_2 \leq \varepsilon^{dual}$

5 Performance evaluation

In this section, we conduct simulation experiments to evaluate the performance of our ADMM-based algorithm.

5.1 Experiment settings

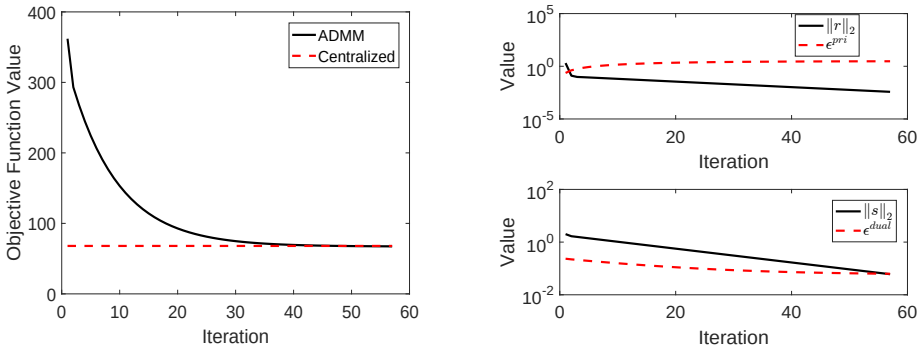
We evaluate our algorithm on two datasets, i.e., a synthetic one and a real-world one. The synthetic dataset contains 1500 data samples, each of which includes nine dimensional feature vector. The generation of data follows the description in Boyd et al. [Boyd, Parikh, Chu et al. (2012)]. Using this dataset, we can focus on evaluating the algorithm performance under different parameter settings, regardless of the impact of data quality. The experiment results are shown in Figs. 1-4. Then, we use a well-known diabetes dataset from [Efron, Hastie, Johnstone et al. (2004)] to further investigate the algorithm performance under realistic conditions. This dataset contains 442 data samples, each of which includes ten dimensional feature vector. The experiment results are shown in Fig. 5. In all experiments, we split the given dataset into 70% training data and 30% validation data. Unless otherwise specified, the simulation parameters are given as follows: $N = 10$, $\lambda = 1.0$, $\rho = 1.0$, $\varepsilon^{abs} = 0.2$, and $\varepsilon^{rel} = 0.5$.

To provide performance benchmarks for the proposed algorithm, we compare it with two baselines, namely centralized training approach and independent training approach. As stated in Section 1, the centralized approach generally can obtain the optimal result, but suffers from problems of scalability, performance and privacy. In the independent approach,

each participating device trains its own model independently with local data samples. This approach overcomes the challenges of scalability and privacy at the cost of modeling performance, and the result accuracy is highly correlated with both the size and the quality of the training set.

5.2 Experiment results

Convergence of our algorithm. Fig. 1 depicts the convergence property of our ADMM algorithm. The left-hand plot (Fig. 1(a)) shows the change of objective function value w.r.t. iterations, while the right-hand plot (Fig. 1(b)) shows the progress of the primal and dual residual norm w.r.t. iterations. The objective value computed using the centralized algorithm is taken as the global optimal result. As shown in Fig. 1, we observe that our algorithm approaches very close to optimum after 30 iterations, and finally converges within 58 iterations according to the given stopping criterion.



(a) Objective value versus iterations.

(b) Norms of primal residual and dual residual versus iteration.

Figure 1: Convergence of the ADMM-based algorithm

To fully investigate the convergence performance, we conduct random independent experiments for 11 times to compare the number of iterations for achieving convergence. As shown in Fig. 2, our algorithm takes at most 138 iterations to converge, and the fastest run only takes 17 iterations. For 80% of the time, our algorithm achieves convergence within 120 iterations. On the other hand, the centralized algorithm takes at least 580 iterations to converge, and even 800 iterations in the worst case. These results clearly indicate that our algorithm converges significantly faster than the centralized approach.

