Robust Regression via Heuristic Corruption Thresholding and Its Adaptive Estimation Variation

XUCHAO ZHANG and SHUO LEI, Virginia Tech
LIANG ZHAO, George Mason University
ARNOLD P. BOEDIHARDJO, U. S. Army Corps of Engineers
CHANG-TIEN LU, Virginia Tech

The presence of data noise and corruptions has recently invoked increasing attention on robust least-squares regression (*RLSR*), which addresses this fundamental problem that learns reliable regression coefficients when response variables can be arbitrarily corrupted. Until now, the following important challenges could not be handled concurrently: (1) rigorous recovery guarantee of regression coefficients, (2) difficulty in estimating the corruption ratio parameter, and (3) scaling to massive datasets. This article proposes a novel Robust regression algorithm via Heuristic Corruption Thresholding (*RHCT*) that concurrently addresses all the above challenges. Specifically, the algorithm alternately optimizes the regression coefficients and estimates the optimal uncorrupted set via heuristic thresholding without a pre-defined corruption ratio parameter until its convergence. Moreover, to improve the efficiency of corruption estimation in large-scale data, a Robust regression algorithm via Adaptive Corruption Thresholding (*RACT*) is proposed to determine the size of the uncorrupted set in a novel adaptive search method without iterating data samples exhaustively. In addition, we prove that our algorithms benefit from strong guarantees analogous to those of state-of-the-art methods in terms of convergence rates and recovery guarantees. Extensive experiments demonstrate that the effectiveness of our new methods is superior to that of existing methods in the recovery of both regression coefficients and uncorrupted sets, with very competitive efficiency.

CCS Concepts: • **Computing methodologies** → *Machine learning algorithms*;

Additional Key Words and Phrases: Robust regression, hard thresholding, adversarial data corruption, adaptive search, discrete optimization

ACM Reference format:
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 Trans. Knowl. Discov. Data* 13, 3, Article 28 (June 2019), 22 pages.
https://doi.org/10.1145/3314105

This work is supported in part by the U. S. Military Research Laboratory and the U. S. Military Research Office under contract number W911NF-12-1-0445.

Authors' addresses: X. Zhang, S. Lei, and C.-T. Lu, Department of Computer Science, Virginia Tech, 7054 Haycock Road, Falls Church, VA, 22043; emails: {xuczhang, slei, ctlu}@vt.edu; L. Zhao, Department of Information Science and Technology, George Mason University, 4400 Univ. Dr, Fairfax, VA, 22043; A. P. Boedihardjo, U.S. Army Corps of Engineers, 7701 Telegraph Rd, Alexandria, VA, 22315; email: arnold.p.boedihardjo@usace.army.mil.

1556-4681/2019/06-ART28 $15.00
https://doi.org/10.1145/3314105

1 INTRODUCTION

The presence of noise and corruption in real-world data can be inevitably caused by experimental errors, accidental outliers, or even adversarial data attacks. As traditional least squares regression methods are vulnerable to outlier observations [21], we study robust least-squares regression (*RLSR*) to handle the challenge of learning a reliable set of regression coefficients given the presence of significant adversarial corruption in its response vector. Due to the ubiquitousness of data corruption and the popularity of regression methods, *RLSR* has become a critical component of several important real-world applications in various domains such as signal processing [11, 31, 42], economics [18, 27], bioinformatics [20], social media [10, 35], and image processing [23, 34].

Given a data matrix $X = [x_1, \dots x_n] \in \mathbb{R}^{p \times n}$ and its corresponding response vector $y \in \mathbb{R}^n$, a commonly adopted model from existing methods assumes the observed response is obtained from the generative model $\boldsymbol{y} = X^T \boldsymbol{\beta}^* + \boldsymbol{\varepsilon}$, where $\boldsymbol{\beta}^*$ is the true regression coefficients that we wish to recover and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is the noise vector under stochastic distributions. However, this method cannot explicitly model adversarial attacks on the data, which may generate arbitrarily corrupted data. Instead, we model the noise vector into two parts: adversarial data corruption and white noise with small bound. Then, the response vector is represented as $\boldsymbol{y} = X^T \boldsymbol{\beta}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where $\boldsymbol{u} \in \mathbb{R}^n$ is the corruption vector with arbitrarily corrupted values and $\boldsymbol{\varepsilon}$ contains the white noises. Notice that for an adaptive adversary, the corruption ratio γ cannot be larger than $1/2$ since the adversary can introduce a corruption vector as $\boldsymbol{u} = X^T(\boldsymbol{\beta}' - \boldsymbol{\beta}^*)$ to make it impossible for any algorithm to distinguish the ground truth $\boldsymbol{\beta}^*$ and adversarially chosen model $\boldsymbol{\beta}'$.

For those seeking to address the robust regression problem, the major challenges can be summarized as follows. (1) *Difficulty in estimating the corrupted data.* For data corruption estimation, a common assumption for corruption vector $\boldsymbol{u}$ is that the vector is sparsely supported such as $\|\boldsymbol{u}\|_0 \leq \gamma \cdot n$, where γ is the corruption ratio. A naive solution is to require the parameter inputted by users, where the parameter is expected to be larger than the exact corruption ratio γ^* to ensure all the corrupted data is eliminated [1]. Unfortunately, it is seldom practical for users to estimate the corruption ratio under the assumption that the response vector can be arbitrarily corrupted. Also, the corruption vector $\boldsymbol{u}$ can be handled in L_1-penalty based methods [25] by solving the problem: $\arg\min_{\boldsymbol{\beta}, \boldsymbol{u}} \|\boldsymbol{\beta}\|_1 + \lambda \|\boldsymbol{u}\|_1$, s.t., $\boldsymbol{y} = X^T \boldsymbol{\beta} + \boldsymbol{u}$. However, it still requires a parameter λ to control the sparsity of data corruption, which is also difficult for users to estimate in practice. (2) *Rigorous recovery guarantee of regression coefficients.* The existing theoretical analysis on robust regression methods always assumes severe restrictions on the data distribution. For example, data is required to be sampled from an isotropic Gaussian ensemble [34] or row-sampled from an incoherent orthogonal matrix [25]. The severe data restrictions of these methods make their recovery properties hard to be applied in real-world data. Moreover, some robust regression methods, such as a sub-sampling algorithm [22] require mild assumptions on the data, but the recovery boundaries of their methods are not rigorous in a massive dataset. (3) *Scaling to massive datasets.* The fast-growing amount of data makes the efficiency of robust regression algorithms more important than ever before. For instance, the system of the urban Internet of Things (IoT) [37] can produce thousands of data records every second in monitoring air quality, energy consumption, and traffic congestion. Therefore, it is necessary for the robust model to handle data faster than the throughput of these real-world systems.

To simultaneously address these technical challenges, this article presents a novel Robust regression model via Heuristic Corruption Thresholding (*RHCT*) and its adaptive estimation variation, named Robust regression via Adaptive Corruption Thresholding (*RACT*). The main contributions of our study are summarized as follows:

— *Proposing efficient algorithms to address the RLSR problem.* Two novel robust regression algorithms, *RHCT* and *RACT*, are proposed to recover the regression coefficients and uncorrupted set based on heuristic corruption thresholding and its adaptive variation, respectively. Unlike with a fixed corruption ratio, our methods can dynamically estimate the data corruption ratio based on the optimized regression coefficients in each iteration. The new design of corruption estimation makes our methods perform efficiently when the data size becomes large.
— *Designing effective approaches to estimate the corruption ratio.* A novel heuristic corruption thresholding method is proposed to estimate the corruption ratio by minimizing a novel heuristic function of residual errors. To improve the efficiency of corruption ratio estimation when the data size is extremely large, we also propose an adaptive variation method to estimate the corruption ratio based on adaptive searching steps without computing heuristic values for all the data samples. Our empirical results show the adaptive variation runs more than 500% faster than heuristic-based methods when the data size is over 1 million.
— *Providing a rigorous robustness guarantee for regression coefficient recovery.* We prove that our *RHCT* algorithm converges at a geometric rate and recovers β^* exactly under the assumption that the least-squares function satisfies both the Subset Strong Convexity (*SSC*) and Subset Strong Smoothness (*SSS*) properties. Specifically, we prove that our corruption thresholding methods ensures that the residual of the estimated uncorrupted set in each iteration has a tight upper error bound for the true uncorrupted set.
— *Conducting extensive experiments for performance evaluation.* Our proposed algorithm was evaluated with eight competing methods in both synthetic and real-world datasets. The results demonstrate that our approaches consistently outperform existing methods in both regression coefficients and uncorrupted set recovery, delivering a competitive running time.

The remainder of this article is organized as follows. Section 2 reviews the background and related work in robust regression models and outlier detection methods. Section 3 gives a formal problem formulation. The proposed *RHCT* algorithm and its adaptive variation, *RACT*, are presented in Section 4. Section 5 presents the proof for the recovery guarantees. In Section 6, the experimental results on both synthetic and real-world datasets are analyzed, and the article concludes with a summary of our work in Section 7.

2 RELATED WORK

The work related to this article is summarized in two categories: robust regression models and outlier detection.

