Online and Distributed Robust Regressions with Extremely Noisy Labels

SHUO LEI, Virginia Tech
XUCHAO ZHANG, NEC Laboratories America
LIANG ZHAO, Emory University
ARNOLD P. BOEDIHARDJO, Maxar Technologies Inc.
CHANG-TIEN LU, Virginia Tech

In today's era of big data, robust least-squares regression becomes a more challenging problem when considering the extremely corrupted labels along with explosive growth of datasets. Traditional robust methods can handle the noise but suffer from several challenges when applied in huge dataset including (1) computational infeasibility of handling an entire dataset at once, (2) existence of heterogeneously distributed corruption, and (3) difficulty in corruption estimation when data cannot be entirely loaded. This article proposes online and distributed robust regression approaches, both of which can concurrently address all the above challenges. Specifically, the distributed algorithm optimizes the regression coefficients of each data block via heuristic hard thresholding and combines all the estimates in a distributed robust consolidation. In addition, an online version of the distributed algorithm is proposed to incrementally update the existing estimates with new incoming data. Furthermore, a novel online robust regression method is proposed to estimate under a biased-batch corruption. We also prove that our algorithms benefit from strong robustness guarantees in terms of regression coefficient recovery with a constant upper bound on the error of state-of-the-art batch methods. Extensive experiments on synthetic and real datasets demonstrate that our approaches are superior to those of existing methods in effectiveness, with competitive efficiency.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**; • **Theory of computation** → *Online learning algorithms;*

Additional Key Words and Phrases: Robust regression, extremely noisy labels, online robust regression, distributed optimization

ACM Reference format:
Shuo Lei, Xuchao Zhang, Liang Zhao, Arnold P. Boedihardjo, and Chang-Tien Lu. 2021. Online and Distributed Robust Regressions with Extremely Noisy Labels. *ACM Trans. Knowl. Discov. Data.* 16, 3, Article 41 (October 2021), 24 pages.
https://doi.org/10.1145/3473038

Authors' addresses: S. Lei and C.-T. Lu, Department of Computer Science, Virginia Tech, Falls Church, VA 22043; emails: {slei, ctlu}@vt.edu; X. Zhang (corresponding author), NEC Laboratories America, Princeton, NJ 08540; email: xuczhang@nec-labs.com; L. Zhao, Department of Computer Science, Emory University, Atlanta, GA 30307; email: liang.zhao@emory.edu; A. P. Boedihardjo, Maxar Technologies Inc., Herndon, VA 20171; email: arnold.p.boedihardjo@vt.edu.

1556-4681/2021/10-ART41 $15.00
https://doi.org/10.1145/3473038

1 INTRODUCTION

In the era of data explosion, the fast-growing amount of data makes processing entire datasets at once remarkably difficult. For instance, urban **Internet of Things** (**IoT**) systems [32] can produce millions of data records every second in monitoring air quality, energy consumption, and traffic congestion. The presence of noise and corruption in real-world data can be inevitably caused by accidental outliers [27], transmission loss [28], or even adversarial data attacks [7]. As the most popular statistical approach, the traditional least-squares regression method is vulnerable to outlier observations [22] and not scalable to large datasets [24, 37]. By considering both robustness and scalability in a least-squares regression model, we study **scalable robust least-squares regression** (***SRLR***) [1, 4] to handle the problem of learning a reliable set of regression coefficients given a large dataset with several corruptions in its response vector. Due to the ubiquitousness of data corruption and explosive data growth, *SRLR* has become a critical component of several important real-world applications in various domains such as economics [2], signal processing [33, 40], and network processing [17].

The goal of *SRLR* problem is to recover the true regression coefficients under the assumption that both the observed response $\boldsymbol{y}$ and data matrix X are too large to be handled simultaneously. A commonly adopted model from existing robust regression methods [4, 34] assumes that the observed response is obtained from the generative model $\boldsymbol{y} = X^T\boldsymbol{\beta}_* + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\beta}_*$ is the true regression coefficients that we wish to recover, $\boldsymbol{u}$ is the corruption vector with arbitrary values, and $\boldsymbol{\varepsilon}$ represents the additive dense noise. Three different corruption distributions are considered in this article: (1) *distributed corruption* assumes samples in mini-batches are arbitrarily corrupted with the corrupted ratio no more than 50%; (2) *batched corruption* assumes that up to k of mini–batches in the online data batch set are arbitrarily corrupted, which contain an overwhelmingly amount of corrupted samples (more than 50%); and (3) *biased-batch corruption* assumes that up to k of mini-batches in the online data batch set are biased corrupted with uniform distribution, like concentrated in the front, middle, or back of the sequence.

Existing robust learning methods typically focus on modeling the entire dataset at once; however, they may meet the bottleneck in terms of computation and memory as more and more datasets are becoming too large to be handled integrally. For those seeking to address this issue, the major challenges can be summarized as follows: (1) **Computational infeasibility of handling the entire dataset at once.** Existing robust methods typically generate the predictor by learning on the entire training dataset. However, the explosive growth of data makes it infeasible to handle the entire dataset up to a terabyte or even petabyte at once. Therefore, a scalable algorithm is required to handle the robust regression task for massive datasets. (2) **Existence of heterogeneously distributed corruption.** Due to the unpredictability of corruptions, the corrupted samples can be arbitrarily distributed in the whole dataset. Considering the entire dataset as the combination of multiple mini-batches, some batches may contain large amounts of outliers. Thus, simply applying the robust method on each batch and averaging all the estimates together is not an ideal strategy, as some estimates will be arbitrarily poor and break down the overall performance of robustness. (3) **Difficulty in corruption estimation when data cannot be entirely loaded.** Most robust methods assume the corruption ratio of input data is a known parameter; however, if a small batch of data can be loaded as inputs for robust methods, it is infeasible to know the corruption ratio of all the mini-batches. Moreover, simply using a unified corruption ratio for all the mini-batches is clearly not an ideal solution as corrupted samples can be regarded as uncorrupted, and vice versa. In addition, even though some robust methods can estimate the corruption ratio based on data observations, it is also infeasible to estimate the ratio when corruption in one mini-batch is greater than 50%. However, the situation can be very common when corruption is heterogeneously distributed.

In order to simultaneously address all these technical challenges, this article presents a novel **Distributed Robust Least-squares Regression** (***DRLR***) method and its two online versions, named **Online Robust Least-squares Regression** (***ORLR***) and **Online Robust Least-squares Regression under Biased-batch Corruption** (***ORLR-BC***). These two online methods are proposed to handle the scalable robust regression problem in large datasets with heterogeneously distributed corruption. In *DRLR*, the regression coefficient of each mini-batch is optimized via heuristic hard thresholding, and then all the estimates are combined in distributed robust consolidation. Based on *DRLR*, the *ORLR* algorithm incrementally updates the existing estimates by replacing old corrupted estimates with those of new incoming data, which is more efficient than *DRLR* in handling new data and reflects the time-varying characteristics. To solve the biased-batch data corruption that cannot be handled by *ORLR* algorithm, we propose a new *ORLR-BC* algorithm. Specifically, the *ORLR-BC* algorithm considers not only the sequence of data corruption but also existing estimates to reduce the impact of the sequence of mini-batches, which is more efficient than *ORLR*, especially when corrupted batches concentratedly arrive at first. In addition, we prove that the proposed algorithms preserve the overall robustness of regression coefficients in the entire dataset. The main contributions of this article are as follows:

- **Formulating a framework for the SRLR problem.** A framework is proposed for SRLR problem where the amount of the entire data with extremely noisy labels is too large to store in memory at once. Specifically, given a large dataset with large-scale corruptions, a reliable set of regression coefficients is learned with limited memory.
- **Proposing online and distributed algorithms to handle the uniform label corruption.** By utilizing robust consolidation methods, we propose both online and distributed algorithms to obtain overall robustness even though the corruption is arbitrarily distributed. Moreover, the online algorithm performs more efficiently in handling new incoming data and presents the time-varying characteristics of regression coefficients.
- **Designing an online algorithm to handle the biased-batch corruption.** Considering not only the relevance to previous regression coefficients but the sequence of data, we extend the online algorithm to handle the biased-batch corruption, which is the worst scenarios for our online algorithm.
- **Providing a rigorous robustness guarantee for regression coefficient recovery.** We prove that our online and distributed algorithms recover the true regression coefficient with a constant upper bound on the error of state-of-the-art batch methods under the assumption that corruption can be heterogeneously distributed. Specifically, the upper bound of online algorithm will be infinitely close to distributed algorithm when the number of mini-batches is large.
- **Conducting extensive experiments for performance evaluations.** The proposed methods were evaluated on both synthetic data and real-world datasets with various corruption and data-size settings. The results demonstrate that the proposed approaches consistently outperform existing methods along multiple metrics with a competitive running time.

This article is a systematic extension of [38]. Compared to the preliminary version, we propose a new adversarial online robust regression method for the biased-batch corruption estimation task. Additional discussions and empirical results are presented. The rest of this article is organized as follows: Section 2 reviews background and related work, and Section 3 introduces the problem setup. The proposed online and distributed robust regression algorithms are presented in Section 4. Section 5 presents the proof of recovery guarantee in regression coefficients. The experiments on both synthetic and real-world datasets are presented in Section 6, and this article concludes with a summary of the research in Section 7.

2 RELATED WORK

The work related to this article is summarized in two categories below.

Robust regression model: A large body of literature on the robust regression problem has been built over the last few decades. Most of studies focus on handling stochastic noise or small bounded noise [6, 20, 26], but these methods, modeling the corruption on stochastic distributions, cannot be applied to data that may exhibit biased-batch corruption [7]. Some studies assume the adversarial label corruption in the data, but most of them lack the strong guarantee of regression coefficients recovery under the arbitrary corruption assumption [7, 24]. Chen et al. [7] proposed a robust algorithm based on a trimmed inner product, but the recovery boundary is not tight to ground truth in a massive dataset. McWilliams et al. [24] proposed a sub-sampling algorithm for large-scale corrupted linear regression, but their recovery result is not close to an exact recovery [4]. To pursue exact recovery results for robust regression problem, some studies focused on L_1 penalty based convex formulations [25, 31]. However, these methods imposed severe restrictions on the data distribution such as row-sampling from an incoherent orthogonal matrix[25].

Currently, most research in this area requires the corruption ratio parameter, which is difficult to determine under the assumption that the dataset can be arbitrarily corrupted. For instance, She and Owen [30] rely on a regularization parameter to control the size of the uncorrupted set based on soft-thresholding. Instead of a regularization parameter, Chen et al. [7] require the upper bound of the outliers number, which is also difficult to estimate. Bhatia et al. [4] proposed a hard-thesholding algorithm with a strong guarantee of coefficient recovery under a mild assumption on input data. However, its recovery error can be more than doubled in size if the corruption ratio is far from the true value. Recently, Zhang et al. [39] proposed a robust algorithm that learns the optimal uncorrupted set via a heuristic method. However, all of these approaches require the entire training dataset to be loaded and learned at once, which is infeasible to apply in massive and fast growing data.

Online and distributed learning: Most of the existing online learning methods optimize surrogate functions such as stochastic gradient descent [13, 18, 21] to update estimates incrementally. For instance, Duchi et al. [13] proposed a new, informative subgradient method that dynamically incorporates the geometric knowledge of the data observed in earlier iterations. Some adaptive linear regression methods such as recursive least squares [15] and **online passive aggressive algorithms** (**OPAAs**) [8] provide an incremental update on the regression model for new data to capture time-varying characteristics. However, these methods cannot handle the outlier samples in the streaming data. For distributed learning [5, 23], most approaches such as MapReduce [10] focus on distributed solutions for large-scale problems that are not robust to noise and corruption in real-world data.