Impacts of parameter settings. Next we study the algorithm performance under different N, ρ, and λ values (with other parameters fixed). All these results are plotted in Fig. 3. As shown in Fig. 3(a), our algorithm can always converge to the same optimal objective value no matter how many IoT devices are involved. We can observe that with the increase of N, our algorithm converges with only moderate increment of iterations. These observations demonstrate the scalability of this algorithm as well as its associated overhead for device

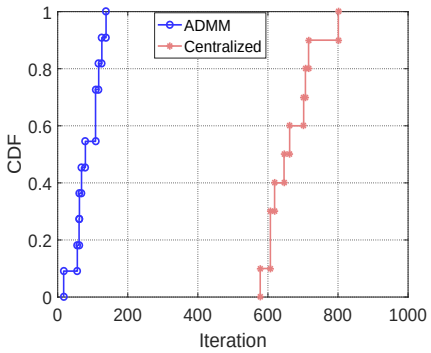
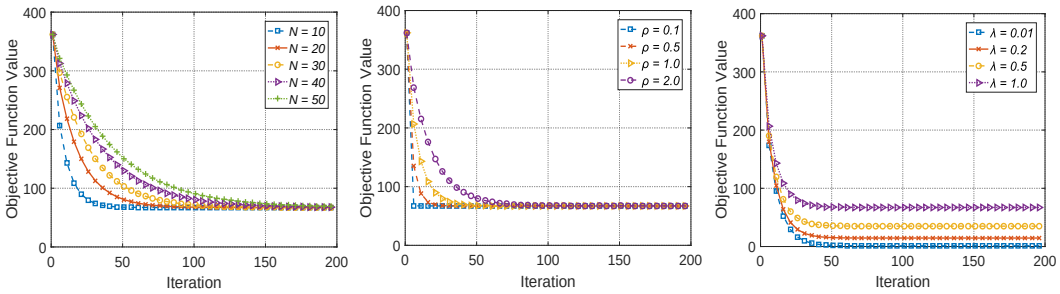


Figure 2: CDF of the number of iterations to achieve convergence.

coordination. Similarly, varying ρ also has little impact on the final optimization objective, as shown in Fig. 3(b). However, a smaller value of ρ tends to speed up the dual update and achieve a faster convergence. From Fig. 3(c), we can see that changing λ only determines the expected optimization objective in (3), but has neglectable impacts on the convergence and its rate of the algorithm.

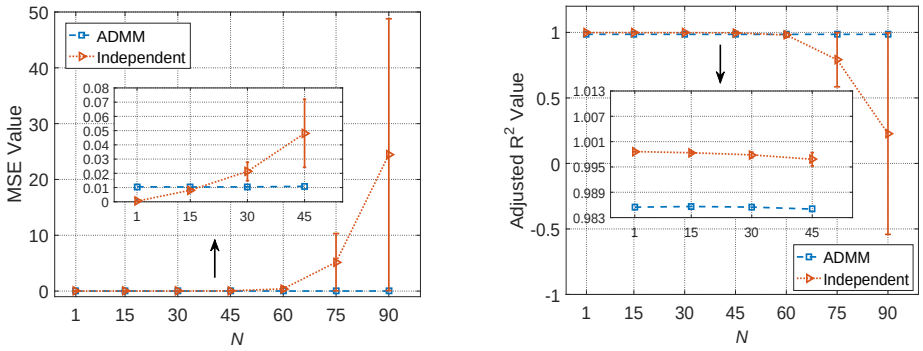


(a) Impact of N on algorithm convergence. (b) Impact of ρ on algorithm convergence. (c) Impact of λ on algorithm convergence

Figure 3: Convergence performance of the algorithm with varying parameter settings

Comparison of modeling performance. In order to evaluate model performance, we use the two most common metrics in regression analysis, including Mean Squared Error (MSE) and Adjusted R-Squared (Adjusted-R^2) [Spencer, Alfandi and Al-Obeidat (2018); Kmenta and Rafailzadeh (1997)]. MSE measures the expected squared distance between actual and predicted values. It must be non-negative, and a value closer to 0 indicates a better model. Adjusted-R^2 is used to measure the correlation between actual and predicted values. It can take on any value no greater than 1, and a value closer to 1 indicates a better fit. Since the performance of the independent approach relies on the size of local training set, we randomly partition the original dataset into N training subsets, and compare the performance of algorithms under different device numbers N. It is noted that when $N = 1$, the independent approach does actually equate with the centralized approach.