2.1 Robust Regression Models

A large body of literature on the robust regression problem has been built up over the last few decades. Most of the studies focus on handling stochastic noise in small amounts [14, 19]; however, these methods cannot be applied to data that exhibits malicious corruption [6, 38, 40]. Some work discovers regression coefficients with adversarial data corruption [1, 6, 16, 39], but most of them [16, 29] lack the theoretical guarantee of regression coefficients recovery. To theoretically guarantee the recovery performance, Chen et al. [6] proposed a trimmed inner product based algorithm, but the recovery boundary of their method is not tight in a massive dataset. Also, McWilliams et al. [22] proposed a sub-sampling algorithm for a large-scale corrupted linear regression; however, the recovery result they provide is not close to an exact recovery [1]. To pursue the exact recovery results for the *RLSR* problem, some work focused on L_1 penalty based convex formulations [25, 33]. However, these methods imposed severe restrictions on the data distribution, such

as row-sampling from an incoherent orthogonal matrix [25]. Although robust regression methods, such as M-estimator [14] and least-trimmed squares [28], have been shown to be effective in many applications, their success relies on the proper choices of threshold parameters, such as corruption ratio [1] and regularization parameter [29], which are difficult to determine manually. For instance, Chen et al. [6] require the upper bound of the outliers number, which is also difficult to estimate when they assume the data can be adversarially corrupted. She and Owen [29] rely on a regularization parameter to control the size of the uncorrupted set based on soft-thresholding. Recently, Bhatia et al. [1] proposed a hard-thresholding algorithm for the *RLSR* problem. Although the method guarantees an exact recovery of regression coefficients $\boldsymbol{\beta}$ under a mild assumption for covariate matrix, their results are highly dependent on the corruption ratio parameter γ inputted by users. Specifically, the parameter is required to be larger than the exact corruption ratio γ^* to ensure its convergence. Unfortunately, it is seldom practical to estimate the corruption ratio under the assumption that the response vector is arbitrarily corrupted.

2.2 Anomaly and Outlier Detection

Outlier detection, also known as anomaly detection, has been well studied in a wide range of practical applications [2, 8, 24]. Literature on this work can be broadly classified into the following two types [9]: parametric methods [7] and non-parametric methods [32]. Parametric outlier detection methods explicitly assume the probabilistic or distribution model for the given data, while the non-parametric methods do not assume any knowledge of the data distribution. In parametric outlier detection, some work utilized heavy-tailed distributions [41] such as Student t-distribution and Poisson distribution, to model the outliers, while others detected outliers based on Gaussian distribution [9, 30] under the assumption that outliers have a small probability of occurrence in the population. However, the lack of prior knowledge regarding the underlying distribution of the dataset makes the distribution-based methods difficult to use in practice. Moreover, the data can be corrupted adversarially without following any distribution, which makes any assumption on data distribution infeasible.

In non-parametric methods, no prior knowledge on the data distribution is assumed. One popular non-parametric approach for outlier detection is based on kernel functions [17, 26]. These approaches utilize kernel functions to approximate the actual density distribution. The instances lying in the low probability area of the kernel density function are declared to be outliers. The kernel-function-based methods are computationally expensive when the data dimension increases. Another type of work, called distance-based outlier detection methods, detects outliers based on local neighborhood or k-nearest neighbors (kNN) in measuring the distance between each data point. However, both neighborhood and kNN searches in large datasets are not expensive, which typically requires $O(N^2)$ distance computation. Compared to distance-based methods, density-based outlier detection methods [3, 4, 15] generally have a stronger capability of modeling outliers by investigating not only the neighbors but the local densities. For instance, Breuning et al. [4] quantify the outlying degree of points using the local outlier factor to distinguish outliers from the rest of the data no matter the parameter of neighborhood distance. However, the approach requires computing the local outlier factor for all data points, which is much more computationally expensive than distance-based methods. Last, all the outlier detection methods require detecting outlier data points before utilizing the regression models, which hardly make a theoretical guarantee on the recovery of regression coefficients.

3 PROBLEM FORMULATION

In this study, we consider the problem of *RLSR* with adversarially corrupted data. Given a covariate matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, where each column $\boldsymbol{x}_i \in \mathbb{R}^{p\times 1}$ and $\boldsymbol{\beta}^*$ represents the ground truth

Table 1. Math Notations

Notations	Explanations
$p, n \in \mathbb{R}$	number of features and data samples
$X \in \mathbb{R}^{p\times n}$	data samples containing all the features
$\boldsymbol{\beta}, \boldsymbol{\beta}^* \in \mathbb{R}^{p\times 1}$	estimated and ground truth regression coefficients
$\boldsymbol{u} \in \mathbb{R}^{n\times 1}$	corruption vector with adversarial values
$\boldsymbol{\varepsilon} \in \mathbb{R}^{n\times 1}$	dense noise vector, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$
$\boldsymbol{y} \in \mathbb{R}^{n\times 1}$	response vector, where $\boldsymbol{y} = X^T\boldsymbol{\beta}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}$
$\boldsymbol{r} \in \mathbb{R}^{n\times 1}$	residual vector, where $\boldsymbol{r} = \lvert\boldsymbol{y} - X^T\boldsymbol{\beta}\rvert$
$S \subseteq [n]$	estimated uncorrupted set
$S_* \subseteq [n]$	ground truth uncorrupted set, where $S_* = \overline{supp(\boldsymbol{u})}$
$\Psi, \Psi_* \subseteq [\mu]$	estimated and ground truth feature sets

coefficients of the regression model, we assume the corresponding response vector $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$ is generated using the following model:

$$\boldsymbol{y} = \boldsymbol{y}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{y}^* = X^T\boldsymbol{\beta}^*$ and $\boldsymbol{u}$ is the unbounded corruption vector introduced by an adversary. $\boldsymbol{\varepsilon}$ represents the additive dense noise, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The goal of our study is to learn a new problem, which is to recover the regression coefficients $\boldsymbol{\beta}^*$ and simultaneously determine the uncorrupted point set $\hat{S}$. The problem is formally defined as follows:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \arg\min_{\boldsymbol{\beta}, S} \left\|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\right\|_2^2 \\ s.t.\ \ S &\subset [n],\ |S| \geq \mathcal{G}(\boldsymbol{\beta}). \end{aligned} \tag{2}$$

Given a subset $S \subset [n]$, $\boldsymbol{y}_S$ 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 $X_S \in \mathbb{R}^{p\times|S|}$. We use the notation $S_* = \overline{supp(\boldsymbol{u})}$ to denote the set of uncorrupted points, where $supp(\cdot)$ is the subset whose elements are not zero. Also, for any vector $\boldsymbol{v} \in \mathbb{R}^n$, the notation $\boldsymbol{v}_S$ represents the $|S|$-dimensional vector containing the components in S. The function $\mathcal{G}(\cdot)$ is to determine the size of set S according to the regression coefficients $\boldsymbol{\beta}$, which is explained in Section 4. The notations used in this paper are summarized in Table 1.

The optimization problem in Equation (2) is challenging because the joint optimization of regression coefficients $\boldsymbol{\beta}$ and the uncorrupted set S is a non-convex problem in general and existing methods cannot guarantee rigorous recovery and provide an efficient convergence rate. To prove the theoretical recovery of regression coefficients, we require that the least squares function satisfies the *SSC* and *SSS* properties [1], which are defined as follows:

Definition 3.1 (SSC and SSS Properties). The least-squares function $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\|_2^2$ satisfies $2\zeta_\alpha$-SSC property and $2\kappa_\alpha$-SSS property if the following holds:

$$\zeta_\alpha I \preceq \frac{1}{2}\nabla^2 f_S(\boldsymbol{\beta}) \preceq \kappa_\alpha I \quad \text{for } \forall S \in S_\alpha. \tag{3}$$

Equation (3) is equivalent to

$$\zeta_\alpha \leq \min_{S\in S_\alpha} \lambda_{min}\left(X_S X_S^T\right) \leq \max_{S\in S_\alpha} \lambda_{max}\left(X_S X_S^T\right) \leq \kappa_\alpha,$$

where λ_{min} and λ_{max} are defined as the smallest and largest eigenvalues of matrix X, respectively. The *SSC* and *SSS* properties will be utilized in Section 5 to prove the theoretical guarantee on the recovery of regression coefficients.

4 PROPOSED METHODOLOGY

We propose a robust regression algorithm via heuristic corruption thresholding, *RHCT*, in Section 4.1. To improve the efficiency of the corruption estimation, an adaptive corruption thresholding based algorithm, *RACT*, is proposed in Section 4.2.

4.1 Robust Regression via Heuristic Corruption Thresholding

In order to solve the problem in Equation (2) with the guarantee on the rigorous recovery of regression coefficients, we propose a novel heuristic corruption thresholding based robust regression algorithm, *RHCT*. As the regression coefficients $\boldsymbol{\beta}$ and uncrorrupted set S in Equation (2) are interdependent, the algorithm iteratively optimizes $\boldsymbol{\beta}$ and S until both of them are converged. Although the optimization of regression coefficients $\boldsymbol{\beta}$ has a closed-form solution when S is fixed, the optimization of S is not trivial because it mounts to a non-convex discrete optimization problem. To handle it, we propose a heuristic corruption thresholding method to determine the size of set S and then apply the estimated uncorrupted size into hard thresholding operator for the optimization of S elements.