The existing distributed robust optimization methods can be divided into two categories: those that use moment information [12, 19] and those that utilize directly on the probability distributions [3, 9, 14]. For instance, Delage et al. [11] proposed a model that describes uncertainty in both the distribution form and moments in a distributed robust stochastic program. However, these methods assume either the moment information or probability distribution as prior knowledge, which is difficult to know in practice. In robust online learning, few methods have been proposed in the past few years. For instance, Sharma et al. [29] proposed an online smoothed passive-aggressive algorithm to update estimates incrementally in a robust manner. However, the method assumes the corruption is in stochastic distributions, which is infeasible for data with extremely corrupted labels. Recently, Feng et al. [16] proposed an **online robust learning** (**ORL**) approach that gives

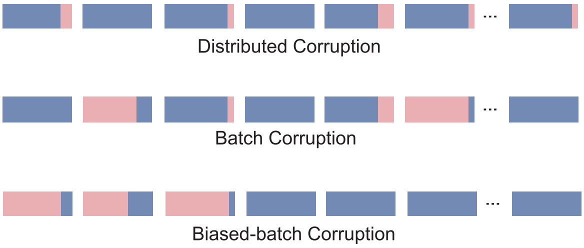


Fig. 1. Illustrations of the distributed corruption model, batched corruption model, and the biased-batch corruption model.

a provable robustness guarantee under the assumption that data corruption is heterogeneously distributed. However, the method requires that the corruption ratio of each data batch be given as parameters, which is not practical for users to estimate.

3 PROBLEM FORMULATION

In the setting of online and distributed learning, we consider the samples to be provided in a sequence of mini-batches as $\{X^{(1)}, \ldots, X^{(m)}\}$, where $X^{(i)} \in \mathbb{R}^{p\times n}$ represents the sample data for the ith batch. We assume the corresponding response vector $\boldsymbol{y}^{(i)} \in \mathbb{R}^{n\times 1}$ is generated using the following model:

$$\boldsymbol{y}^{(i)} = \left[\boldsymbol{X}^{(i)}\right]^T \boldsymbol{\beta}_* + \boldsymbol{u}^{(i)} + \boldsymbol{\varepsilon}^{(i)}, \tag{1}$$

where $\boldsymbol{\beta}_* \in \mathbb{R}^{p\times 1}$ is the ground truth coefficients of the regression model and $\boldsymbol{u}^{(i)}$ is the sparse corruption vector of the ith mini-batch. $\boldsymbol{\varepsilon}^{(i)}$ represents the additive dense noise for the ith mini-batch, where $\varepsilon_j^{(i)} \sim \mathcal{N}(0, \sigma^2)$.

When the entire dataset is too large to be handled simultaneously, it is usually processed by multiple mini-batches. Due to the unpredictability of corruptions, the corrupted samples can be arbitrarily distributed in the whole dataset. For example, some batches may contain large amounts of outliers and some batches may have few corrupted data. Since the online learning methods are sensitive to the data sequence, modeling different corrupted data sequence scenarios is non-trivial. Here, we consider three types of corrupted data distributions: distributed corruption, batched corruption, and biased-batch corruption. Specifically, distributed corruption models the situation where the corrupted data are arbitrarily distributed in mini-batches and the corrupted ratio of each batch is less than half. Batch corruption considers the situation where the corrupted ratio of some mini-batches are more than half. In other words, the corrupted data in the whole dataset are concentrated mainly in a small number of mini-batches. When the corrupted ratio of the mini-batch is more than half, it is called corrupted batches. Furthermore, if the corrupted batches concentrated in the data streaming and the corrupted data inside has an uniform distribution, it is defined as biased-batch corruption. We illustrate these corruption models in Figure 1, where the blue parts denote the uncorrupted data and the pink parts denote the corrupted data.

Definition 3.1 (Distributed Corruption). Samples in mini-batches are arbitrarily corrupted with the corrupted ratio no more than half.

Definition 3.2 (Batch Corruption). Given a sequence of mini-batches $\{X^{(1)}, \ldots, X^{(m)}\}$, k out of m mini-batches are arbitrarily corrupted batches. Here, if the amount of corrupted samples in the batch $X^{(i)}$ are more than half, $X^{(i)}$ is called corrupted batch.

Table 1. Math Notations

Notations	Explanations
$X^{(i)} \in \mathbb{R}^{p \times n}$	collection of data samples of the ith mini-batch
$\boldsymbol{y}^{(i)} \in \mathbb{R}^{n \times 1}$	response vector of the ith mini-batch
$\boldsymbol{\beta}^{(i)} \in \mathbb{R}^{p \times 1}$	estimated regression coefficient of the ith batch
$\boldsymbol{\beta}_*^{(i)} \in \mathbb{R}^{p \times 1}$	ground truth regression coefficient of the ith batch
$\boldsymbol{u}^{(i)} \in \mathbb{R}^{n \times 1}$	corruption vector of the ith batch
$\boldsymbol{r}^{(i)} \in \mathbb{R}^{n \times 1}$	residual vector of the ith batch
$\boldsymbol{\varepsilon}^{(i)} \in \mathbb{R}^{n \times 1}$	dense noise vector of the ith batch
$Z^{(i)} \subseteq [n]$	estimated uncorrupted set of the ith batch
$Z_*^{(i)} \subseteq [n]$	ground truth uncorrupted set, where $Z_*^{(i)} = \overline{supp(\boldsymbol{u}^{(i)})}$
$S \subseteq [m \cdot n]$	estimated uncorrupted set of entire dataset

Definition 3.3 (Biased-batch Corruption). Given a sequence of mini-batches $\{X^{(1)}, \ldots, X^{(m)}\}$, k out of m mini-batches are corrupted batches with uniform distribution.

The goal of addressing our problem is to recover the regression coefficients $\hat{\boldsymbol{\beta}}$ and determine the uncorrupted set $\hat{S}$ for the entire dataset. The problem is formally defined as follows:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \underset{\boldsymbol{\beta}, S}{\arg\min} \left\| \boldsymbol{y}_S - X_S^T \boldsymbol{\beta} \right\|_2^2 \\ s.t. \quad S &\in \left\{ \Omega\left(Z\right) \,\middle|\, \forall i \leq m : |Z^{(i)}| \geq |Z_*^{(i)}| \right\} \end{aligned} . \tag{2}$$

We define $Z^{(i)}$ as the estimated uncorrupted set for the ith mini-batch and $Z = \{Z^{(1)}, \ldots, Z^{(m)}\}$ as the collection of uncorrupted sets for all the mini-batches. The size of set $Z^{(i)}$ is represented as $|Z^{(i)}|$. The function $\Omega(\cdot)$ consolidates the estimates of all the mini-batches in terms of the distributed or online setting. $\boldsymbol{y}_S$ restricts the row of $\boldsymbol{y}$ to indices in S, and X_S signifies that the columns of X are restricted to indices in S. Therefore, we have $\boldsymbol{y}_S \in \mathbb{R}^{|S| \times 1}$ and $X_S \in \mathbb{R}^{p \times |S|}$, where p is the number of features and $|S|$ is the size of the uncorrupted set $S \subset [m \cdot n]$. The notation $Z_*^{(i)} = \overline{supp(\boldsymbol{u}^{(i)})}$ represents the true set of uncorrupted points in the ith mini-batch. The constraint of $Z^{(i)}$ is $|Z^{(i)}| \geq |Z_*^{(i)}|$; however, it is infeasible to get the ground truth uncorrupted set $Z_*^{(i)}$. Thus, the function $h(\cdot)$ is designed to estimate the size of the uncorrupted set of each mini-batch according to the residual vector $\boldsymbol{r}^{(i)}$. The residual vector $\boldsymbol{r}^{(i)} \in \mathbb{R}^n$ of the ith mini-batch is defined as $\boldsymbol{r}^{(i)} = \boldsymbol{y}^{(i)} - [X^{(i)}]^T \boldsymbol{\beta}$. We use the notation $\boldsymbol{r}_Z^{(i)}$ to represent the $|Z^{(i)}|$-dimensional residual vector containing the components in $Z^{(i)}$. The uncorrupted set of each mini-batch is consolidated by function $\Omega(\cdot)$ in both online and distributed approaches. The details of the heuristic function $h(\cdot)$ and consolidation function $\Omega(\cdot)$ are explained in Section 4. The notations used in this article are summarized in Table 1.

The problem defined above is challenging in the following three aspects. First, the least-squares function can be naively solved by taking the derivative to zero. However, as the data samples of all m mini-batches are too large to be loaded into memory simultaneously, it is impossible to calculate $\boldsymbol{\beta}$ from all the batches directly by this method. Moreover, based on the fact that the corruption ratio can be varied for each mini-batch, we cannot simply estimate the corruption set by using a fixed ratio for each mini-batch. In addition, since corruption is not uniformly distributed, some mini-batches may contain an overwhelmingly amount of corrupted samples. The corresponding estimates of regression coefficients can be arbitrarily poor and break down the overall result. In the next section, we present both online and distributed robust regression algorithms based on heuristic hard thresholding and robust consolidation to address all three challenges.

4 METHODOLOGY

In this section, we propose both online and distributed robust regression algorithms to handle large datasets in multiple mini-batches. To handle each single mini-batch among these mini-batches, a **heuristic robust regression** (***HRR***) method is proposed in Section 4.1. Based on *HRR*, a new approach, *DRLR*, is presented in Section 4.2 to process multiple mini-batches in distributed manner. In Section 4.3, a novel online version of *DRLR*, namely *ORLR*, is proposed to incrementally update the estimate of regression coefficients with new incoming data.

4.1 Single-Batch Heuristic Robust Regression

In order to efficiently solve the single batch problem when $m = 1$ in Equation (2), we propose a robust regression algorithm, *HRR*, based on heuristic hard thresholding. The algorithm heuristically determines the uncorrupted set $Z^{(i)}$ for the ith mini-batch according to its residual vector $\boldsymbol{r}^{(i)}$. Specifically, a novel heuristic function $h(\cdot)$ is proposed to estimate the lower-bound size of the uncorrupted set $Z^{(i)}$ for each mini-batch, which is formally defined as

$$h(\boldsymbol{r}^{(i)}) := \underset{\tau \in \mathbb{Z}^+, \tau \le n}{\arg\max}\ \tau \quad s.t.\ \ r^{(i)}_{\varphi(\tau)} \le \frac{2\tau r^{(i)}_{\varphi(\tau_o)}}{\tau_o}, \tag{3}$$

where the residual vector of ith mini-batch is denoted by $\boldsymbol{r}^{(i)} = \boldsymbol{y}^{(i)} - [X^{(i)}]^T \boldsymbol{\beta}^{(i)}$, and $r^{(i)}_{\varphi(k)}$ represents the kth elements of $\boldsymbol{r}^{(i)}$ in ascending order of magnitude. The variable τ_o in the constraint is defined as

$$\tau_o = \underset{\lceil n/2 \rceil \le \tau \le n}{\arg\min} \left| \left(r^{(i)}_{\varphi(\tau)} \right)^2 - \frac{\|\boldsymbol{r}^{(i)}_{Z_{\tau'}}\|_2^2}{\tau'} \right|, \tag{4}$$

where $\tau' = \tau - \lceil n/2 \rceil$ and $Z_{\tau'}$ is the position set containing the smallest τ' elements in residual $\boldsymbol{r}^{(i)}$.

The design of the heuristic estimator follows a natural intuition that data points with unbounded corruption have a residual higher in magnitude than that of uncorrupted data. The constraint in Equation (3) ensures the residual of the largest element τ in our estimation cannot be larger than the residual of a smaller element τ_o. If the element τ_o is too small, some uncorrupted elements will be excluded from our estimation, but if the element is too large, some corrupted elements will be included. The formal definition of τ_o is shown in Equation (4), in which τ_o is defined as a value whose squared residual is closest to $\|\boldsymbol{r}^{(i)}_{Z_{\tau'}}\|_2^2/\tau'$, where τ' is less than the ground truth threshold τ_*. This design ensures that $|Z^{(i)}_* \cap Z^{(i)}_t| \ge \tau - n/2$, which means at least $\tau - n/2$ elements are correctly estimated in $Z^{(i)}_t$. In addition, the precision of the estimated uncorrupted set can be easily achieved when fewer elements are included in the estimation, but with low recall value. To increase the recall of our estimation, the objective function in Equation (3) chooses the maximum uncorrupted set size.