As shown in Fig. 4(a), the average MSE of the independent approach keeps increasing with the increment of N. The variance of MSE values also becomes larger as N increases, implying that different IoT devices with same-sized data are more likely to obtain distinct training results. The MSE of our algorithm is always kept at about 0.01, and does not depend on N. From Fig. 4(b), we observe that our algorithm obtains a steady value of Adjusted-R^2 around 0.985. The independent approach has a slightly better performance than ours when $N < 60$, but its performance degrades significantly when $N \geq 60$. An IoT device may even obtain a negative Adjusted-R^2 when $N = 90$, meaning that its resulting model doesn't fit the data [Kmenta and Rafailzadeh (1997)]. Note that for the centralized algorithm, MSE = 0.000451 and Adjusted-R^2 = 0.998639. From these results, we know that the lasso models trained by the independent approach may not be robust for individual IoT devices, due to the limitation on data size and the lack of data diversity. Our algorithm can always converge to near-optimal and obtain modestly accurate models comparable to that of centralized approach, by utilizing data samples contributed by many IoT devices.



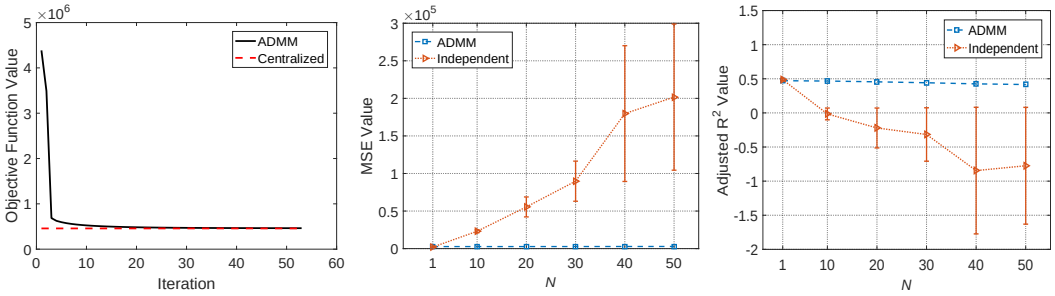
(a) MSE. (b) Adjusted-R^2.

Figure 4: Comparisons on model performance of different algorithms using different metrics

Results of real-world dataset. Furthermore, we evaluate our algorithm on the diabetes dataset in Efron et al. [Efron, Hastie, Johnstone et al. (2004)] using the same metrics. Due to space limitation, typical experimental results are chosen to be plotted in Figs. 5(a)-5(c), respectively.

We obtain some new observations that are different from those presented above and need special attention. Firstly, as shown in Fig. 5(a), the objective value of our algorithm falls off spectacularly fast during the initial 3 iterations, and converges after 25 iterations. We attribute this fast convergence rate to the significantly small size of this dataset. Secondly, the performance of the independent approach is obviously the worst among all three algorithms, according to Figs. 5(b) and Fig. 5(c). We can notice that start from $N = 10$, its MSE increases sharply to values far greater than 0 and those of other algorithms. Meanwhile, its Adjusted-R^2 decreases quickly and directly to negative values. These observations in-

dicate that when training data are irregularly distributed overall, which is very common in reality, a good model is hardly to obtain by individual devices due to the severe lack of data diversity. Thirdly, our algorithm still achieves a good modeling performance comparable to that obtained by centralized approach.



(a) Objective value versus iterations. (b) MSE. (c) Adjusted-R^2.

Figure 5: Experimental results on the diabetes dataset

6 Conclusion

In this paper, we present a collaborative learning scheme for training lasso regression models based on data samples from IoT devices in edge computing. The target optimization problem of lasso is solved in a distributed fashion by leveraging ADMM, while the participating devices only need to share indispensable intermediate results with the edge server for model training. Simulation results on two typical datasets demonstrate that the proposed scheme can quickly converge to near-optimal within dozens of iterations, and significantly outperforms other baseline solutions in performance evaluation. Our future work involves implementing our scheme on a real edge computing platform to evaluate its feasibility and performance.

Acknowledgement: This work was supported by the Fundamental Research Funds for the Central Universities of China under Grants 2017JBM021 and 2016JBZ006, and CETC Joint Fund under Grant 6141B08020101. Note that Yangyang Li and Xue Wang have equal contributions to this work.