Let residual vector $\boldsymbol{r} = \boldsymbol{y} - X^T\boldsymbol{\beta}$ and $r_{\delta(k)}$ be the kth elements of $\boldsymbol{r}$ in ascending order of magnitude. The heuristic corruption thresholding method determines the size of the optimal uncorrupted set $\hat{\tau}$ by optimizing the following problem:

$$\hat{\tau} = \arg\min_{\tau} \mathcal{L}(\tau) \quad s.t. \quad r_{\delta(\tau)} \leq \frac{2\tau r_{\delta(\tau_o)}}{\tau_o}, \tag{4}$$

where the function $\mathcal{L}$ is defined as

$$\mathcal{L}(\tau) := \frac{(r_{\delta(\tau)} + 1)/\tau}{(r_{\delta(n)} - r_{\delta(\tau)})/(n - \tau)}. \tag{5}$$

The variable τ_o in the constraint is defined as follows:

$$\tau_o = \arg\min_{1 \leq \tau \leq n} \left| r^2_{\delta(\tau)} - \frac{\|r_{S_{\tau'}}\|_2^2}{\tau'} \right| \quad s.t. \ \tau_o \in \mathbb{Z}^+, \tag{6}$$

where $\tau' = \tau - \lceil n/2 \rceil$ and $S_{\tau'}$ is the position set containing the smallest τ' elements in residual $\boldsymbol{r}$. The constraint is imposed to avoid the case when τ is close to n, where the residual becomes so arbitrary that the denominator can become very large, making $\mathcal{L}$ much smaller than the value of the estimated threshold $\hat{\tau}$. Applying the estimated uncorrupted set size $\hat{\tau}$ and residual vector $\boldsymbol{r}$, the hard thresholding operator determines the elements in the uncorrupted set S by selecting $\hat{\tau}$ samples with minimum values from residual vector $\boldsymbol{r}$ in ascending order. The formal definition of the hard thresholding operator is as follows:

Definition 4.1 (Hard Thresholding Operator). Defining $\delta^{-1}_{\boldsymbol{r}}(i)$ 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

$$\mathcal{H}_{\hat{\tau}}(\boldsymbol{r}) = \{i \in [n] : \delta^{-1}_{\boldsymbol{r}}(i) \leq \hat{\tau}\}. \tag{7}$$

We will first present the reasoning behind our choice of function in this section, and then show that our heuristic function can indeed ensure a rigorous recovery of regression coefficients $\boldsymbol{\beta}$ in Section 5. Basically, our method follows an intuition that when the coefficients $\boldsymbol{\beta}$ are close to $\boldsymbol{\beta}^*$, the residual $r_i = y_i - X_i\boldsymbol{\beta}$ of the uncorrupted sample i is smaller than that of corrupted sample in

high possibility. The intuition can be explained by the generative model in Equation (1), where the corrupted samples have the residual $\boldsymbol{r} \approx \boldsymbol{u} + \boldsymbol{\varepsilon}$, but the residual of uncorrupted samples only contains the white noise $\boldsymbol{\varepsilon}$. Moreover, as the corruption vector $\boldsymbol{u}$ is arbitrary and unbounded, when the residual vector $\boldsymbol{r}$ is sorted in ascending order, the slope of overall corrupted data is always much larger than the slope of the uncorrupted data. As Figure 1 shows, point p^* has the minimum $\mathcal{L}$ value in the feasible domain. Therefore, we can estimate the corresponding threshold τ_* of point p^* as the optimal threshold. To avoid a zero value for the numerator of $\mathcal{L}(\tau)$, we add 1 to all the values in the residual vector.

Algorithm 1 shows the detailed steps of the heuristic corruption thresholding method. The result of the $\mathcal{L}$ function for each uncorrupted size τ is computed in Lines 2–3 and stored in vector $\boldsymbol{l}$, which is initialized in Line 1. The method will keep searching the uncorrupted size from the index $\tau = \delta_{\boldsymbol{l}}^{-1}(1)$, which has the smallest value in the $\boldsymbol{l}$ vector to the largest index $\delta_{\boldsymbol{l}}^{-1}(\lceil n/2 \rceil)$ in the ascending order of $\boldsymbol{l}(i)$ until the current index τ satisfies the constraint in Equation (2). Then, the estimated uncorrupted set $\hat{S} = \mathcal{H}_\tau(\boldsymbol{r})$ is returned in Line 9. Otherwise, $\mathcal{H}_{\lceil n/2 \rceil}(\boldsymbol{r})$ is returned in Line 13 in case no τ satisfies the constraint in Equation (2).

The robust regression algorithm *RHCT*, based on heuristic hard thresholding, is proposed in Algorithm 2. It follows an intuitive strategy of updating $\boldsymbol{\beta}$ to provide a better fit for the current estimated set S in Line 3, and updating the residual vector $\boldsymbol{r}$ in Lines 4–5. It then estimates an active set S of uncorrupted points via heuristic hard thresholding in Line 6 based on the residual vector $\boldsymbol{r} = \boldsymbol{y} - X\boldsymbol{\beta}$ in the current iteration. The active set is initialized using all the data samples in Line 1. The algorithm continues until the change in the residual vector falls within a small range. Figure 2 shows the residual of the uncorrupted set in the 1st and 5th iterations. It intuitively explains the convergence progress of our algorithm: the optimization steps of $\boldsymbol{\beta}$ based on S_t make $\boldsymbol{r}_{S_t}$ smaller than its previous iteration, and it leads to smaller $\mathcal{L}$ values for items in S_t. Then, these items in S_t

ALGORITHM 1: Heuristic Corruption Thresholding

Input: Residual vector $\boldsymbol{r}$, sample number n.
Output: Uncorrupted Set $\hat{S}$

1 $\boldsymbol{l} \leftarrow \boldsymbol{0}$, $i \leftarrow 1$
2 **for** $j = 1..n$ **do**
3 $\quad l_j \leftarrow \frac{(r_{\delta(j)}+1)/j}{(r_{\delta(n)}-r_{\delta(j)})/(n-j)}$ // compute the heuristic value for each data sample.
4 **repeat**
5 $\quad \tau \leftarrow \delta_{\boldsymbol{l}}^{-1}(i)$
6 $\quad \tau' \leftarrow \tau - \lceil n/2 \rceil$
7 $\quad \tau_o \leftarrow \arg\min_{1 \le k \le n} \left| r_{\delta(k)}^2 - \frac{\|\boldsymbol{r}_{S_{\tau'}}\|_2^2}{\tau'} \right| \quad s.t.\ k \in \mathbb{Z}^+$ // compute the value of τ_o based on current τ.
8 $\quad$ **if** $r_{\delta(\tau)} \le \frac{2\tau r_{\delta(\tau_o)}}{\tau_o}$ **then**
9 $\quad\quad$ **return** $\mathcal{H}_\tau(\boldsymbol{r})$ // return the heuristic hard thresholding $\mathcal{H}(\cdot)$ when constraint is satisfied.
10 $\quad$ **end**
11 $\quad i \leftarrow i + 1$
12 **until** $i \le \lceil n/2 \rceil$
13 **return** $\mathcal{H}_{\lceil \frac{n}{2} \rceil}(\boldsymbol{r})$ // return estimated corruption set with $\tau = \lceil \frac{n}{2} \rceil$ if no proper value of τ is found.

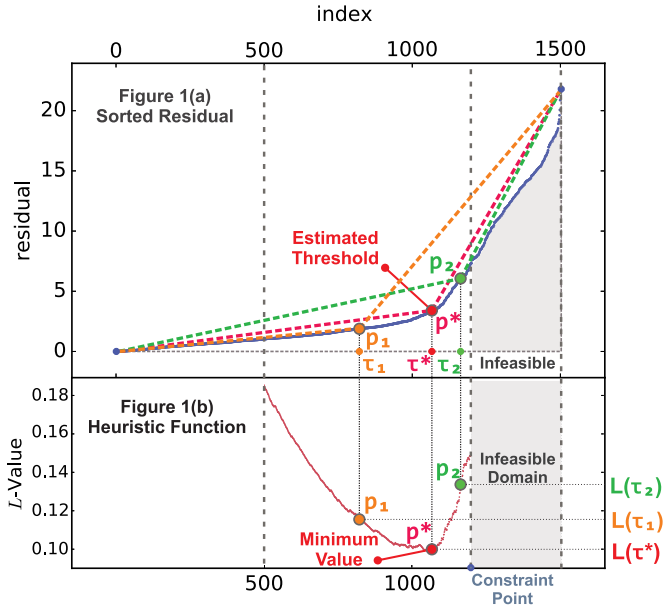


Fig. 1. The blue line in Figure 1(a) indicates the values of residual vector $\boldsymbol{r}$ in ascending order, and the red point in Figure 1(b) shows the corresponding value of the heuristic function. τ_* is the estimated threshold with the minimum $\mathcal{L}$; τ_1 and τ_2 are the candidate threshold values in τ_*'s left- and right-hand sides, respectively.

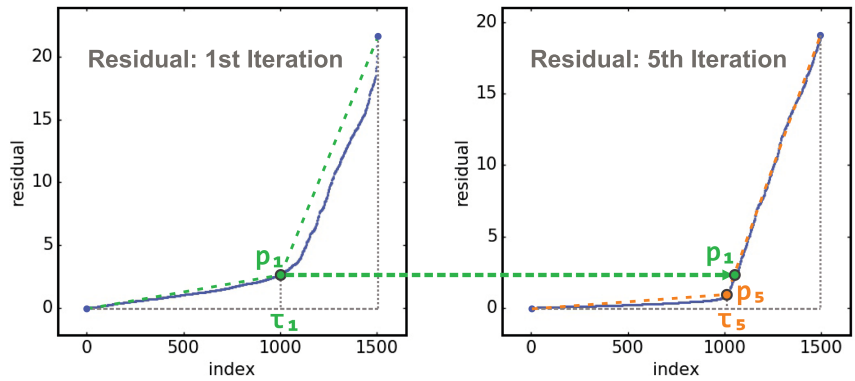


Fig. 2. Residual $\boldsymbol{r}$ in ascending order for the 1st (left) and 5th (right) iterations of *RHCT* algorithm.

have a much higher possibility of being kept in S_{t+1} than items in $[n] \setminus S_t$. This progress continues until the active set is fixed.