Applying the uncorrupted set size generated by $h(\cdot)$, the heuristic hard thresholding is defined as follows:

Definition 4.1 (Heuristic Hard Thresholding). Defining $\varphi_{\boldsymbol{r}}^{-1}(i)$ as the position of the ith element in residual vector $\boldsymbol{r}$'s ascending order of magnitude, the heuristic hard thresholding of $\boldsymbol{r}$ is defined as

$$\mathcal{H}(\boldsymbol{r}) = \{ i \in [n] : \varphi_{\boldsymbol{r}}^{-1}(i) \le h(\boldsymbol{r}) \}. \tag{5}$$

ALGORITHM 1: HRR Algorithm

```
  Input: Corrupted data samples $X \in \mathbb{R}^{p \times n}$ and response vector $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$ for single mini batch,
         tolerance $\epsilon$
  Output: solution $\hat{\boldsymbol{\beta}}$, $\hat{Z}$
1 $Z_0 = [\mathrm{n}]$, $t \leftarrow 0$
2 repeat
3     $\boldsymbol{\beta}^{t+1} \leftarrow (X_{Z_t} X_{Z_t}^T)^{-1} X_{Z_t} \boldsymbol{y}_{Z_t}$
4     $\boldsymbol{r}^{t+1} \leftarrow |\boldsymbol{y} - X^T \boldsymbol{\beta}^{t+1}|$
5     $Z_{t+1} \leftarrow \mathcal{H}(\boldsymbol{r}^{t+1})$, where $\mathcal{H}(\cdot)$ is defined in Equation (5).
6     $t \leftarrow t + 1$
7 until $\|\boldsymbol{r}^{t+1}_{Z_{t+1}} - \boldsymbol{r}^{t}_{Z_t}\|_2 < \epsilon n$
8 return $\boldsymbol{\beta}^{t+1}$, $Z_{t+1}$
```

The optimization of $Z^{(i)}$ is formulated as solving Equation (5), where the set returned by $\mathcal{H}(\boldsymbol{r}^{(i)})$ will be used to determine regression coefficients $\boldsymbol{\beta}^{(i)}$.

The details of the *HRR* algorithm are shown in Algorithm 1, which follows an intuitive strategy of updating regression coefficient $\boldsymbol{\beta}^{(i)}$ to provide a better fit for the current estimated uncorrupted set Z_t in Line 3, and updating the residual vector in Line 4. The uncorrupted set Z_{t+1} is estimated via heuristic hard thresholding in Line 5 based on residual vector $\boldsymbol{r}$ in the current iteration. The algorithm continues until the change in the residual vector falls within a small range.

4.2 Distributed Robust Regression

Given data samples $\{(X^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (X^{(m)}, \boldsymbol{y}^{(m)})\}$ in a sequence of mini-batches, a distributed robust regression algorithm, named *DRLR*, is proposed to optimize the robust regression coefficients in distributed approach without loading entire data at one time. Before we dive into the details of the *DRLR* algorithm, we provide some key definitions.

Definition 4.2 (Estimate Distance). Defining $\boldsymbol{\beta}^{(i)}$ and $\boldsymbol{\beta}^{(j)}$ as the estimate of the regression coefficients for the ith and jth mini-batches, respectively, the distance between the two estimates is defined as

$$d_{i,j} = \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(j)}\|_2. \tag{6}$$

Based on the definition of estimate distance, we define the distance vector of the ith mini-batch as $\boldsymbol{d}^{(i)} \in \mathbb{R}^{m \times 1}$, where m is the total number of batches and $\boldsymbol{d}^{(i)}_j$ represents the distance from the estimate of the ith batch to the jth batch ($1 \leq j \leq m$). We also define $\sigma_k(\boldsymbol{d}^{(i)})$ and $\delta_k(\boldsymbol{d}^{(i)})$ as the value and index of the kth smallest value in distance vector $\boldsymbol{d}^{(i)}$, respectively. For instance, if the third batch is the fifth smallest distance in $\boldsymbol{d}^{(i)}$ with $d^{(i)}_3 = 0.3$, then we have $\sigma_5(\boldsymbol{d}^{(i)}) = 0.3$ and $\delta_5(\boldsymbol{d}^{(i)}) = 3$.

Definition 4.3 (Pivot Batch). Given a set of mini-batch estimates $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ and defining $\boldsymbol{d}^{(i)}$ as the distance vector of the ith batch, the pth batch is defined as pivot batch if it satisfies:

$$p = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)}), \tag{7}$$

where $\tilde{m} = \lfloor m/2 \rfloor + 1$ is the upper number of half batches. By using the definition of *pivot batch*, we define the deterministic set as follows:

Definition 4.4 (Deterministic Set). Given a set of mini-batch estimates $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ and defining $\boldsymbol{d}^{(p)}$ as the distance vector of the pivot batch, the deterministic set Ψ is defined as

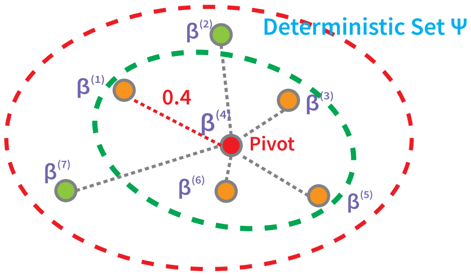


Fig. 2. Example of deterministic set detection.

ALGORITHM 2: Drlr Algorithm

```
**Input**: Corrupted data $\{(X^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (X^{(m)}, \boldsymbol{y}^{(m)})\}$ in $m$ mini batches, where $X^{(i)} \in \mathbb{R}^{p \times n}$ and
        $\boldsymbol{y}^{(i)} \in \mathbb{R}^{n \times 1}$.
**Output**: solution $\hat{\boldsymbol{\beta}}$
1 for $i = 1..m$ do
2     $\boldsymbol{\beta}^{(i)} \leftarrow \mathrm{HRR}(X^{(i)}, \boldsymbol{y}^{(i)})$
3 $p = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$ ;                                  // Optimize pivot batch p
4 $\Psi = \left\{\delta_k(\boldsymbol{d}^{(p)}) | 1 \leq k \leq \tilde{m}\right\}$ ;                        // Find deterministic set Ψ
5 $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{T} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}$ ;                // Robust consolidation
6 return $\hat{\boldsymbol{\beta}}$
```

follows:

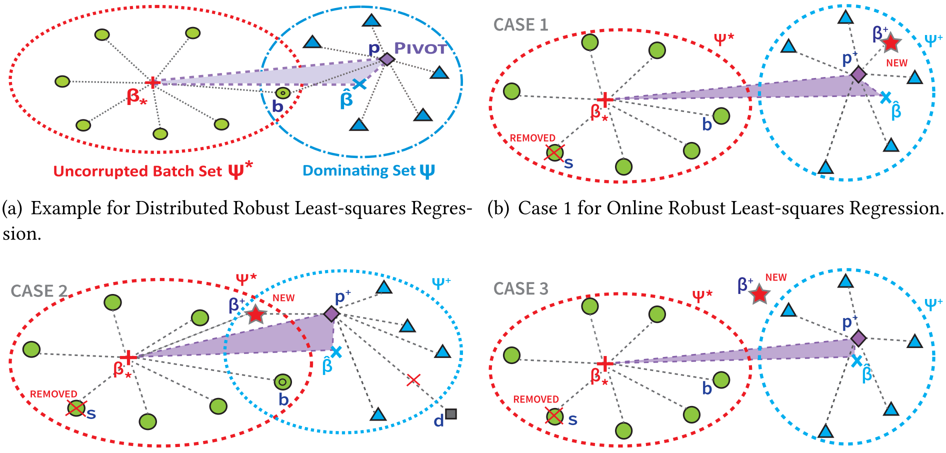
$$\Psi = \left\{\delta_k(\boldsymbol{d}^{(p)}) | 1 \leq k \leq \tilde{m}\right\}. \tag{8}$$

The deterministic set Ψ selects the smallest $\tilde{m}$ batches from the distance vector $\boldsymbol{d}^{(p)}$ of the pivot batch, which makes a small distance between the pivot batch and any batch $j \in \Psi$. Figure 2 shows the process of detecting the deterministic set in mini-batches. In Figure 2, the red circle node presents the pivot batch and the green dotted circle denotes the deterministic set Ψ, which contains $\tilde{m}$ batches represented by orange circle nodes. And the property will be used later in the proof of Lemma 5.4. Then, we define the general robust consolidation of a set of regression coefficients as follows:

Definition 4.5 (Robust Consolidation). Given a set of mini-batch estimates $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ and using Ψ to denote its deterministic set, the robust consolidation of the given estimates $\hat{\boldsymbol{\beta}}$ is defined as follows:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{T} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}. \tag{9}$$

The *DRLR* algorithm, shown in Algorithm 2, uses m mini-batches' data as input and outputs the consolidated estimate of regression coefficients $\hat{\boldsymbol{\beta}}$. First, the algorithm optimizes the coefficient estimate $\boldsymbol{\beta}^{(i)}$ of each mini-batch in Line 1–2, then it combines all the estimates of mini-batches in terms of overall robustness via distributed robust consolidation. Specifically, the algorithm determines the pivot batch based on all the estimates in Line 3 and generates the deterministic set Ψ in Line 4. Finally, all the batch estimates are combined via robust consolidation in Line 5. Figure 3(a) shows an example of distributed robust consolidation. The domination set Ψ contains $\tilde{m}$ closest batches to pivot batch p and the green circle node denotes the uncorrupted batch whose distance



(a) Example for Distributed Robust Least-squares Regression.

(b) Case 1 for Online Robust Least-squares Regression.

(c) Case 2 for Online Robust Least-squares Regression.

(d) Case 3 for Online Robust Least-squares Regression.

Fig. 3. Examples for DRLR and ORLR.

to ground truth coefficients $\boldsymbol{\beta}_*$ is less than a small error bound ε. We call the set containing all the green circle nodes as uncorrupted batch set Ψ^*. The example shows a case that only one uncorrupted batch b is contained in Ψ, which determines the distance between $\boldsymbol{\beta}_*$ and pivot batch p. The distance between $\boldsymbol{\beta}_*$ and $\hat{\boldsymbol{\beta}}$ is upper bounded by the summation of distance $d_{\boldsymbol{\beta}_*,p}$ and $d_{\hat{\boldsymbol{\beta}},p}$.

4.3 Online Robust Regression

The *DRLR* algorithm, proposed in Section 4.2, provides a distributed approach when a large amount of data has been collected. In this section, we present an online robust regression algorithm, named *ORLR*, that incrementally updates the robust estimate based on new incoming data. Specifically, suppose the regression coefficients of the previous m mini-batches $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ have been estimated by *DRLR*, the *ORLR* algorithm achieves an incremental update of robust consolidation $\hat{\boldsymbol{\beta}}$ when new incoming mini-batch data $X^+ \in \mathbb{R}^{p\times n}$ and $\boldsymbol{y}^+ \in \mathbb{R}^{n\times 1}$ are given.

The details of algorithm *ORLR* are shown in Algorithm 3. In Line 1, the regression coefficients $\boldsymbol{\beta}^+$ of the new data is optimized by *HRR* algorithm. The index of swapped estimate s is generated in Line 2 by selecting the minimum value from $[m] \setminus \Psi$, which represents the set of estimates that are not included in deterministic set Ψ. Since new estimates are appended to the tail of Π, the usage of minimum index ensures that the oldest corrupted estimate can be swapped out. In Line 3, the selected estimate $\boldsymbol{\beta}^{(s)}$ is removed from Π while the new estimate $\boldsymbol{\beta}^+$ is appended to the tail of Π. Lines 4 through 6 re-consolidate all the estimates based on newly updated Π in the same steps as the *DRLR* algorithm. It is important to note that the distance vectors used in Lines 4 and 5 are also updated corresponding to the new Π. Also, the *ORLR* algorithm can be invoked repeatedly for the incoming mini-batches, where the outputs Π and Ψ of the previous invocation can be used as the input of the next one.