References

Allen-Zhu, Z. (2017): Katyusha: the first direct acceleration of stochastic gradient methods. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1200-1205.

Bazerque, J. A.; Mateos, G.; Giannakis, G. B. (2010): Distributed lasso for in-network linear regression. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2978-2981.

Bertsekas, D. P.; Tsitsiklis, J. N. (1989): *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, USA.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. (2011): Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1-122.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. (2012): Matlab scripts for alternating direction method of multipliers. *Technical Report*, pp. 1-50.

Dhar, S.; Yi, C.; Ramakrishnan, N.; Shah, M. (2015): Admm based scalable machine learning on spark. *IEEE International Conference on Big Data (Big Data)*, pp. 1174-1182.

Efron, B.; Hastie, T.; Johnstone, I.; Tibshirani, R. (2004): Least angle regression. *The Annals of Statistics*, vol. 32, no. 2, pp. 407-499.

Gong, Y.; Fang, Y.; Guo, Y. (2016): Private data analytics on biomedical sensing data via distributed computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 3, pp. 431-444.

Ha, K.; Pillai, P.; Lewis, G.; Simanta, S.; Clinch, S. et al. (2013): The impact of mobile multimedia applications on data center consolidation. *IEEE International Conference on Cloud Engineering (IC2E)*, pp. 166-176.

Hager, W. W.; Zhang, H. (2006): A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, vol. 2, no. 1, pp. 35-58.

Ismail, B. I.; Goortani, E. M.; Karim, M. B. A.; Tat, W. M.; Setapa, S. et al. (2015): Evaluation of docker as edge computing platform. *IEEE Conference on Open Systems (ICOS)*, pp. 130-135.

Joye, M.; Libert, B. (2013): A scalable scheme for privacy-preserving aggregation of time-series data. *International Conference on Financial Cryptography and Data Security*, pp. 111-125.

Kmenta, J.; Rafailzadeh, B. (1997): *Elements of Econometrics*. University of Michigan Press, MI, USA.

Kwon, D.; Hodkiewicz, M. R.; Fan, J.; Shibutani, T.; Pecht, M. G. (2016): Iot-based prognostics and systems health management for industrial applications. *IEEE Access*, vol. 4, pp. 3659-3670.

Longo, M.; Mateos, C.; Zunino, A. (2018): A model for hour-wise prediction of mobile device energy availability. *Information Technology-New Generations*, pp. 351-358.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B. A. (2017): Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273-1282.

Meeds, E.; Hendriks, R.; Al Faraby, S.; Bruntink, M.; Welling, M. (2015): Mlitb: machine learning in the browser. *PeerJ Computer Science*, vol. 1, no. 11, pp. 1-24.

Nesterov, Y. (2013): *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, NY, USA.

Portelli, K.; Anagnostopoulos, C. (2017): Leveraging edge computing through collaborative machine learning. *5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 164-169.

Richter, A. N.; Khoshgoftaar, T. M.; Landset, S.; Hasanin, T. (2015): A multi-dimensional comparison of toolkits for machine learning with big data. *IEEE International Conference on Information Reuse and Integration*, pp. 1-8.

Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. (2016): Edge computing: vision and challenges. *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646.

Spencer, B.; Alfandi, O.; Al-Obeidat, F. (2018): A refinement of lasso regression applied to temperature forecasting. *Procedia Computer Science*, vol. 130, no. 2018, pp. 728-735.

Tibshirani, R. (1996): Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267-288.

Truong, H.; Dustdar, S. (2015): Principles for engineering iot cloud systems. *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68-76.

Xi, X.; Sheng, V. S.; Sun, B.; Wang, L.; Hu, F. (2018): An empirical comparison on multi-target regression learning. *Computers, Materials & Continua*, vol. 56, no. 2, pp. 185-198.

Zhang, C.; Lee, H.; Shin, K. (2012): Efficient distributed linear classification algorithms via the alternating direction method of multipliers. *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pp. 1398-1406.

Zhou, J.; Cao, Z.; Dong, X.; Vasilakos, A. V. (2017): Security and privacy for cloud-based iot: Challenges. *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26-33.

Zinkevich, M. A.; Weimer, M.; Smola, A.; Li, L. (2010): Parallelized stochastic gradient descent. *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, vol. 2.