4.2 Adaptive Corruption Thresholding

The *RHCT* algorithm estimates the uncorrupted set S by computing the heuristic function $\mathcal{L}$ values in Equation (5) for each data sample and the $\mathcal{L}$ values of all the data samples are required to be sorted for further uncorrupted set estimation. Although the computation of the heuristic function for each data sample is efficient, when the data size is extremely large, both the heuristic function computation and sorting operation are very time consuming. To solve this problem, a

ALGORITHM 2: RHCT ALGORITHM

Input: Corrupted training data $\{\boldsymbol{x}_i, y_i\}$, i = 1 ... n, tolerance ϵ
Output: solution $\hat{\boldsymbol{\beta}}$

1 $S_0 = [\mathrm{n}]$, $t \leftarrow 0$
2 **repeat**
3 $\quad \boldsymbol{\beta}^{t+1} \leftarrow (X_{St} X_{St}^T)^{-1} X_{St} \boldsymbol{y}_{St}$ // Update the regression coefficients $\boldsymbol{\beta}$ with estimated corruption set S_t.
4 $\quad$ **for** $i = 1..n$ **do**
5 $\quad\quad r_i^{t+1} \leftarrow |y_{S_t i} - \boldsymbol{x}_{S_t i}^T \boldsymbol{\beta}|$ // Update the residual value for each data sample.
6 $\quad S_{t+1} \leftarrow HCT(\boldsymbol{r}^{t+1})$ // calculate the estimated corruption set S_{t+1} based on residual vector $\boldsymbol{r}^{t+1}$.
7 $\quad t \leftarrow t + 1$
8 **until** $\|\boldsymbol{r}_{S_{t+1}}^{t+1} - \boldsymbol{r}_{S_t}^t\|_2 < \epsilon n$
9 **return** $\boldsymbol{\beta}^{t+1}$

novel adaptive corruption thresholding method is proposed to estimate the uncorrupted set without computing the heuristic function values for each data sample. Instead, the new method starts searching for the uncorrupted size from n to $\lceil n/2 \rceil + 1$ in adaptive descent steps. Before introducing the details of the new proposed method, we will re-formalize the problem of uncorrupted size estimation as

$$\hat{\tau} := \underset{\lceil n/2 \rceil < \tau \le n}{\arg\max}\ \tau \quad s.t.\ r_{\delta(\tau)} \le \frac{2\tau r_{\delta(\tau_o)}}{\tau_o}, \tau \in \mathbb{Z}^+, \tag{8}$$

where $r_{\delta(k)}$ represents the kth elements of residual vector $\boldsymbol{r}$ in ascending order of magnitude. The variable τ_o in the constraint is defined as an intermediate variable whose $r_{\delta(\tau_o)}^2$ has the closest value to $\frac{\|\boldsymbol{r}_{\mathcal{H}_{\tau'}(\boldsymbol{r})}\|_2^2}{\tau'}$, where $\tau' = \tau - \lceil n/2 \rceil$ and $\mathcal{H}_{\tau'}(\boldsymbol{r})$ represents the position set containing the smallest τ' elements in the residual $\boldsymbol{r}$.

Compared to the problem defined in Equation (4), the new problem can be more efficiently solved without computing and sorting heuristic function values. One intuitive way to solve the problem is searching from n to $\lceil n/2 \rceil + 1$ one by one and returning the first $\hat{\tau}$ that satisfies the constraint in Equation (8). Although the intuitive method is easy to implement and can always achieve an optimal solution, it is not efficient to verify the constraint from n to the estimated uncorrupted size $\hat{\tau}$ one by one when the interval between n and $\hat{\tau}$ is very large. We define an intermediate variable Δ as $\Delta \equiv \frac{\tau_o r_{\delta(\tau)}}{2\tau r_{\delta(\tau_o)}}$; then, the constraint in Equation (8) can be rewritten as $\Delta \le 1$.

In the case that Δ is larger than 1, we get $\frac{r_{\delta(\tau)}}{r_{\delta(\tau_o)}} > \frac{2\tau}{\tau_o} > 2$, which shows the residual of $\delta(\tau)$ is at least two times larger than $\delta(\tau_o)$. Because $\boldsymbol{r}_{\delta(\tau)} = X_{\delta(\tau)}^T(\boldsymbol{\beta}^* - \boldsymbol{\beta}^t) + \boldsymbol{u}_{\delta(\tau)} + \boldsymbol{\varepsilon}_{\delta(\tau)}$ and the variance of dense noise vector $\boldsymbol{\varepsilon}$ is small, the value of $\boldsymbol{r}_{\delta(\tau)}$ is mainly dependent on the value of data corruption $\boldsymbol{u}_{\delta(\tau)}$. Since $\boldsymbol{u}_{\delta(\tau_o)} \approx 0$, a decrease of τ can make $u_{\delta(\tau)}$ smaller but keeps the value $\boldsymbol{u}_{\delta(\tau_o)}$ unchanged. Therefore, $\frac{r_{\delta(\tau)}}{r_{\delta(\tau_o)}}$ becomes smaller and the value of Δ is closer to 1. Similarly, in the case that $\Delta < 1$, we need to increase the value of τ to make Δ closer to 1. It is important to note that the estimation method in Equation (8) requires the coefficients $\boldsymbol{\beta}$ to be optimized simultaneously. Thus, we optimize the uncorrupted set S along with coefficients $\boldsymbol{\beta}$ until both of them converge.

The detailed steps of the adaptive corruption thresholding method are shown in Algorithm 3. The uncorrupted size τ_i of the ith iteration is initialized to n in Line 1. The result of auxiliary variables τ_o and Δ_i are computed in Line 3 and Line 4, respectively. After that, the corrupted size τ_{i+1} is updated to $\tau_i - \lfloor \eta \cdot (\Delta_i - 1) \cdot n \rfloor$ in Line 6 for the $i + 1$ iteration, where the step size is

ALGORITHM 3: Adaptive Corruption Thresholding

Input: Residual vector $\boldsymbol{r}$, sample number n, threshold ϵ, step length η
Output: Uncorrupted Set $\hat{S}$

1 $i \leftarrow 0$, $\tau_i \leftarrow n$
2 **repeat**
3 $\quad \tau_o \leftarrow \arg\min_{1 \le k \le n} \left| r^2_{\delta(k)} - \frac{\|\boldsymbol{r}_{S_{\tau_i'}}\|_2^2}{\tau_i'} \right| \quad s.t.\ k \in \mathbb{Z}^+$
4 $\quad \Delta_i \leftarrow \frac{\tau_o r_{\delta(\tau_i)}}{2\tau_i r_{\delta(\tau_o)}}$ `// compute the value of intermediate variable Δ in the ith iteration.`
5 $\quad$ **repeat**
6 $\quad\quad \tau_{i+1} \leftarrow \tau_i - \lfloor \eta \cdot (\Delta_i - 1) \cdot n \rfloor$ `// update τ with an adaptive searching step.`
7 $\quad\quad \tau'_{i+1} \leftarrow \tau_{i+1} - \lceil n/2 \rceil$
8 $\quad\quad$ **if** $\tau'_{i+1} < 0$ **then**
9 $\quad\quad\quad \eta \leftarrow \eta/2$ `// reduct the size of parameter η if current τ is too small.`
10 $\quad\quad$ **end**
11 $\quad$ **until** $\tau'_{i+1} > 0$
12 $\quad i \leftarrow i + 1$
13 **until** $|\Delta_i - 1| > \varepsilon$
14 **return** $\mathcal{H}_{\tau_{i-1}}(\boldsymbol{r})$ `// return the hard thresholding based on the τ in the` $i-1^{th}$ `iteration.`

adaptively controlled by the value of $\Delta_i - 1$. Notice that if the value of $\Delta_i - 1$ is larger than zero, a smaller uncorrupted size will be estimated; otherwise, the size is increased. To prevent the step size from being too large, which makes the estimated corrupted size smaller than $\lceil n/2 \rceil$, the step size parameter η will also be adaptively shrunk in Lines 8–10. Finally, the estimated uncorrupted set will be returned by hard thresholding based on the residual vector $\boldsymbol{r}$ in Line 14.

5 THEORETICAL RECOVERY ANALYSIS

In this section, the theoretical recovery analyses of our proposed algorithms will be presented. Both the *RHCT* and *RACT* algorithms share the same constraint and recovery steps, which leads them to have the same recovery property. Therefore, the recovery analysis of algorithm *RHCT* without dense noise is shown in this section, i.e., $\boldsymbol{y} = X^T\boldsymbol{\beta} + \boldsymbol{u}$. For the case with dense noise, $\boldsymbol{y} = X^T\boldsymbol{\beta} + \boldsymbol{u} + \boldsymbol{\varepsilon}$ will be presented in Appendix A.

The convergence proof relies on the optimality of two steps carried out by the algorithm, the $\boldsymbol{\beta}$ optimization step that selects the best coefficients based on the uncorrupted set, and the heuristic hard threshold step that automatically discovers the best active set based on the current regression coefficients.

Lemma 5.1. *For a given residual vector $\boldsymbol{r} \in \mathbb{R}^n$, let $\delta(k)$ be the kth position of the ascending order in vector $\boldsymbol{r}$, i.e., $r_{\delta(1)} \le r_{\delta(2)} \le \cdots \le r_{\delta(n)}$. For any $1 \le \tau_1 < \tau_2 \le n$, let $S_1 = \{\delta(i) | 1 \le i \le \tau_1\}$ and $S_2 = \{\delta(i) | 1 \le i \le \tau_2\}$. We then have $\|\boldsymbol{r}_{S_1}\|_2^2 \le \frac{\tau_1}{\tau_2}\|\boldsymbol{r}_{S_2}\|_2^2 \le \|\boldsymbol{r}_{S_2}\|_2^2$.*

Proof. Let $S_3 = \{\delta(i) : \tau_1 + 1 \le i \le \tau_2\}$. Clearly , we have $\|\boldsymbol{r}_{S_2}\|_2^2 = \|\boldsymbol{r}_{S_1}\|_2^2 + \|\boldsymbol{r}_{S_3}\|_2^2$. Moreover, since each element in S_3 is larger than any of the element in S_1, we have $\|\boldsymbol{r}_{S_1}\|_2^2 \le \|\boldsymbol{r}_{S_2}\|_2^2 + \frac{|S_3|}{|S_1|}\|\boldsymbol{r}_{S_1}\|_2^2 \le \frac{|S_1|}{|S_1|+|S_3|}\|\boldsymbol{r}_{S_2}\|_2^2 = \frac{\tau_1}{\tau_2}\|\boldsymbol{r}_{S_2}\|_2^2 \le \|\boldsymbol{r}_{S_2}\|_2^2$. □