Figure 3 shows three cases for *ORLR* algorithm. The first case shows the condition that the new estimate $\boldsymbol{\beta}^+ \in \Psi^+$ but not belongs to Ψ^*, and estimate s is removed. Although the estimate b is excluded from Ψ^+, the distance $d_{\boldsymbol{\beta}_*,p^+}$ can still be determined by the position of b. The error between $\boldsymbol{\beta}_*$ and $\hat{\boldsymbol{\beta}}$ can be increased, but still upper bounded by $d_{\boldsymbol{\beta}_*,p^+}$ and $d_{\hat{\boldsymbol{\beta}},p^+}$. In the second case, $\boldsymbol{\beta}^+ \in \{\Psi^* \cap \Psi^+\}$. Because the farthest node d in Ψ^+ is replaced by $\boldsymbol{\beta}^+$, the error between $\boldsymbol{\beta}_*$ and $\hat{\boldsymbol{\beta}}$ can be decreased, but it still upper bounded by the position of pivot batch p^+. The third case is

ALGORITHM 3: Orlr Algorithm

```
Input: New incoming corrupted data $X^+ \in \mathbb{R}^{p\times n}$ and $\boldsymbol{y}^+ \in \mathbb{R}^{n\times 1}$. Previous $m$ mini-batch estimates
       $\Pi = \{\boldsymbol{\beta}^{(1)}, \dots, \boldsymbol{\beta}^{(m)}\}$ and their corresponding $\Psi$.
Output: solution $\hat{\boldsymbol{\beta}}$, $\Pi$, $\Psi$
1 $\boldsymbol{\beta}^+ \leftarrow \mathrm{HRR}(X^+, \boldsymbol{y}^+)$
2 $s \leftarrow \min([m] \setminus \Psi)$                                        // Select removed estimate s
3 $\Pi^+ = \Pi \setminus \{\boldsymbol{\beta}^{(s)}\} \cup \{\boldsymbol{\beta}^+\}$
4 $p^+ = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$                        // Optimize new pivot batch $p^+$
5 $\Psi^+ = \left\{\delta_k(\boldsymbol{d}^{(p^+)}) | 1 \le k \le \tilde{m}\right\}$        // Find new deterministic set $\Psi^+$
6 $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{\tilde{m}} \sum_{i\in\Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}$   // Robust consolidation
7 return $\hat{\boldsymbol{\beta}}, \Pi^+, \Psi^+$
```

$\boldsymbol{\beta}^+ \notin \{\Psi^* \cup \Psi^+\}$, which is shown in Figure 3(d). The case is similar to the above two cases except a new estimate is added outside of Ψ^* and Ψ. However, the change will not impact the result of $\hat{\boldsymbol{\beta}}$.

4.4 Online Robust Least-Squares Regression under Biased-Batch Corruption

The *ORLR* algorithm updates the robust estimate based on new incoming data by removing the oldest one from $[m] \setminus \Psi$, which represents the set of estimates that are not included in deterministic set Ψ. In general, it can deal with the batched corruption, where corrupted mini-batches are arbitrarily distributed in the mini-batch set. However, when the data corruption is biased-batch corruption, it is infeasible to estimate by using *ORLR*. Table 3 shows the performance of regression coefficient recovery in different settings of biased-batch corruption. From Table 3, we find that the sequence of corrupted mini-batches is important to *ORLR*. When uncorrupted mini-batches arrive at first, it can establish better deterministic set. However, when corrupted mini-batches concentratedly arrive at first, it will establish the wrong deterministic set and cannot update it efficiently, which will effect the performance of the remaining data. Concretely, suppose more than half of the previous m mini-batches data are corrupted and concentratedly arrive at first, the previous pivot batch p and deterministic set Π will be established by corrupted data by using *ORLR*. Under the extreme circumstance, the regression coefficients cannot be updated due to new incoming mini-batch estimate $\boldsymbol{\beta}^+ \notin \{\Psi^* \cup \Psi^+\}$, which is the third case shown in Figure 3.

To solve these problems, we propose an online robust regression algorithm under biased-batch corruption, named *ORLR-BC*, which updates the deterministic set Π over time. Specifically, when determining the removed estimate, it considers the relevance to the deterministic set and the sequence of data as well. The residual value v_i of the estimate $\boldsymbol{\beta}^{(i)}$ is defined as follows:

$$v_i = \boldsymbol{\mu}\left(\frac{\|\hat{\boldsymbol{\beta}}' - \boldsymbol{\beta}^{(i)}\|_2}{\sum_{i\in\Pi'} \|\hat{\boldsymbol{\beta}}' - \boldsymbol{\beta}^{(i)}\|_2}\right) + \boldsymbol{\lambda}\left(\frac{i}{\sum_{i=1}^{m} i}\right), \tag{10}$$

where $\Pi' = \Pi \cup \{\boldsymbol{\beta}^+\}$, μ is a pivot related hyperparameter, λ is a time related hyperparameter, and $\hat{\boldsymbol{\beta}}$ is the robust consolidation in previous mini-batch.

The details of algorithm *ORLR-BC* are shown in Algorithm 4. In Line 1, the regression coefficients $\boldsymbol{\beta}^+$ of the new data is optimized by *HRR* algorithm. In Line 3, the robust consolidation $\hat{\boldsymbol{\beta}}'$ is obtained from previous mini-batch set Π. Lines 4 and 5 calculate the residual value of each estimate $\boldsymbol{\beta}^{(i)}$. The index of swapped estimate s is generated in Line 6 by selecting the minimum value from set $V = \{v_1, \dots, v_n\}$. Since new estimates are appended to the tail of Π, the usage of selective method ensures that the corrupted estimate can be swapped out. In Line 7, the selected estimate $\boldsymbol{\beta}^{(s)}$ is removed from Π while the new estimate $\boldsymbol{\beta}^+$ is appended to the tail of Π. Lines

ALGORITHM 4: ORLR-BC Algorithm

```
Input: New incoming corrupted data $X^+ \in \mathbb{R}^{p\times n}$ and $\boldsymbol{y}^+ \in \mathbb{R}^{n\times 1}$. Previous $m$ mini-batch estimates
       $\Pi = \{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ and their corresponding $\Psi$. Pivot related hyperparameters $\mu$ and time
       related hyperpatameters $\lambda$
Output: solution $\hat{\boldsymbol{\beta}}, \Pi, \Psi$
1  $\boldsymbol{\beta}^+ \leftarrow \mathrm{HRR}(X^+, \boldsymbol{y}^+)$
2  $\Pi' = \Pi \cup \{\boldsymbol{\beta}^+\}$
3  $\hat{\boldsymbol{\beta}}' = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{m} \sum_{i\in\Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}$    // Robust consolidation $\hat{\boldsymbol{\beta}}'$ in previous mini-batch
4  for $i = 1 \ldots m$ do
5      $v_i = \boldsymbol{\mu}(\frac{\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(i)}\|_2}{\sum_{i\in\Pi'}\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(i)}\|_2}) + \boldsymbol{\lambda}(\frac{i}{\sum_{i=1}^{m} i})$    // Update the residual value for each data sample.
6  $s \leftarrow \min(V)$    // Select removed estimate s
7  $\Pi^+ = \Pi \setminus \{\boldsymbol{\beta}^{(s)}\} \cup \{\boldsymbol{\beta}^+\}$
8  $p^+ = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$    // Optimize new pivot batch $p^+$
9  $\Psi^+ = \left\{\delta_k(\boldsymbol{d}^{(p^+)}) | 1 \leq k \leq \tilde{m}\right\}$    // Find new deterministic set $\Psi^+$
10 $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{\tilde{m}} \sum_{i\in\Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}$    // Robust consolidation
11 return $\hat{\boldsymbol{\beta}}, \Pi^+, \Psi^+$
```

8 through 10 re-consolidate all the estimates based on newly updated Π in the same steps as the *DRLR* algorithm. It is important to note that the distance vectors used in Lines 3, 5, and 10 are also updated corresponding to the new Π. Also, the *ORLR-BC* algorithm can be invoked repeatedly for the incoming mini-batches, where the outputs Π and Ψ of the previous invocation can be used as the input of the next one.

4.5 Complexity Analysis

In the *HRR* algorithm, for each iteration, $\boldsymbol{\beta}$ is computed for an estimated uncorrupted set, resulting in $O(p^3 + np^2)$ time complexity, where n is the number of samples in the set and p is the number of features. Then, computing $\boldsymbol{r}$ has a time complexity of $O(np)$. Finally, when updating the uncorrupted set Z_{t+1}, the time complexity of $H(\cdot)$ is $O(n)$. Therefore, if after t iterations *HRR* algorithm yields an ϵ-accurate solution, it results in $O(t(p^3 + np^2))$ time complexity, where $t = O(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}^{(i)}\|_2}{\sqrt{n}\epsilon})$ is proofed in Theorem 5.2 in Section 5. For *DRLR* algorithm, the estimated regression coefficient set is computed by *HRR*$(\cdot)$ over each mini-batch, resulting in $O(mt(p^3+np^2))$ complexity, where m is the number of mini-batches. Optimizing pivot batch results in $O(m^2p + m\log_2 m)$ and robust consolidation has a time complexity of $O(p)$. Thus, the computational complexity of *DRLR* algorithm is $O(mt(p^3 + np^2) + m^2p)$. Since *ORLR* and *ORLR-BC* are online algorithms, we analyse the computational complexity of each batch in the data streaming. For a new incoming data batch, estimated regression coefficient $\beta^{(i)}$ is computed by *HRR*$(\cdot)$ and results in an $O(t(p^3 + np^2) + m^2p)$ time complexity.

5 THEORETICAL RECOVERY ANALYSIS

In this section, the recovery properties of regression coefficients for the proposed distributed and online algorithms are presented in Theorems 5.5 and 5.6, respectively. Before that, the recovery property of *HRR* is presented in Theorem 5.2.

To prove the theoretical recovery of regression coefficients for a single mini-batch, we require that the least-squares function satisfies the ***Subset Strong Convexity** (SSC)* and ***Subset Strong Smoothness** (SSS)* properties, which are defined as follows:

Definition 5.1 (SSC and SSS Properties). The least-squares function $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T \boldsymbol{\beta}\|_2^2$ satisfies the $2\zeta_\gamma$-SSC property and $2\kappa_\gamma$-SSS property if the following holds:

$$\zeta_\gamma I \leq \frac{1}{2} \nabla^2 f_S(\boldsymbol{\beta}) \leq \kappa_\gamma I \quad \text{for } \forall S \in S_\gamma. \tag{11}$$

Note that Equation (11) is equivalent to

$$\zeta_\gamma \leq \min_{S \in S_\gamma} \lambda_{min}(X_S X_S^T) \leq \max_{S \in S_\gamma} \lambda_{max}(X_S X_S^T) \leq \kappa_\gamma, \tag{12}$$

where λ_{min} and λ_{max} denote the smallest and largest eigenvalues of matrix X, respectively.

Theorem 5.2 (HRR Recovery Property). *Let $X^{(i)} \in \mathbb{R}^{p \times n}$ be the given data matrix of the ith mini-batch and the corrupted response vector $\boldsymbol{y}^{(i)} = [\boldsymbol{X}^{(i)}]^T \boldsymbol{\beta}_* + \boldsymbol{u}^{(i)} + \boldsymbol{\varepsilon}^{(i)}$ with $\|\boldsymbol{u}^{(i)}\|_0 = \gamma n$. Let Σ_0 be an invertible matrix such that $\tilde{X}^{(i)} = \Sigma_0^{-1/2} X^{(i)}$; $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S^{(i)} - \tilde{X}_S^{(i)} \boldsymbol{\beta}\|_2^2$ satisfies the SSC and SSS properties at level α, γ with $2\zeta_{\alpha,\gamma}$ and $2\kappa_{\alpha,\gamma}$. If the data satisfies $\frac{\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}} < \frac{1}{2}$, after $t = O(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}^{(i)}\|_2}{\sqrt{n}\epsilon})$ iterations, Algorithm 1 yields an ϵ-accurate solution $\boldsymbol{\beta}_t^{(i)}$ with $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_t^{(i)}\|_2 \leq \epsilon + \frac{C\|\boldsymbol{\varepsilon}^{(i)}\|_2}{\sqrt{n}}$ for some $C > 0$.*

The proof of Theorem 5.2 can be found in the supplementary material.[1] The theoretical analyses of regression coefficients recovery for Algorithms 2 and 3 are as follows:

Lemma 5.3. *Suppose Algorithm 1 yields an ϵ-accurate solution $\hat{\boldsymbol{\beta}}$ with corruption ratio γ_0, and m mini-batches of data have a corruption ratio less than $\gamma_0/2$, more than $\lfloor \frac{m}{2} \rfloor + 1$ batches can yield an ϵ-accurate solution by Algorithm 1.*

Proof. Let Ψ_* denote the set of mini-batches that yield ϵ-accurate solutions and γ_i represent the corruption ratio for the ith mini-batch. Then, we have

$$\sum_{i \in [m] \backslash \Psi_*} \gamma_i n \overset{(a)}{\leq} \sum_i^m \gamma_i n = \frac{\gamma_0}{2} \cdot m \cdot n \tag{13}$$

$$(\gamma_0 n + 1)(m - |\Psi_*|) \overset{(b)}{\leq} \frac{\gamma_0}{2} \cdot m \cdot n.$$

Inequality *(a)* is based on $\sum_i^m \gamma_i n = \sum_{i \in \Psi_*} \gamma_i n + \sum_{i \in [m] \backslash \Psi_*} \gamma_i n$. And inequality *(a)* follows each corrupted mini-batch that contains at least $\gamma_0 n + 1$ corrupted samples. Applying simple algebra steps, we have

$$|\Psi_*| \geq m - \frac{\frac{\gamma_0}{2} mn}{\gamma_0 n + 1} \geq m - \frac{\frac{\gamma_0}{2} mn}{\gamma_0 n} \geq \frac{m}{2}. \tag{14}$$

Since $|\Psi_*|$ is an integer, then we have $|\Psi_*| \geq \lfloor \frac{m}{2} \rfloor + 1$. □

Lemma 5.4. *Given a set of mini-batch estimates $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ with $\tilde{m} = \lfloor m/2 \rfloor + 1$, defining the p^{th} batch as its pivot batch, then we have $\sigma_{\tilde{m}}(\boldsymbol{d}^{(p)}) \leq 2\epsilon$.*

[1] https://goo.gl/HRwZsp.

Proof. Suppose kth mini-batch is in the uncorrupted set Ψ_*, we have $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \leq \epsilon$. Similarly, for $\forall i \in \Psi_*$, we have $\|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}_*\|_2 \leq \epsilon$. According to the triangle inequality, for $\forall i \in \Psi_*$, it satisfies:

$$\begin{gathered} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 - \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \leq \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}_*\|_2 \leq \epsilon \\ \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 \leq 2\epsilon \end{gathered} . \tag{15}$$

Since $|\Psi_*| \geq \tilde{m}$, we have $\sigma_{\tilde{m}}(\boldsymbol{d}^{(k)}) \leq 2\epsilon$. According to the definition of pivot batch $p = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$, we have $\sigma_{\tilde{m}}(\boldsymbol{d}^{(p)}) \leq \sigma_{\tilde{m}}(\boldsymbol{d}^{(k)}) \leq 2\epsilon$. □

Theorem 5.5 (DRLR Recovery Property). *Given data samples in m mini-batches $\{(X^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (X^{(m)}, \boldsymbol{y}^{(m)})\}$ with a corruption ratio of $\gamma_0/2$, Algorithm 2 yields an ϵ-accurate solution $\hat{\boldsymbol{\beta}}$ with $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2 \leq 5\epsilon$.*

Proof. Let Ψ_* denotes the set of mini-batches that yield ϵ-accurate solutions. According to Lemma 5.3, we have $|\Psi_*| \geq \lfloor \frac{m}{2} \rfloor + 1$. Because of Lemma 5.4, we have $\forall i \in [1, \tilde{m}]$, $\sigma_i(\boldsymbol{d}^{(p)}) \leq 2\epsilon$, where p is the index of pivot batch and $\tilde{m} = \lfloor m/2 \rfloor + 1$. Using $\Psi = \{\delta_k(\boldsymbol{d}^{(p)}) | 1 \leq k \leq \tilde{m}\}$ defined in Algorithm 2, we have $\forall i, j \in \Psi$, $\|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(j)}\|_2 \leq 2\epsilon$. As $|\Psi_*| \geq \lfloor \frac{m}{2} \rfloor + 1$, we have $|\Psi_* \cap \Psi| \geq 1$. For any $k \in \{\Psi_* \cap \Psi\}$, we have the following two properties of the kth mini-batch: (1) $\forall i \in \Psi$, $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(i)}\|_2 \leq 2\epsilon$; and (2) $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \leq \epsilon$. Applying these properties, we get the error bound of $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2$ as follows:

$$\begin{aligned} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2 &= \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(k)} + \boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \\ &\overset{(a)}{\leq} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(k)}\|_2 + \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \\ &\overset{(b)}{\leq} \frac{1}{\tilde{m}} \sum_{i \in \Psi} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)}\|_2 + \frac{1}{\tilde{m}} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 + \epsilon \cdot \\ &\overset{(c)}{\leq} \frac{1}{\tilde{m}} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(i)}\|_2 + 3\epsilon \leq 5\epsilon \end{aligned} \tag{16}$$

Inequality *(a)* is based on the triangle inequality of the L_2 norm, and inequality *(b)* follows $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(k)}\|_2 = \frac{1}{T} \sum_{i \in \Psi} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)} + \boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2$. Inequity *(c)* follows the definition of $\hat{\boldsymbol{\beta}}$, which makes $\sum_{i \in \Psi} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)}\| \leq \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(i)}\|$. □

Theorem 5.6 (ORLR Recovery Property). *Given m mini-batch estimates of regression coefficients $\Pi = \{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$, their corresponding deterministic set Ψ, and incoming corrupted data $X^+ \in \mathbb{R}^{p \times n}$ and $\boldsymbol{y}^+ \in \mathbb{R}^{n \times 1}$, Algorithm 3 yields an ϵ-accurate solution $\hat{\boldsymbol{\beta}}$ with $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2 \leq 5\epsilon + \frac{4\epsilon}{\tilde{m}}$.*

Proof. Let e and s denote the index of added and removed mini-batch, respectively. According to Line 2 in Algorithm 3, the removed batch $s \notin \Psi$. As $|\Psi \cap \Psi_*| \geq 1$, there exists a mini-batch $k \in \{\Psi \cap \Psi_*\}$ that satisfies: (1) $\forall i \in \Psi$, $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(i)}\|_2 \leq 2\epsilon$; and (2) $\forall j \in \{\Psi^+ \setminus e\}$, $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(j)}\|_2 \leq 2\epsilon$. So we have

$$\begin{aligned} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(k)}\|_2 &= \frac{1}{\tilde{m}} \sum_{i \in \Psi^+} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)} + \boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 \\ &\overset{(a)}{\leq} \frac{1}{\tilde{m}} \sum_{i \in \Psi^+} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)}\|_2 + \frac{1}{\tilde{m}} \sum_{i \in \Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 . \\ &\overset{(b)}{\leq} \frac{2}{\tilde{m}} \sum_{i \in \Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(k)}\|_2 \end{aligned} \tag{17}$$

Inequality *(a)* is based on the triangle inequality of the L_2 norm, and inequality *(b)* follows the definition of $\hat{\boldsymbol{\beta}}$, which has $\sum_{i\in\Psi^+}\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(i)}\| \leq \sum_{i\in\Psi^+}\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}^{(i)}\|$.

Two conditions exist for added mini-batch e. For the condition $e \notin \Psi^+$, the new deterministic set $\Psi^+ = \Psi$. So $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(k)}\|_2 \leq \frac{2}{\tilde{m}}\sum_{i\in\Psi}\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}^{(k)}\|_2 \leq 4\epsilon$. For condition $e \in \Psi^+$, we have

$$\begin{aligned}\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(k)}\|_2 &\leq \frac{2}{\tilde{m}}\sum_{i\in\Psi^+}\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}^{(k)}\|_2 \\ &\overset{(c)}{\leq} \frac{2}{\tilde{m}}\left(\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}^{(e)}\|_2 + \sum_{i\in\{\Psi^+\cap\Psi\}}\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}^{(k)}\|_2\right) \\ &\overset{(d)}{\leq} \frac{4\epsilon}{\tilde{m}}(\tilde{m}-1) + \frac{2}{\tilde{m}}\left(\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}^{(p)}\|_2 + \|\boldsymbol{\beta}^{(p)}-\boldsymbol{\beta}^{(e)}\|_2\right) \\ &\overset{(e)}{\leq} \frac{4\epsilon}{\tilde{m}}(\tilde{m}-1) + \frac{8\epsilon}{\tilde{m}} \leq 4\epsilon + \frac{4\epsilon}{\tilde{m}}\end{aligned}. \tag{18}$$

Inequality *(c)* expands the set Ψ^+ into the new mini-batch e and set $\{\Psi^+\cap\Psi\}$. Inequality *(d)* uses the fact that $\forall i \in \Psi$, $\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}^{(i)}\|_2 \leq 2\epsilon$ and the triangle inequality of $\boldsymbol{\beta}^{(p)}$, where p is the pivot batch corresponding to Π. As $\max(\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}^{(p)}\|_2, \|\boldsymbol{\beta}^{(p)}-\boldsymbol{\beta}^{(e)}\|_2) \leq 2\epsilon$, inequality *(e)* is satisfied. Combining two conditions, we conclude $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(k)}\|_2 \leq 4\epsilon + \frac{4\epsilon}{\tilde{m}}$. Therefore, the error bound of $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}_*\|_2$ is as follows:

$$\begin{aligned}\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}_*\|_2 &\leq \|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^{(k)}\|_2 + \|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}_*\|_2 \\ &\overset{(f)}{\leq} 4\epsilon + \frac{4\epsilon}{\tilde{m}} + \epsilon \leq 5\epsilon + \frac{4\epsilon}{\tilde{m}}\end{aligned}. \tag{19}$$

Inequality *(f)* utilizes the fact that $\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}_*\|_2 \leq \epsilon$. Note that if $\tilde{m}$ is large enough, $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}_*\|_2 \lesssim 5\epsilon$, which is the same as the error bound in Theorem 5.5.