Lemma 5.2. *Let S_t be the estimated uncorrupted set at the t*th *iteration. If $\tau_t \ge \tau_* = \gamma n$, then $|S_* \cap S_t| \ge \tau_t - \frac{n}{2}$.*

Proof. When S_t contains all the elements of $[n] \setminus S_*$, $|S_* \cap S_t|$ gets the smallest value $\tau_t - |[n] \setminus S_*|$. So we have

$$|S_* \cap S_t| \geq \tau_t - |[n] \setminus S_*| = \tau_t - (1-\gamma)n, \tag{9}$$

Because $\gamma > \frac{1}{2}$, we have

$$|S_* \cap S_t| \geq \tau_t - n + \frac{n}{n} = \tau_t - \frac{n}{2}. \tag{10}$$

□

Lemma 5.3. *Let τ_t be the estimated uncorrupted threshold at the tth iteration. If $\tau_t > \tau_* = \gamma n$, then $\|\boldsymbol{r}^t_{S_t}\|_2^2 \leq [1 + \frac{128(1-\gamma)}{2\gamma-1}]\|\boldsymbol{r}^t_{S_*}\|_2^2$.*

Proof. To simplify the notation, we will omit all the subscripts t that signify the tth iteration in the explanation below and assumes the residual vector $\boldsymbol{r}$ is sorted in ascending order of magnitude. According to the optimization step in Equation (4), we have the following properties:

$$\begin{aligned} r_\tau &\leq 2 \cdot \frac{\tau r_{\tau_o}}{\tau_o} \overset{(a)}{\leq} 8 \cdot r_{\tau_o} \\ r_\tau^2 &\overset{(b)}{\leq} \frac{64}{\tau'}\|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \\ |S_t \setminus S_*| r_\tau^2 &\overset{(c)}{\leq} (1-\gamma) \cdot n \cdot \frac{64}{\tau'}\|\boldsymbol{r}_{S_* \cap S_t}\|_2^2. \end{aligned} \tag{11}$$

Inequality *(a)* follows from $\tau_o \geq \tau/4$, and inequality *(b)* follows from the definition of τ_o in Equation (6) and the fact that $|S_* \cap S_t| \geq \tau'$ in Lemma 5.2. Inequality *(c)* follows from $|S_t \setminus S_*| \leq (1-\gamma) \cdot n$ and $\|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 \leq |S_t \setminus S_*| r_\tau^2$. Then, we have

$$\begin{aligned} \|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 &\leq \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'} + 1\right] \|\boldsymbol{r}_{S_* \setminus S_t}\|_2^2 \\ &\quad + \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'}\right] \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \\ \|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 + \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 &\overset{(d)}{\leq} \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'} + 1\right] \|\boldsymbol{r}_{S_*}\|_2^2 \\ \|\boldsymbol{r}_{S_t}\|_2^2 &\overset{(e)}{\leq} \left[1 + \frac{128(1-\gamma)}{2\gamma - 1}\right] \|\boldsymbol{r}_{S_*}\|_2^2. \end{aligned} \tag{12}$$

Inequality *(d)* follows from $\|\boldsymbol{r}_{S_*}\|_2^2 = \|\boldsymbol{r}_{S_* \setminus S_t}\|_2^2 + \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2$. Inequality *(e)* follows from $\tau' = \tau_t - \frac{n}{2}$. □

Theorem 5.4. *Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^p$ be the given data matrix and $\boldsymbol{y} = X^T\boldsymbol{\beta}^* + \boldsymbol{u}$ be the corrupted output with $\|\boldsymbol{u}\|_0 = \gamma n$. Let Σ_0 be an invertible matrix such that $\tilde{X} = \Sigma_0^{-1/2}X$; $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - \tilde{X}_S\boldsymbol{\beta}\|_2^2$ satisfies the SSC and SSS properties at level α, γ with $2\zeta_{\alpha,\gamma}$ and $2\kappa_{\alpha,\gamma}$. If the data satisfies $\frac{\kappa_\gamma}{\zeta_{1-\alpha}} < \frac{1}{\sqrt{\lambda}}(\sqrt{2}-1)$, then after $t = O(\log_{\frac{1}{\eta}} \frac{\mu\|\boldsymbol{u}\|_2}{\epsilon})$ iterations, Algorithm 2 yields an ϵ-accurate solution $\boldsymbol{\beta}^t$.*

Proof. Let $G_t = (X_{S_t}X_{S_t}^T)^{-1}X_{S_t}$, the tth iteration of Algorithm 2 satisfies

$$\boldsymbol{\beta}^{t+1} = G_t\boldsymbol{y}_{S_t} = G_t\left(X_{S_t}^T\boldsymbol{\beta}^* + \boldsymbol{u}_{S_t}\right) = \boldsymbol{\beta}^* + G_t\boldsymbol{u}_{S_t}.$$

Thus, the residual in the $(t+1)$th iteration for any set $S \subset [n]$, yields

$$\boldsymbol{r}_S^{t+1} = \boldsymbol{y}_S - X_S^T\boldsymbol{\beta}^{t+1} = \boldsymbol{u}_S - X_S^TG_t\boldsymbol{u}_{S_t}.$$

For each iteration, we have two conditions when choosing different values of τ^{t+1}. For *condition 1*, $\tau^{t+1} \leq \tau_*$, and we have $\|\boldsymbol{r}_{S_{t+1}}^{t+1}\|_2^2 \leq \|\boldsymbol{r}_{S_*}^{t+1}\|_2^2$ (see Lemma 5.1).

$$\begin{aligned}\|\boldsymbol{u}_{S_{t+1}}\|_2^2 &= \left\|\boldsymbol{u}_{S_{t+1}} - X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 - \left\|X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 \\ &\quad + 2\boldsymbol{u}_{S_{t+1}}^T X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t} \\ &\overset{(a)}{\leq} \left\|X_{S_*}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 - \left\|X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 + 2\boldsymbol{u}_{S_{t+1}}^T X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t} \\ &\overset{(b)}{\leq} \frac{\kappa_{\alpha_1}^2}{\zeta_{1-\gamma}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\frac{\kappa_{\alpha_1}}{\zeta_{1-\gamma}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2,\end{aligned} \tag{13}$$

where $\alpha_1 = \max_t\{1 - \frac{\tau_t}{n}\}$. Inequality *(a)* follows from $\|\boldsymbol{r}_{S_{t+1}}^{t+1}\|_2^2 \leq \|\boldsymbol{r}_{S_*}^{t+1}\|_2^2$, and inequality *(b)* follows from the setting $\tilde{X} = \Sigma_0^{-1/2}X$, the *SSC*/*SSS* properties, $|S_t| \leq (1-\gamma)\cdot n$, and $|S_* \setminus S_{t+1}| \leq \alpha_1 \cdot n$. Solving the quadratic equation for the corruption vector gives us

$$\|\boldsymbol{u}_{S_{t+1}}\|_2 \leq (1+\sqrt{2})\frac{\kappa_{\alpha_1}}{\zeta_{1-\gamma}}\|\boldsymbol{u}_{S_t}\|_2. \tag{14}$$

For *condition 2*, $\tau^{t+1} > \tau_*$. According to Lemma 5.3, $\|\boldsymbol{r}_{S_t}^t\|_2^2 \leq \lambda\|\boldsymbol{r}_{S_*}^t\|_2^2$ where $\lambda = 1 + \frac{128(1-\gamma)}{2\gamma-1}$, so we have

$$\begin{aligned}\|\boldsymbol{u}_{S_{t+1}}\|_2^2 &= \left\|\boldsymbol{u}_{S_{t+1}} - X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 - \left\|X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 \\ &\quad + 2\boldsymbol{u}_{S_t}^T X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t} \\ &\overset{(c)}{\leq} \lambda\left\|X_{S_*}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 - \left\|X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t}\right\|_2^2 + 2\boldsymbol{u}_{S_t}^T X_{S_{t+1}}^T G_t \boldsymbol{u}_{S_t} \\ &\overset{(d)}{\leq} \lambda\frac{\kappa_\gamma^2}{\zeta_{1-\alpha_2}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2 \\ &\overset{(e)}{\leq} \lambda\frac{\kappa_\gamma^2}{\zeta_{1-\alpha_2}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\sqrt{\lambda}\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2,\end{aligned} \tag{15}$$

where $\alpha_2 = \max_t\{1 - \frac{\tau_t}{n}\}$. Inequality *(c)* follows from Lemma 5.3, and inequality *(d)* follows from the definition of the *SSC*/*SSS* properties, $|S_t| \leq (1-\alpha_2)\cdot n$, and $|S_* \setminus S_{t+1}| \leq \gamma \cdot n$. Inequality *(e)* follows from the fact that $\sqrt{\lambda} \geq 1$. Solving the quadratic equation in Equation (15) gives us

$$\|\boldsymbol{u}_{S_{t+1}}\|_2 \leq (1+\sqrt{2})\sqrt{\lambda}\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}\|_2. \tag{16}$$

Combine these two conditions and let t_1 be the iterations for the case of condition 1. We get

$$\begin{aligned}\|\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^*\|_2 &= \|G_t\boldsymbol{u}_{S_t}\|_2 \leq \mu\|\boldsymbol{u}_{S_t}\|_2 \\ &\leq \mu\cdot\eta_1^{t_1}\cdot\eta^{t+1-t_1}\|\boldsymbol{u}\|_2 \leq \mu\cdot\eta^{t+1}\|\boldsymbol{u}\|_2,\end{aligned}$$

where $\mu = \max\{\frac{\sqrt{\kappa_{\alpha_1}}}{\zeta_{1-\gamma}}, \frac{\sqrt{\kappa_\gamma}}{\zeta_{1-\alpha_2}}\}$, $\eta_1 = \frac{(1+\sqrt{2})\kappa_{\alpha_1}}{\zeta_{1-\gamma}}$, and $\eta = \frac{(1+\sqrt{2})\sqrt{\lambda}\kappa_\gamma}{\zeta_{1-\alpha_2}}$. When $\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}} < \frac{\sqrt{2}-1}{\sqrt{\lambda}}$, we have $\eta < 1$, and after $t = O(\log_{\frac{1}{\eta}}\frac{\mu\|\boldsymbol{u}\|_2}{\epsilon})$, $\|\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^*\|_2 \leq \epsilon$. □