□

6 EXPERIMENT

In this section, the proposed algorithms *DRLR*, *ORLR*, and *ORLR-BC* are evaluated on both synthetic and real-world datasets. After the experiment setup has been introduced in Section 6.1, we present results on the effectiveness of the methods against several existing methods on both synthetic and real-world datasets, along with an analysis of efficiency for all the comparison methods. All the experiments were conducted on a 64-bit machine with an Intel(R) Core(TM) quad-core processor (Xeon W-2125@4.00GHz) and 64.0 GB memory. Details of both the source code and datasets used in the experiment can be downloaded here.[2]

6.1 Experiment Setup

6.1.1 Datasets and Labels. Our dataset is composed of synthetic and real-world data. For the synthetic data, the data samples were randomly generated according to the model in Equation (1) for each mini-batch, sampling a regression coefficient $\boldsymbol{\beta}_* \in \mathbb{R}^p$ as a random unit norm vector. The covariance data $X^{(i)}$ for each mini-batch was drawn independently and identically distributed from $\boldsymbol{x}_i \sim \mathcal{N}(0, I_p)$ and the uncorrupted response variables were generated as $\boldsymbol{y}_*^{(i)} = [\boldsymbol{X}^{(i)}]^T\boldsymbol{\beta}_* + \boldsymbol{\varepsilon}^{(i)}$, where the additive dense noise was $\varepsilon_i^{(i)} \sim \mathcal{N}(0, \sigma^2)$. The corrupted response vector for each mini-batch was generated as $\boldsymbol{y}^{(i)} = \boldsymbol{y}_*^{(i)} + \boldsymbol{u}^{(i)}$, where the corruption vector $\boldsymbol{u}^{(i)}$ was sampled from the

[2] https://goo.gl/b5qqYK.

uniform distribution $[-5\|\boldsymbol{y}_*^{(i)}\|_\infty, 5\|\boldsymbol{y}_*^{(i)}\|_\infty]$. The set of uncorrupted points $Z_*^{(i)}$ was selected as a uniformly random $\gamma^{(i)}$ n-sized subset of $[n]$, where $\gamma^{(i)}$ is the corruption ratio of the ith mini-batch. We define γ as the corruption ratio of the total m mini-batches; $\gamma^{(i)}$ is randomly chosen in the condition of $\gamma = \sum_i^m \gamma^{(i)}$, where γ should be less than $1/2$ to ensure the number of uncorrupted samples is greater than the number of corrupted ones. For biased-batch corruption, we synthesize the worst scenarios, in which the corrupted response vector for each mini-batch was generated as $\boldsymbol{y}'^{(i)} = \boldsymbol{y}_*^{(i)} + \boldsymbol{u}'^{(i)}$, where the corrupted vector $\boldsymbol{u}'^{(i)} = \left[X^{(i)}\right]^T \boldsymbol{\beta}'_* + \boldsymbol{\varepsilon}^{(i)}$ was generated to make the corrupted data have the uniform distribution, which mostly distracts the model.

The real-world datasets we use contain house rental transaction data from *New York City* and *Los Angeles* on Airbnb[3] website from January 2015 to October 2016. The datasets can be downloaded here.[4] For the *New York City* dataset, we use the first 321,530 data samples from January 2015 to December 2015 as training data and the remaining 329,187 samples from January to October 2016 as testing data. For the *Los Angeles* dataset, the first 106,438 samples from May 2015 to May 2016 are chosen as training data, and the remaining 103,711 samples are used as testing data.

6.1.2 Evaluation Metrics. For the synthetic data, we measured the performance of the regression coefficients recovery using the averaged L_2 error

$$e = \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2$$

where $\hat{\boldsymbol{\beta}}$ represents the recovered coefficients for each compared method and $\boldsymbol{\beta}_*$ is the ground truth regression coefficients. To compare the scalability of each method, the CPU running time for each of the competing methods was also measured. For the real-world dataset, we use the **mean absolute error** (**MAE**) to evaluate the performance of rental price prediction. Defining $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ as the predicted price and ground truth price, respectively, the MAE between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ can be presented as follows:

$$\mathrm{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right|$$

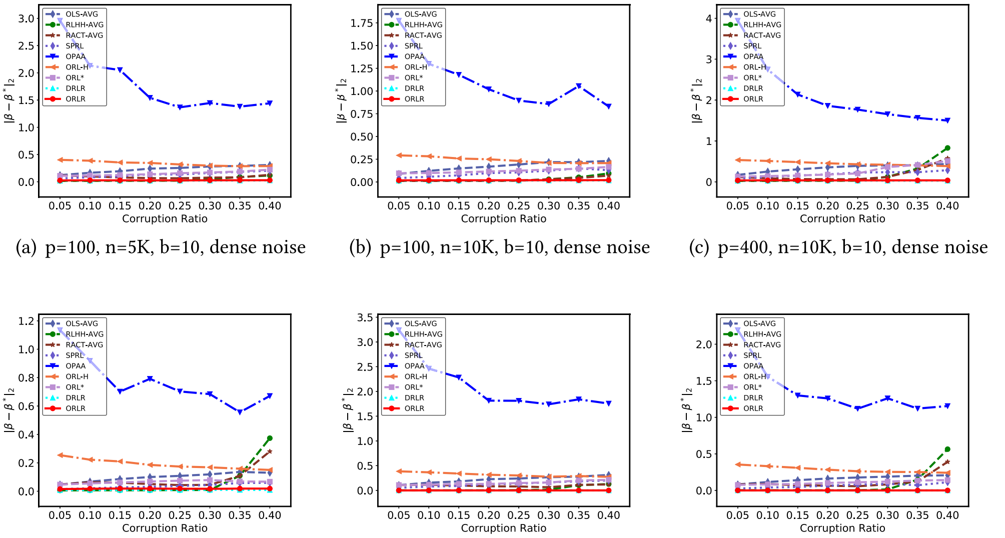
6.1.3 Implementation Details. For our proposed methods, we used *DRLR* and *ORLR* to evaluate our methods in both distributed and online settings. We evaluated *ORLR-BC* in biased-batch corruption online settings. For *ORLR*, we set the number of previous mini-batch estimates to seven if not specified. All the results from comparison methods were averaged over 10 runs. In each real-world dataset, after removing duplicated and missing records, we totally collected 650,717 rental transaction records for the *New York City* dataset and 210,149 records for the *Los Angeles* dataset. Furthermore, there are two types of features in the two datasets: categorical features and continuous features. We maintained all continuous features from original data and transferred all categorical features to integer type, including "neighborhood_cleansed", "property_type", "room_type", and "bed_type". In this way, we collected 21 features for each record.

6.2 Comparison Methods

The following methods are included in the performance comparison presented here: The ***averaged ordinary least-squares*** (***OLS-AVG***) method takes the average over the regression coefficients of each mini-batch, which is computed by the ordinary least-squares method. *RLHH-AVG* applies a robust method, ***Robust Least squares regression algorithm via Heuristic Hard thresholding (RLHH)*** [39], on each mini-batch and averages the regression coefficients of all the mini-batches.

[3] https://www.airbnb.com/.

[4] http://insideairbnb.com/get-the-data.html.



(a) p=100, n=5K, b=10, dense noise (b) p=100, n=10K, b=10, dense noise (c) p=400, n=10K, b=10, dense noise

(d) p=100, n=10K, b=30, dense noise (e) p=100, n=5K, b=10, no dense noise (f) p=200, n=10K, b=20, no dense noise

Fig. 4. Performance on regression coefficients recovery for different corruption ratios in uniform distribution.

Different from *OLS-AVG*, *RLHH-AVG* can estimate the corrupted samples in each mini-batch by a heuristic method. ***Robust regression via Adaptive Corruption Thresholding (RACT)*** [35] is an adaptive variation method for the robust least-squares regression problem. It improves the efficiency by estimating the corruption ratio based on adaptive searching steps without computing heuristic values for all the data samples. ***Self-paced robust learning (SPRL)*** *algorithm* [36] is a recently proposed self-paced learning method, which trains the model in a process from more reliable (clean) data instances to less reliable (noisy) ones under the supervision of well-labeled data. Since the corrupted samples are arbitrarily distributed among mini-batches, it is hard to choose the reliable data. We simply set the first two mini-batches as reliable data. The *OPAA* [8] is an online algorithm for adaptive linear regression, which updates the model incrementally for each new data sample. We set the threshold parameter ξ, which controls the inaccuracy sensitively, to 22. We also compared our method to an *ORL* approach [16], which addresses both the robustness and scalability issues in the regression problem. As the method requires a parameter for the corruption ratio, which is difficult to estimate in practice, we chose two versions with different parameter settings: *ORL** and *ORL-H*. *ORL** uses the true corruption ratio as its parameter, and *ORL-H* sets the outlier fraction λ to 0.5, which is a recommended setting in [16] if it is unknown.

6.3 Recovery of Regression Coefficients

We selected seven competing methods with which to evaluate the recovery performance of all the mini-batches: *OLS-AVG*, *RLHH-AVG*, *OPAA*, *ORL-H*, *ORL**, *DRLR*, and *ORLR*. Figure 4 shows the performance of coefficients recovery for different corruption ratios in uniform distribution. Specifically, Figures 4(a) and (b) show the recovery performance for different data sizes when the feature number is fixed. Looking at the results, we can conclude: (1) The *DRLR* and *ORLR* methods outperform all the competing methods, including *ORL**, whose corruption ratio parameter uses the ground truth value. Also, the error of the *ORLR* method has a small difference compared to *DRLR*, which indicates that the online robust consolidation performs as well as the distributed one. (2) The results of the *ORL* methods are significantly affected by their corruption ratio parameters; *ORL-H* performs almost three times as badly as *ORL** when the corruption ratio is less than 25%. When the corruption ratio increases, the error of *ORL-H* decreases because the actual corruption ratio is closer to 0.5, which is the estimated corruption ratio of *ORL-H*. However, both *DRLR* and *ORLR* perform consistently throughout, with no impact of the parameter. (3) *RLHH-AVG* and *RACT-AVG* have very competitive performance when the corruption ratio is less than 30% because almost no mini-batch contains corrupted samples larger than 50% when the corruption samples are randomly chosen. However, when the corruption ratio increases, some of the batches may contain large amounts of outliers, which makes some estimates be arbitrarily poor and break down the overall performance. Thus, although *RLHH-AVG* and *RACT-AVG* work well on mini-batches with fewer outliers, it cannot handle the case when the corrupted samples are arbitrarily distributed. (4) *SPRL* underperforms *ORL** because *SPRL* is affected by the reliability of the initial data. When the first mini-batch contains corrupted data, the model will be disguised and more noisy data will be introduced. However, it is hard to guarantee the initial data batch is clean when the corrupted samples are arbitrarily distributed. (5) *OPAA* generally exhibits worse performance than the other algorithms because the incremental update for each data sample makes it very sensitive to outliers. Figures 4(c) and (d) show the similar performance when the number of features and batches increases. Figures 4(e) and (f) show that both the *DRLR* and *ORLR* methods still outperform the other methods without dense noise, with both achieving an exact recovery of ground truth regression coefficients $\boldsymbol{\beta}_*$.

6.4 Performance on Different Corrupted Mini-Batches

Table 2 shows the performance of regression coefficient recovery in different settings of corrupted mini-batches, ranging from 0 to 8 corrupted mini-batches out of 20 mini-batches in total. Each corrupted mini-batch used in the experiment contains 90% corrupted samples and each uncorrupted mini-batch has 10% corrupted samples. We show the result of averaged L_2 error $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2$ in 10 different synthetic datasets with randomly ordered mini-batches. From the result in Table 2, we conclude: (1) When some mini-batches are corrupted, the *DRLR* method outperforms all the competing methods, and *ORLR* achieves the best performance compared to other online methods. (2) *RLHH-AVG* performs the best when no mini-batch is corrupted, but its recovery error is dramatically increased when the number of corrupted mini-batches increases. However, our methods perform consistently when the number of corrupted mini-batches increases. (3) *ORL** has competitive performance in different settings of corrupted mini-batches. However, its recovery error still increases two times when the number of corrupted mini-batches increases from two to eight. (4) When the corruption mini-batches are larger than one, *SPRL* outperforms the other competing methods except for DRLR and ORLR, because little corrupted data is contained in the initial mini-batch.

6.5 Performance on Different Settings of Biased-Batch Corruption

Table 3 shows the performance of biased-batch corruption coefficient recovery in different sequence of data mini-batches, and the corruption ratio was from 0% to 40% in total. As for the sequence settings, it was mainly divided into three parts: corrupted mini-batches arriving last, randomly, and first. Each corrupted mini-batch used in the experiment contains 90% corrupted samples and each uncorrupted mini-batch has 10% corrupted samples. We show the result of averaged L_2 error $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2$ in 10 different synthetic datasets. From the result in Table 3, we conclude: (1) When some mini-batches are corrupted, the *DRLR* method outperforms all the competing methods, and *ORLR-BC* achieves the best performance compared to other online methods. (2) Although *ORLR* has competitive performance in the corrupted batches arriving last settings, its recovery error increases dramatically when the corrupted batches arrive first. It is reasonable that *ORLR* has

Table 2. Performance on Regression Coefficients Recovery in Different Corrupted Mini-Batches

	0/20	**1/20**	**2/20**	**4/20**	**6/20**	**8/20**
OLS-AVG	0.120	0.136	0.142	0.172	0.188	0.211
RLHH-AVG	**0.011**	0.181	0.238	0.357	0.424	0.479
RACT-AVG	0.012	0.176	0.215	0.336	0.392	0.474
SPRL	0.043	0.048	0.067	0.059	0.076	0.110
OPAA	1.271	1.398	1.393	1.431	1.489	1.532
ORL-H	0.347	0.357	0.362	0.387	0.412	0.434
ORL*	0.078	0.080	0.089	0.139	0.226	0.347
ORLR	0.025	**0.026**	**0.027**	**0.026**	**0.026**	**0.026**
DRLR	0.015	**0.015**	**0.015**	**0.015**	**0.015**	**0.015**

Bold indicates the best performance among all methods.

Table 3. Performance on Regression Coefficients Recovery in Different Settings of Biased-Batch Corruption

	corrupted last					**random order**					**corrupted first**				
	0%	10%	20%	30%	40%	0%	10%	20%	30%	40%	0%	10%	20%	30%	40%
OLS-AVG	0.089	0.109	0.108	0.108	0.107	0.071	0.127	0.186	0.241	0.296	0.071	0.125	0.178	0.232	0.297
RLHH-AVG	0.055	0.098	0.168	0.228	0.300	0.034	0.100	0.168	0.234	0.300	0.034	0.100	0.163	0.226	0.301
RACT-AVG	0.055	0.097	0.167	0.228	0.301	0.033	0.098	0.165	0.235	0.300	0.033	0.099	0.165	0.237	0.303
SPRL	0.089	0.112	0.107	0.107	0.112	0.071	0.125	0.182	0.242	0.298	0.070	0.126	0.181	0.245	0.301
OPAA	5.181	5.145	5.166	5.158	5.161	5.153	5.221	5.141	5.193	5.203	5.140	5.147	5.247	5.134	5.165
ORL-H	0.280	0.295	0.294	0.291	0.292	0.278	0.298	0.316	0.357	0.397	0.273	0.284	0.298	0.312	0.343
ORL*	0.108	0.142	0.186	0.190	0.229	0.083	0.113	0.154	0.224	0.341	0.083	0.094	0.107	0.149	0.248
ORLR	0.042	0.043	0.143	0.167	0.203	0.038	0.038	0.038	0.237	0.168	0.042	0.043	0.705	0.704	0.703
DRLR	**0.035**	**0.035**	**0.035**	**0.035**	**0.035**	**0.034**	**0.034**	**0.034**	**0.034**	**0.034**	**0.034**	**0.035**	**0.035**	**0.035**	**0.035**
ORLR-BC	**0.044**	**0.042**	**0.042**	**0.042**	**0.042**	**0.037**	**0.037**	**0.038**	**0.037**	**0.038**	**0.038**	**0.038**	**0.038**	**0.038**	**0.038**

Bold indicates the best performance among all methods.

time-vary characteristic. But the error of the *ORLR-BC* method has a small difference compared to *DRLR*, which means it depends less on the sequence of data flow. (3) Both *ORL** and *OLS-AVG* have competitive performance in different settings of biased-batch corruption, but their recovery error is dramatically increased when the number of corrupted batches increases. However, our methods perform consistently when the number of corrupted batches increases. (4) *RLHH-AVG* and *RACT-AVG* are not sensitive to the sequence of corrupted data because they estimate the coefficient of each mini-batch independently. However, their recovery error is dramatically increased when the corrupted ratio is larger than 30%. (5) *SPRL* achieves competitive performance in the corrupted last setting but has poor performance in the other two settings. Since *SPRL* applies prior knowledge to the self-paced training process, it relies on the quality of the initial data. In contrast, our methods can adapt to new coming data even if corrupted batches arrive first.

6.6 Performance on Biased-Batch Corruption with Different Corruption Ratios

Table 4 shows the performance of regression coefficient recovery on biased-batch corruption with different corruption ratios. Each corrupted mini-batch used in the experiment contains 90% corrupted samples and each uncorrupted mini-batch has 10% corrupted samples. We show the result of averaged L_2 error $||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*||_2$ in 10 different synthetic datasets with corrupted-first ordered mini-batches, which is the worst scenario in our online setting. From the result in Table 4, we conclude: (1) When some batches are biased-batch corrupted, the *DRLR* method outperforms all the competing methods, and *ORLR-BC* achieves the best performance compared to other online methods.

Table 4. Performance on Regression Coefficients Recovery on Biased-Batch Corruption with Different Corruption Ratios

	p = 100, n = 5K, b = 20					**p = 100, n = 5K, b = 30**				
	5%	10%	20%	30%	40%	5%	10%	20%	30%	40%
OLS-AVG	0.099	0.127	0.185	0.241	0.298	0.090	0.125	0.182	0.245	0.299
RLHH-AVG	0.068	0.101	0.168	0.235	0.302	0.057	0.100	0.166	0.238	0.302
RACT-AVG	0.068	0.101	0.164	0.233	0.300	0.056	0.090	0.165	0.237	0.301
SPRL	0.101	0.129	0.180	0.239	0.296	0.090	0.125	0.183	0.245	0.299
OPAA	5.656	5.555	5.504	5.571	5.525	5.453	5.378	5.423	5.469	5.432
ORL-H	0.403	0.404	0.423	0.441	0.451	0.389	0.401	0.413	0.429	0.445
ORL*	0.104	0.119	0.149	0.218	0.335	0.093	0.105	0.137	0.207	0.330
ORLR	0.042	0.043	0.705	0.704	0.703	0.043	0.042	0.695	0.716	0.705
DRLR	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**	**0.036**	**0.036**	**0.036**	**0.036**	**0.036**
ORLR-BC	**0.041**	**0.043**	**0.044**	**0.044**	**0.043**	**0.042**	**0.042**	**0.043**	**0.042**	**0.043**
	p = 100, n = 10K, b = 20					**p = 100, n = 5K, b = 40**				
	5%	10%	20%	30%	40%	5%	10%	20%	30%	40%
OLS-AVG	0.102	0.122	0.185	0.236	0.294	0.097	0.128	0.191	0.232	0.287
RLHH-AVG	0.068	0.098	0.168	0.230	0.298	0.067	0.101	0.172	0.227	0.292
RACT-AVG	0.067	0.097	0.166	0.229	0.297	0.066	0.100	0.172	0.226	0.291
SPRL	0.102	0.122	0.185	0.237	0.295	0.098	0.129	0.192	0.232	0.288
OPAA	5.285	5.264	5.310	5.307	5.316	5.304	5.333	5.315	5.325	5.397
ORL-H	0.403	0.296	0.317	0.324	0.353	0.391	0.392	0.407	0.414	0.437
ORL*	0.099	0.099	0.122	0.159	0.253	0.089	0.097	0.128	0.197	0.313
ORLR	0.038	0.039	0.703	0.689	0.695	0.043	0.709	0.731	0.679	0.677
DRLR	**0.035**	**0.036**	**0.035**	**0.035**	**0.035**	**0.036**	**0.035**	**0.035**	**0.036**	**0.036**
ORLR-BC	**0.037**	**0.039**	**0.037**	**0.038**	**0.038**	**0.044**	**0.041**	**0.042**	**0.044**	**0.044**
	p = 100, n = 20K, b = 20					**p = 1000, n = 5K, b = 20**				
	5%	10%	20%	30%	40%	5%	10%	20%	30%	40%
OLS-AVG	0.099	0.131	0.187	0.240	0.303	0.090	0.132	0.188	0.243	0.298
RLHH-AVG	0.063	0.095	0.155	0.213	0.279	0.082	0.110	0.175	0.240	0.306
RACT-AVG	0.062	0.094	0.155	0.213	0.279	0.081	0.109	0.174	0.239	0.306
SPRL	0.098	0.131	0.187	0.24	0.304	0.107	0.131	0.189	0.244	0.300
OPAA	5.114	5.098	5.142	5.133	5.084	6.782	6.795	6.776	6.817	6.806
ORL-H	0.203	0.209	0.221	0.239	0.276	0.779	0.786	0.788	0.795	0.796
ORL*	0.087	0.093	0.103	0.131	0.197	0.234	0.298	0.438	0.576	0.701
ORLR	0.035	0.035	0.647	0.634	0.651	0.100	0.100	0.706	0.704	0.704
DRLR	**0.034**	**0.034**	**0.034**	**0.034**	**0.034**	**0.067**	**0.067**	**0.067**	**0.068**	**0.068**
ORLR-BC	**0.035**	**0.035**	**0.034**	**0.035**	**0.035**	**0.100**	**0.100**	**0.101**	**0.100**	**0.101**

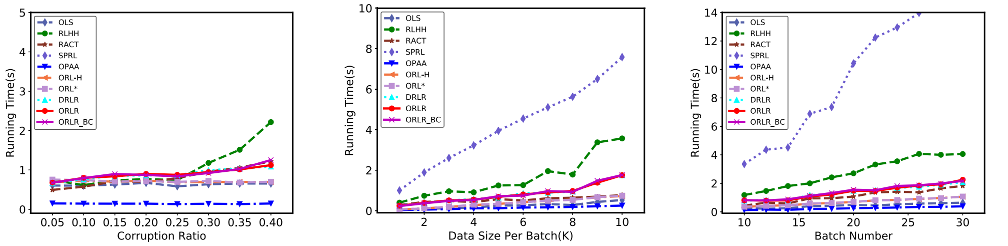
Bold indicates the best performance among all methods.

(2) When the number of corrupted mini-batches is smaller, the performance of *ORLR* is competitive. But when the number of corrupted batches increases, its recovery error is dramatically increased. Instead, *ORLR-BC* performs consistently when the number of corrupted batches increases. (3) *ORL**, *RLHH-AVG*, *RACT-AVG*, and *SPRL* have competitive performance in different settings of corrupted batches. However, their recovery error still increases two times when the number of corrupted batches increases from two to eight. (4) When the number of feature increases, all methods have higher recovery error. However, the performance of *DRLR* and *ORLR-BC* were still best and the recovery error increased slowly.

Table 5. MAE of Rental Price Prediction

	New York City (Corruption Ratio)					
	5%	**10%**	**20%**	**30%**	**40%**	**Avg.**
OLS-AVG	3.256 ± 0.449	3.519 ± 0.797	3.976 ± 0.786	4.230 ± 1.292	4.356 ± 1.582	3.867 ± 0.981
RLHH-AVG	**2.823 ± 0.000**	2.824 ± 0.000	13.092 ± 25.354	35.184 ± 37.426	42.713 ± 19.304	19.327 ± 16.417
RACT-AVG	2.823 ± 0.001	**2.822 ± 0.001**	12.351 ± 23.984	33.823 ± 34.259	39.548 ± 20.728	18.273 ± 15.795
SPRL	5.076 ± 0.391	5.524 ± 0.573	5.594 ± 0.420	5.843 ± 0.387	5.957 ± 0.703	5.599 ± 0.495
OPAA	91.287 ± 51.475	100.864 ± 72.239	121.087 ± 64.618	92.735 ± 38.063	152.479 ± 57.553	111.690 ± 56.790
ORL-H	6.832 ± 0.004	6.828 ± 0.007	6.732 ± 0.240	6.803 ± 0.107	6.573 ± 0.189	6.754 ± 0.109
ORL*	6.538 ± 0.293	6.384 ± 0.274	6.394 ± 0.208	6.406 ± 0.180	6.471 ± 0.190	6.439 ± 0.229
DRLR	2.824 ± 0.000	2.824 ± 0.000	**2.823 ± 0.000**	3.185 ± 0.523	4.342 ± 1.784	3.200 ± 0.461
ORLR	2.824 ± 0.001	2.824 ± 0.000	**2.823 ± 0.000**	**2.883 ± 0.187**	**3.563 ± 0.935**	**2.983 ± 0.225**
	Los Angeles (Corruption Ratio)					
	5%	**10%**	**20%**	**30%**	**40%**	**Avg.**
OLS-AVG	4.641 ± 0.664	4.876 ± 0.948	5.607 ± 1.349	6.199 ± 1.443	6.797 ± 2.822	5.624 ± 1.445
RLHH-AVG	**3.994 ± 0.002**	3.998 ± 0.003	4.092 ± 0.290	28.788 ± 47.322	30.414 ± 35.719	14.257 ± 16.667
RACT-AVG	**3.994 ± 0.002**	**3.997 ± 0.041**	4.079 ± 0.470	25.502 ± 42.743	27.485 ± 33.428	13.011 ± 15.329
SPRL	5.012 ± 0.351	5.571 ± 0.528	5.627 ± 0.839	5.912 ± 0.641	5.803 ± 0.423	6.084 ± 0.593
OPAA	150.668 ± 52.344	209.298 ± 124.058	113.267 ± 44.270	121.880 ± 55.938	146.425 ± 104.995	148.308 ± 76.321
ORL-H	6.819 ± 0.045	6.745 ± 0.039	6.667 ± 0.084	6.619 ± 0.300	6.317 ± 0.394	6.633 ± 0.172
ORL*	6.257 ± 0.497	6.303 ± 0.304	6.415 ± 0.172	6.308 ± 0.377	6.186 ± 0.531	6.294 ± 0.376
DRLR	3.995 ± 0.005	3.999 ± 0.008	**3.993 ± 0.003**	4.837 ± 1.108	6.336 ± 2.388	4.632 ± 0.702
ORLR	3.997 ± 0.008	3.999 ± 0.009	3.994 ± 0.004	**4.466 ± 1.141**	**5.802 ± 1.990**	**4.452 ± 0.630**

Bold indicates the best performance among all methods.