6 EXPERIMENTAL RESULTS

In this section, we report the extensive experimental evaluation carried out to verify the robustness and efficiency of the proposed method. All the experiments were conducted on a 64-bit machine

with an Intel(R) core(TM) quad-core processor (i7CPU@3.6GHz) and 32.0GB memory. Details of both the source code and sample data used in the experiments can be downloaded here.[1]

6.1 Experiment Setup

6.1.1 Datasets and Labels. To demonstrate the performance of our proposed method, we carried out comprehensive experiments in both synthetic and real datasets. For the synthetic dataset, following the setting in in [1], the simulation samples were randomly generated according to the model in Equation (1) for the *RLSR* problem, sampling the regression coefficients $\boldsymbol{\beta}^* \in \mathbb{R}^p$ as a random unit norm vector. The covariance data X was drawn independently and identically distributed from $\boldsymbol{x}_i \sim \mathcal{N}(0, I_p)$, and the uncorrupted response variables were generated as $y_i^* = \boldsymbol{x}_i^T \boldsymbol{\beta}^*$. The set of corrupted points S was selected as a uniformly random $(n-\tau_*)$-sized subset of $[n]$, where τ_* is the size of the uncorrupted set. The corrupted response vector was generated as $\boldsymbol{y} = \boldsymbol{y}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where the corruption vector $\boldsymbol{u}$ was sampled from the uniform distribution $[-5\|\boldsymbol{y}^*\|_\infty, 5\|\boldsymbol{y}^*\|_\infty]$ and the additive dense noise was $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.

For the real-world datasets, we use house rental transaction data from *New York City* and *Los Angeles* on the Airbnb[2] website from January 2015 to October 2016. The datasets can be downloaded here.[3] 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. In each dataset, there were 21 features after data preprocessing, including the number of beds and bathrooms, location, and average price in the area.

6.1.2 Evaluation Metrics. For the synthetic data, performance of the regression coefficients recovery is measured by the standard L_2 error as follows:

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

where $\hat{\boldsymbol{\beta}}$ represents the recovered coefficients for each method and $\boldsymbol{\beta}^*$ is the true regression coefficients. To validate the performance for corrupted set discovery, the F1-score is measured by comparing the discovered corrupted sets with the actual ones. For the real-world dataset, mean absolute error (MAE) is used 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:

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

To compare the scalability of each method, the CPU running time for each of the competing methods was also measured in different settings of data size, corruption ratio, and feature number.

6.1.3 Comparison Methods. The following methods are included in the performance comparison presented here: *Ordinary least squares* (*OLS*). The *OLS* method trains the model based on the whole dataset without considering the corrupted samples in the dataset. We also compared our method to the *Huber Loss* [13] and regularized L_1 algorithm for robust regression [25, 33]. For extensive L_1 minimization solvers, Yang et al. [36] showed that the *Homotopy* and *DALM* solvers outperform other proposed methods both in terms of recovery properties and running time. Both

[1] https://github.com/xuczhang/RHCT_ACT.
[2] https://www.airbnb.com/.
[3] http://insideairbnb.com/get-the-data.html.

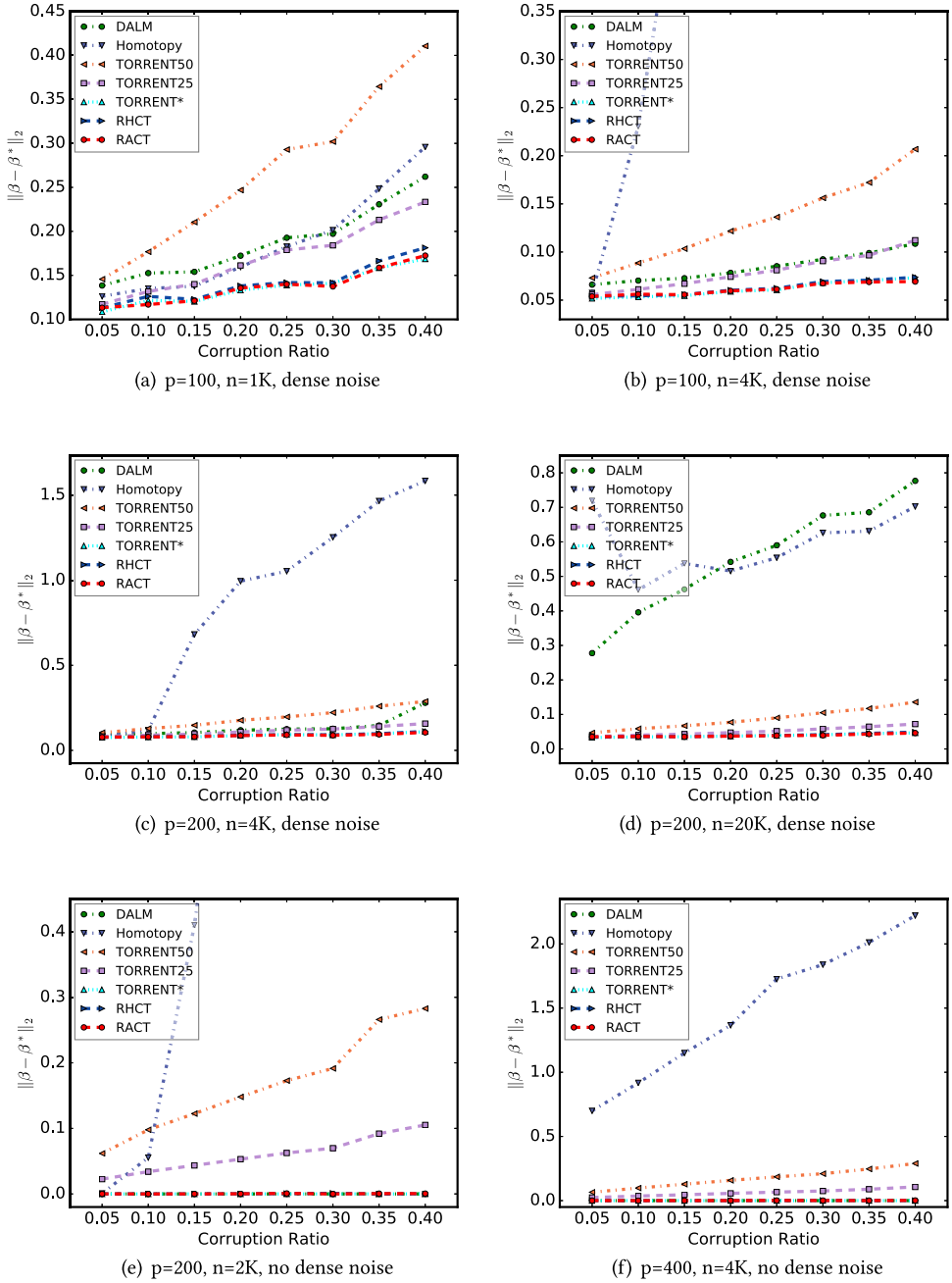
of the L_1 solver methods are parameter free. And, we compared our method to RELM-IRLS [5] with Bisquare loss function, which is a unified model for robust regularized extreme learning machine (ELM) [12] using iteratively reweighted least squares (IRLS) [5]. Another recently proposed hard thresholding method [1], *Torrent (abbr. Torr)*, developed for robust regression, was also compared to our method. As the method requires a parameter for the corruption ratio, which is difficult to estimate in practice, we chose four versions with different parameter settings: *TORR**, *TORR25*, *TORR50*, and *TORR80*. *TORR** uses the true corruption ratio as its parameter, and the others apply parameters that are uniformly distributed across the range of ±25%, ±50%, and ±80% off the true value, respectively. For the *RACT* method, we chose the step length $\eta = 0.01$ in all the experiments. All the results are averaged over 10 runs.

6.2 Recovery of Regression Coefficients

As RELM-IRLS does not explicitly estimate regression coefficient, we selected seven competing methods with which to evaluate the recovery performance of regression coefficients β: OLS, Huber, *DALM*, *Homotopy*, *TORR**, *TORR25*, and *TORR50*. As the recovery error for the *OLS and Huber* method is almost 10 times larger than those of the other methods, its result is not shown in Figure 3 in order to present the other results properly. Figures 3(a) and 3(b) show the recovery performance for different data sizes when the feature number is fixed. Looking at the results, we can conclude the following: (1) Both the *RHCT* and *RACT* methods outperform all the competing methods except for *TORR**, whose parameter is rarely given in practice. (2) The results of the *TORRENT*-based methods are significantly affected by their corruption ratio parameters; *TORR50* performs almost twice as badly as *TORR** and yields worse results than one of the L_1-Solver methods, *DALM*. However, *RHCT* and *RACT* perform consistently throughout, with no impact of the parameter. (3) The L_1-Solver methods generally exhibit worse performance than the hard-thresholding-based algorithms. Specifically, compared to *DALM*, *Homotopy* is more sensitive to the number of corrupted instances in the data. Figure 3(c) shows the similar performance when the feature number increases. Specifically, the overall recovery error of hard-thresholding-based methods increases less than 10% when the feature number increases 100% while L_1-Solver methods increase more than 50%. Figure 3(d) shows the performance of hard-thresholding-based methods are almost 200% better than L_1-Solver methods when data samples are much larger than the feature number, even for the *TORRENT* methods with mistakenly estimated corruption ratios. Figures 3(e) and 3(f) show that *RHCT* and *RACT* perform equally as well as *TORR** without dense noise, with all achieving almost exact recovery of regression coefficients $\boldsymbol{\beta}$.