(a) p=200, n=5K, b=10, no dense noise (b) p=100, cr=0.4, b=10, dense noise (c) p=100, cr=0.4, n=5K, dense noise

Fig. 5. Running time for different corruption ratios and data sizes.

6.7 Result of Rental Price Prediction

To evaluate the robustness of our proposed methods in a real-world dataset, we compared the performance of rental price prediction in different corruption settings, ranging from 5% to 40%. The additional corruption was sampled from the uniform distribution $[-0.5|\boldsymbol{y}_i|, 0.5|\boldsymbol{y}_i|]$, where $|\boldsymbol{y}_i|$ represents the absolute price value of the ith sample data. Table 5 shows the MAE of rental price prediction and its corresponding standard deviation from 10 runs in the *New York City* and *Los Angeles* datasets. From the result, we can conclude: (1) The *DRLR* and *ORLR* methods outperform all the other methods in different corruption settings except when the corruption ratio is less than 10%. (2) The *RLHH-AVG* method performs the best when the corruption ratio is less than or equal to 10%. However, as the corruption ratio rises, the error increases dramatically because some mini-batches are entirely corrupted. (3) The *OLS-AVG* method has a very competitive performance in all the corruption settings because the deviation of sampled corruption is small, which is less than 50% from the labeled data. (4) *SPRL* performs almost two times worse than *ORLR* because it is affected by the corrupted data contained in the first mini-batch.

6.8 Efficiency

To evaluate the efficiency of our proposed method, we compared the performance of all the competing methods for three different data settings: different corruption ratios, data sizes per mini-batch, and batch numbers. In general, as Figure 5 shows, we can conclude: (1) The *OPAA* method outperforms the other methods in the three different settings because it does not consider the robustness of the data. Also, the *ORL-H* and *ORL** methods have performed similarly to *OPAA* method, as they use fixed corruption ratios without taking additional steps to estimate the corruption ratio. (2) The *DRLR* and *ORLR* methods have very competitive performance even though they take additional corruption estimation and robust consolidation steps for each mini-batch. Moreover, with increases of the corruption ratio, data size per batch, and batch number, the running time of both the *DRLR* and *ORLR* methods increases linearly, which is an important characteristic for the two methods to be extended to a large scale problem. (3) Our methods outperform the *RLHH* method although it only estimates the corruption for each mini-batch but ignores the overall robustness, which indicates that the corruption estimation step in our method performs more efficiently than that in *RLHH.* (4) The running time for *SPRL* increases dramatically when the data size or batch number increases, which means our methods are more efficient than the self-paced-learning approach.

7 CONCLUSION

In this article, distributed and two online robust regression algorithms, *DRLR*, *ORLR,* and *ORLR-BC*, are proposed to handle the scalable least-squares regression problem in the presence of extremely noisy labels. To achieve this, we proposed a heuristic hard thresholding method to estimate the corruption set for each mini-batch and designed both online and distributed robust consolidation methods to ensure the overall robustness. We demonstrate that our algorithms can yield a constant upper bound on the coefficient recovery error of state-of-the-art robust regression methods. Extensive experiments on both synthetic data and real-world rental price data demonstrated that the proposed algorithms outperform the effectiveness of other comparable methods with competitive efficiency.

REFERENCES

[1] Jean-Yves Audibert and Olivier Catoni. 2011. Robust linear least squares regression. *The Annals of Statistics* 39, 5 (2011), 2766–2794.

[2] Markus Baldauf and J. M. C. Santos Silva. 2012. On the use of robust regression in econometrics. *Economics Letters* 114, 1 (2012), 124–127.

[3] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. 2013. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science* 59, 2 (2013), 341–357.

[4] Kush Bhatia, Prateek Jain, and Purushottam Kar. 2015. Robust regression via hard thresholding. In *Proceedings of the 28th International Conference on Neural Information Processing Systems.* 721–729.

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

[6] Yudong Chen and Constantine Caramanis. 2013. Noisy and missing data regression: Distribution-oblivious support recovery. In *Proceedings of the 30th International Conference on International Conference on Machine Learning.* PMLR, 383–391.

[7] Yudong Chen, Constantine Caramanis, and Shie Mannor. 2013. Robust sparse regression under adversarial corruption. In *Proceedings of the 30th International Conference on Machine Learning.* PMLR, 774–782.

[8] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7, Mar (2006), 551–585.

[9] C. Cromvik and M. Patriksson. 2010. On the robustness of global optima and stationary solutions to stochastic mathematical programs with equilibrium constraints, Part 1: Theory. *Journal of Optimization Theory and Applications* 144, 3 (2010), 461–478.

[10] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Communications of the ACM* 51, 1 (2008), 107–113.

[11] Erick Delage and Yinyu Ye. 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research* 58, 3 (2010), 595–612. DOI: https://doi.org/10.1287/opre.1090.0741

[12] Xuan Vinh Doan, Serge Kruk, and Henry Wolkowicz. 2012. A robust algorithm for semidefinite programming. *Optimization Methods and Software* 27, 4–5 (2012), 667–693.

[13] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[14] Jitka Dupačová and Miloš Kopa. 2012. Robustness in stochastic programs with risk constraints. *Annals of Operations Research* 200, 1 (2012), 55–74.

[15] Yaakov Engel, Shie Mannor, and Ron Meir. 2004. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing* 52, 8 (2004), 2275–2285.

[16] Jiashi Feng, Huan Xu, and Shie Mannor. 2017. Outlier robust online learning. arXiv:1701.00251. Retrieved from http://arxiv.org/abs/1701.00251.

[17] Chao Huang, Baoxu Shi, Xuchao Zhang, Xian Wu, and Nitesh V. Chawla. 2019. Similarity-aware network embedding with self-paced learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2113–2116.

[18] Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V. Chawla. 2019. Online purchase prediction via multi-scale modeling of behavior dynamics. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2613–2622.

[19] Seong-Cheol Kang, Theodora S. Brisimi, and Ioannis Ch Paschalidis. 2015. Distribution-dependent robust linear optimization with applications to inventory control. *Annals of Operations Research* 231, 1 (2015), 229–263.

[20] Po-Ling Loh and Martin J. Wainwright. 2011. High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*. 2726–2734.

[21] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research* 11, Jan (2010), 19–60.

[22] R. A. R. D. Maronna, R. Douglas Martin, and Victor Yohai. 2006. *Robust Statistics*. John Wiley & Sons, Chichester.

[23] G. Mateos, J. A. Bazerque, and G. B. Giannakis. 2010. Distributed sparse linear regression. *IEEE Transactions on Signal Processing* 58, 10 (Oct 2010), 5262–5276. DOI: https://doi.org/10.1109/TSP.2010.2055862

[24] Brian McWilliams, Gabriel Krummenacher, Mario Lucic, and Joachim M. Buhmann. 2014. Fast and robust least squares estimation in corrupted linear models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. 415–423.

[25] Nam H. Nguyen and Trac D. Tran. 2013. Exact recoverability from dense corrupted observations via L1-Minimization. *IEEE Transactions on Information Theory* 59, 4 (2013), 2017–2035.

[26] Mathieu Rosenbaum and Alexandre B. Tsybakov. 2010. Sparse recovery under matrix uncertainty. *The Annals of Statistics* 38, 5 (2010), 2620–2651.

[27] Peter J. Rousseeuw and Annick M. Leroy. 2005. *Robust Regression and Outlier Detection*. Vol. 589, John Wiley & Sons.

[28] B. Saltzberg. 1967. Performance of an efficient parallel data transmission system. *IEEE Transactions on Communication Technology* 15, 6 (1967), 805–811.

[29] Shekhar Sharma, Swanand Khare, and Biao Huang. 2016. Robust online algorithm for adaptive linear regression parameter estimation and prediction. *Journal of Chemometrics* 30, 6 (2016), 308–323. DOI: https://doi.org/10.1002/cem.2792

[30] Yiyuan She and Art B. Owen. 2011. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association* 106, 494 (2011), 626–639. Retrieved from http://www.jstor.org/stable/41416397.

[31] John Wright and Yi Ma. 2010. Dense error correction via L1-minimization. *IEEE Transactions on Information Theory* 56, 7 (July 2010), 3540–3560. DOI: https://doi.org/10.1109/TIT.2010.2048473

[32] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. 2014. Internet of things for smart cities. *IEEE Internet of Things Journal* 1, 1 (2014), 22–32.

[33] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 6845–6852.

[34] Xuchao Zhang, Shuo Lei, Liang Zhao, Arnold Boedihardjo, and Chang-Tien Lu. 2018. Robust regression via online feature selection under adversarial data corruption. In *Proceedings of the 2018 IEEE International Conference on Data Mining*. IEEE, 1440–1445.

[35] Xuchao Zhang, Shuo Lei, Liang Zhao, Arnold P. Boedihardjo, and Chang-Tien Lu. 2019. Robust regression via heuristic corruption thresholding and its adaptive estimation variation. *ACM Transactions on Knowledge Discovery from Data* 13, 3 (2019), 1–22.

[36] Xuchao Zhang, Xian Wu, Fanglan Chen, Liang Zhao, and Chang-Tien Lu. 2020. Self-paced robust learning for leveraging clean labels in noisy data. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, 6853–6860.

[37] Xuchao Zhang, Liang Zhao, Arnold P. Boedihardjo, and Chang-Tien Lu. 2017. Online and distributed robust regressions under adversarial data corruption. In *Proceedings of the 2017 IEEE International Conference on Data Mining*. 625–634. DOI: https://doi.org/10.1109/ICDM.2017.72

[38] Xuchao Zhang, Liang Zhao, Arnold P. Boedihardjo, and Chang-Tien Lu. 2017. Online and distributed robust regressions under adversarial data corruption. In *Proceedings of the 2017 IEEE International Conference on Data Mining*. 625–634.

[39] Xuchao Zhang, Liang Zhao, Arnold P. Boedihardjo, and Chang-Tien Lu. 2017. Robust regression via heuristic hard thresholding. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* . AAAI. Retrieved from http://dl.acm.org/citation.cfm?id=3060832.3060872.

[40] A. M. Zoubir, V. Koivunen, Y. Chakhchoukh, and M. Muma. 2012. Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts. *IEEE Signal Processing Magazine* 29, 4 (July 2012), 61–80. DOI : https://doi.org/10.1109/MSP.2012.2183773

Received October 2020; revised April 2021; accepted June 2021