6.3 Recovery of Uncorrupted Sets

As most competing methods do not explicitly estimate uncorrupted sets, we compared our proposed methods with the *TORR* algorithm using a number of different parameter settings ranging from the true corrupted ratio up to a deviation of 80%. As the results show in Table 2, we found the following: (1) The F1 score of *RHCT* is 1.1% less than that of *TORR** on average, although it is important to note that the latter uses the true corruption ratio, which cannot be estimated exactly in practice. This indicates that the *RHCT* method has a very competitive result even though it assumes that the corruption ratio is unknown. (2) The *RACT* method significantly outperforms the other methods, doing even better than *TORR** when the data contains dense noise. This is because dense noise will change the actual corruption ratio when some data samples containing dense noise accidentally have larger corruption residuals, which makes the *TORR** method, using a fixed corruption ratio, perform worse than the *RACT* method, which uses a dynamically estimated corruption ratio. (3) The results of the *TORRENT*-based methods are highly dependent on the corruption ratio parameter: The results for a 25% corruption estimation error are much better



(a) p=100, n=1K, dense noise

(b) p=100, n=4K, dense noise

(c) p=200, n=4K, dense noise

(d) p=200, n=20K, dense noise

(e) p=200, n=2K, no dense noise

(f) p=400, n=4K, no dense noise

Fig. 3. Performance on regression coefficients recovery.

than those for an 80% error. However, *RHCT* and *RACT* are parameter-free methods that are capable of consistently obtaining good results. (4) When increasing the data size and corruption ratio, the F1 scores slightly increase for all the methods. In contrast, the F1 score decreases when the feature number increases. (5) In a no-dense-noise setting, the *RHCT* and *RACT* methods perform a near optimal recovery result, while *TORR** exactly recovers the result only because it is using the

Table 2. F1 Scores for Performance on Uncorrupted Set Recovery

	p = 100, n = 1K				p = 100, n = 2K				p = 100, n = 4K			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
TORR80	0.949	0.881	0.779	0.612	0.950	0.883	0.783	0.622	0.951	0.883	0.785	0.626
TORR50	0.967	0.925	0.865	0.781	0.968	0.926	0.868	0.785	0.968	0.926	0.871	0.787
TORR25	0.981	0.958	0.927	0.887	0.982	0.960	0.929	0.891	0.982	0.960	0.931	0.892
RHCT	0.989	0.979	0.973	0.956	0.991	0.987	0.977	0.964	0.992	0.987	0.978	0.971
RACT	**0.993**	**0.989**	**0.984**	**0.972**	**0.993**	**0.989**	**0.984**	**0.978**	**0.993**	**0.989**	**0.985**	**0.981**
TORR*	**0.993**	0.987	0.979	0.971	**0.995**	**0.990**	0.980	0.972	**0.995**	**0.989**	0.982	0.975
	p = 200, n = 10K				p = 400, n = 10K				p = 800, n = 10K			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
TORR80	0.950	0.883	0.785	0.627	0.950	0.883	0.785	0.624	0.950	0.883	0.783	0.621
TORR50	0.968	0.927	0.871	0.788	0.968	0.926	0.870	0.786	0.968	0.926	0.869	0.785
TORR25	0.982	0.960	0.933	0.894	0.982	0.960	0.932	0.892	0.982	0.960	0.931	0.892
RHCT	0.991	0.988	0.981	0.973	0.991	0.987	0.979	0.970	0.991	0.985	0.978	0.966
RACT	**0.992**	**0.991**	**0.987**	**0.981**	**0.993**	**0.989**	**0.986**	**0.981**	**0.993**	**0.989**	**0.985**	**0.980**
TORR*	**0.995**	0.990	0.985	0.976	**0.995**	**0.989**	0.984	0.975	**0.995**	**0.989**	0.983	0.975
	p = 200, n = 100K				p = 400, n = 10K (nd)				p = 400, n = 100K (nd)			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
TORR80	0.950	0.883	0.786	0.627	0.953	0.889	0.793	0.636	0.953	0.889	0.793	0.636
TORR50	0.968	0.927	0.871	0.788	0.971	0.933	0.880	0.800	0.971	0.933	0.880	0.800
TORR25	0.982	0.960	0.933	0.893	0.986	0.968	0.943	0.909	0.986	0.968	0.943	0.909
RHCT	**0.992**	0.989	0.982	0.974	**0.994**	**0.994**	0.993	**0.993**	**0.993**	**0.994**	**0.994**	**0.993**
RACT	0.990	**0.990**	**0.987**	**0.982**	0.992	0.993	**0.994**	**0.993**	0.943	**0.994**	0.991	0.990
TORR*	**0.995**	**0.990**	0.984	0.976	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

true corruption ratio. (6) Although the F1 score of the *RACT* method is 1.6% better than the *RHCT* method on average when the data contains dense noise, *RHCT* outperforms *RHCT* has the similar performance as *RACT* in the no-dense-noise setting.

6.4 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. We selected eight competing methods with which to evaluate rental price prediction performance: *OLS*, *Huber*, *DALM*, *Homotopy*, *RELM-IRLS*, *TORR**, *TORR25*, and *TORR50*. Since the *DALM* and *Homotopy* methods require allocation of identity matrices with the dimension of whole data samples, it leads to an out-of-memory issue when there are more than 10,000 data samples. To solve the issue, we randomly divide the whole dataset into batches with 10,000 samples and average the result for each batch. Table 3 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 the following: (1) Our proposed methods, *RHCT* and *RACT*, outperform the other methods except *TORR** more than 50% in different corruption ratio settings for both datasets. Moreover, *RHCT* has slightly better results when the corruption ratio is less than 30%, but *RACT* works better when the corruption ratio is large. (2) Although *RELM-IRLS* uses Bisquare to enhance the robustness, its performance

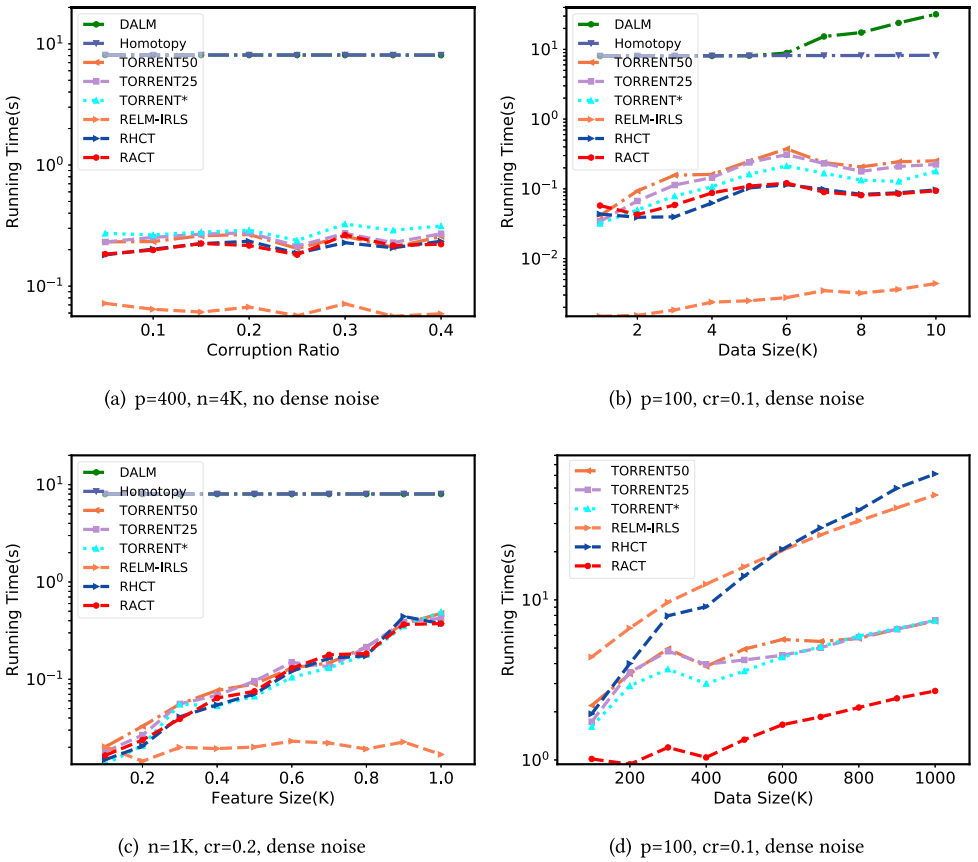
Table 3. Mean Absolute Error of Rental Price Prediction

	New York City (Corruption Ratio)					
	5%	10%	20%	30%	40%	Avg.
OLS	18.126±3.62	22.285±5.485	30.251±7.943	42.960±10.074	50.230±12.508	32.770±7.926
Huber	10.1±0.689	12.29±1.609	14.931±2.928	21.67±5.322	25.627±7.268	16.9236±3.5632
DALM	112.965±51.809	155.342±40.9	229.954±111.281	293.061±145.39	253.036±130.827	208.872±96.041
Homotopy	86.566±51.948	147.668±42.132	221.66±120.232	290.429±148.648	251.684±137.501	199.601±100.092
TORR25	8.15±0.078	8.342±0.226	9.203±0.307	9.628±1.017	11.022±1.402	9.269±0.606
TORR50	8.63±0.331	9.615±0.925	11.185±1.634	15.095±3.166	18.862±4.519	12.6774±2.115
RELM-IRLS	34.757±1.069	36.178±2.854	35.683±1.835	37.133±4.51	34.524±1.485	35.655±2.3506
RHCT	**7.973±0.002**	**7.974±0.004**	**7.979±0.008**	7.984±0.008	7.986±0.016	**7.979±0.0076**
RACT	8.0321±0.022	8.015±0.038	7.99±0.014	**7.979±0.008**	**7.973±0.004**	7.998±0.017
TORR*	**7.971±0.000**	**7.971±0.000**	**7.971±0.000**	**7.971±0.000**	**7.971±0.000**	**7.971±0.000**
	Los Angeles (Corruption Ratio)					
	5%	10%	20%	30%	40%	Avg.
OLS	33.459±13.176	47.012±12.971	64.571±23.784	84.631±27.027	190.651±74.737	84.0648±30.339
Huber	19.274±2.816	22.885±3.451	31.779±9.521	40.696±9.498	58.016±21.436	34.53±9.3444
DALM	155.424±55.351	135.488±67.74	208.605±87.44	306.279±171.85	567.848±305.205	274.729±137.512
Homotopy	130.769±64.978	112.375±68.634	186.989±110.681	301.435±189.407	550.084±335.766	256.330±153.893
TORR25	16.171±0.175	16.493±0.161	17.658±1.248	19.75±1.865	36.487±7.633	21.312±2.216
TORR50	16.907±0.404	19.523±1.325	24.755±3.314	31.954±6.268	69.398±18.887	32.507±6.040
RELM-IRLS	52.908±2.02	53.818±2.465	53.745±1.719	59.742±6.245	72.742±21.153	58.591±6.720
RHCT	**15.935±0.004**	**15.931±0.007**	**15.968±0.027**	16.171±0.239	16.36±0.507	16.073±0.157
RACT	16.209±0.418	16.123±0.343	15.981±0.087	**15.947±0.014**	**15.949±0.035**	**16.042±0.179**
TORR*	**15.904±0.001**	**15.906±0.000**	**15.906±0.001**	**15.91±0.005**	**15.911±0.008**	**15.907±0.003**

is still worse than other competing methods except *DALM* and *Homotopy* because the corrupted data will have a high impact on the *ELM* based regression. (3) The *TORR** method outperforms all the other methods; however, the method requires a corruption ratio parameter that is hard to estimate in practice. (4) Similar to the result in synthetic data, *TORRENT*-based methods' performance is significantly affected by the corruption parameters. Specifically, the results of *TORR25* and *TORR50* are at least 50% and 80% worse than *TORR**, respectively. (5) Both the *DALM* and *Homotopy* methods perform even worse than the *OLS* and *Huber* method because their results are combined with results in small batches due to their scalability issues. As *Huber* is a loss function in robust regression, which makes it less sensitive to corrupted data, it has a better performance than *OLS* method.

6.5 Efficiency

To evaluate the efficiency of our proposed method, we compared the performance of all the competing methods for three different data settings as follows: different corruption ratios, data sizes, and feature numbers in Figures 4(a)–4(c), respectively. Since the *DALM* and *Homotopy* methods cannot be scaled to a large data size, we compared our methods with *TORR*-based methods in large datasets, ranging from 100,000 to 1 million samples, in Figure 4(d). In general, as Figure 4 shows, we can conclude the following: (1) The hard-thresholding-based methods significantly outperformed the L_1-Solver-based methods. (2) The running time of the *RHCT* and *RACT* methods increases slowly when either the feature number or data size increases, just as in the *TORRENT*-based methods. (3) Figure 4(a) shows that the corruption ratio has little impact on the



(a) p=400, n=4K, no dense noise

(b) p=100, cr=0.1, dense noise

(c) n=1K, cr=0.2, dense noise

(d) p=100, cr=0.1, dense noise

Fig. 4. Running time for different corruption ratios and data sizes.

efficiency of all the methods because the difference of running time in various corruption ratios is less than 5%. (4) Even though *RHCT* performs the additional step of estimating the uncorrupted set in each optimization iteration, the efficiency of *RHCT* still outperforms *TORR* in small datasets, which indicates that the heuristic corruption thresholding step in *RHCT* performs efficiently and the *RHCT* algorithm can converge quickly when the data size is small. However, when the data size increases from 100,000 to 1 million in Figure 4(d), the running time of *RHCT* increases more than 50 times, which is much larger than *TORRENT*-based methods. This fact shows the efficiency of RHCT is highly impacted by the number of data samples, because the heuristic corruption thresholding step requires computation of the heuristic values for each data sample and sorts them altogether. (6) The running time of *RELM-IRLS* is less than the other competing methods in a small dataset because of the fast training speed of ELM regression. However, when the data size increases, its efficiency is dramatically decreased. Instead, *RACT* outperforms all the competing methods when the data size is larger than 100,000 in Figure 4(d) because its corrupted set is adaptively estimated without computing heuristic values as in *RHCT*, which makes *RACT* less impacted by the data size.

7 CONCLUSION

In this article, a novel robust regression algorithm, *RHCT*, is proposed to recover the regression coefficients and the uncorrupted set in the presence of adversarial corruption in the response

vector. To determine the corrupted set, we designed a heuristic corruption thresholding method to estimate the optimal uncorrupted set that is alternately updated with the optimized regression coefficients. Moreover, an adaptive corruption thresholding based algorithm, *RACT*, is designed to improve running-time efficiency when the amount of data becomes extremely large. We demonstrate that our algorithms can recover regression coefficients rigorously in the condition of the SSC and smoothness properties, with a geometric convergence rate. Extensive experiments on a massive amount of simulation data demonstrated that the proposed algorithms outperform other comparable methods in both effectiveness and efficiency.

APPENDIX

A ADDITIONAL THEORETICAL ANALYSIS

In this section, the regression coefficient recovery analysis for the case with dense noise is presented. Specifically, the corrupted response vector $\boldsymbol{y}$ is represented as $\boldsymbol{y} = X^T\boldsymbol{\beta} + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where $\boldsymbol{u}$ and $\boldsymbol{\varepsilon}$ stand for the vector of adversarial data corruption and dense noise, respectively.

Theorem A.1. *Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p\times n}$ be the given data matrix and the corrupted response vector $\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{\beta}_* + \boldsymbol{u} + \boldsymbol{\varepsilon}$ with $\|\boldsymbol{u}\|_0 = \gamma n$. Let Σ_0 be an invertible matrix such that $\tilde{X} = \Sigma_0^{-1/2}X$; $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - \tilde{X}_S\boldsymbol{\beta}\|_2^2$ 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 $t = O(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}\|_2}{\sqrt{n}\epsilon})$ iterations, Algorithm 2 yields an ϵ-accurate solution $\boldsymbol{\beta}_t$ with $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_t\|_2 \le \epsilon + \frac{C\|\boldsymbol{\varepsilon}\|_2}{\sqrt{n}}$ for some $C > 0$.*

Proof. We use S_t to represent the uncorrupted set after consolidation in the tth iteration, and we observe that the optimality of the regression coefficient $\boldsymbol{\beta}$ on the estimated corrupted set S_t ensures the following:

$$\begin{aligned}\left\|\boldsymbol{y}_{S_t} - X_{S_t}^T\boldsymbol{\beta}_{t+1}\right\|_2 =& \left\|X_{S_t}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}) + \boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\right\|_2 \\ \le& \left\|\boldsymbol{y}_{S_t} - X_{S_t}^T\boldsymbol{\beta}_*\right\|_2 = \|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2.\end{aligned}$$

Using the triangle inequality of the L_2 norm, we have

$$\begin{aligned}\left\|X_{S_t}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 - \|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \le& \|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \\ \left\|X_{S_t}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 \le& 2\|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \\ \sqrt{\zeta_\alpha}\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}\|_2 \overset{(a)}{\le}& 2\|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \\ \|\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}\|_2 \overset{(b)}{\le}& \frac{2}{\sqrt{\zeta_\alpha}}(\|\boldsymbol{\varepsilon}_{S_t}\|_2 + \|\boldsymbol{u}_{S_t}\|_2).\end{aligned}$$

Inequality *(a)* follows from $|S_t| \le \alpha n$, where $\alpha = \max_t\{\frac{\tau_t}{n}\}$. Inequality *(b)* follows from the triangle inequality $\|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \le \|\boldsymbol{\varepsilon}_{S_t}\|_2 + \|\boldsymbol{u}_{S_t}\|_2$. According to the hard thresholding step in Equation (5), we have the following:

$$\begin{aligned}\left\|X_{S_{t+1}}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}) + \boldsymbol{\varepsilon}_{S_{t+1}} + \boldsymbol{u}_{S_{t+1}}\right\|_2 =& \left\|\boldsymbol{y}_{S_{t+1}} - X_{S_{t+1}}^T\boldsymbol{\beta}_{t+1}\right\|_2 = \|\boldsymbol{r}_{S_{t+1}}\|_2 \\ \|\boldsymbol{u}_{S_{t+1}}\|_2 - \left\|X_{S_{t+1}}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 - \|\boldsymbol{\varepsilon}_{S_{t+1}}\|_2 \overset{(c)}{\le}& \lambda\|\boldsymbol{r}_{S_*}\|_2 \\ \overset{(d)}{\le}& \lambda\left\|X_{S_*}^T(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}) + \boldsymbol{\varepsilon}_{S_*}\right\|_2.\end{aligned}$$

Inequality *(c)* follows from the Lemma 5.3, where $\lambda = 1 + \frac{128(1-\gamma)}{2\gamma-1}$. Inequality *(d)* utilizes the definition of $\boldsymbol{r}_{S_*}$. Let $\mathrm{FP}_{t+1} = S_{t+1} \setminus S_*$, $\mathrm{FN}_{t+1} = S_* \setminus S_{t+1}$, $\mathrm{TP}_{t+1} = S_* \cap S_{t+1}$, and according to the

triangle inequality property, we have

$$\begin{aligned}
\|\boldsymbol{u}_{S_{t+1}}\|_2 \leq & \left\|X^T_{\mathrm{FP}_{t+1}}(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 + \|\boldsymbol{\varepsilon}_{\mathrm{FP}_{t+1}}\|_2 + \sqrt{\lambda^2-1}\left\|X^T_{\mathrm{TP}_{t+1}}(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 \\
& + \sqrt{\lambda^2-1}\|\boldsymbol{\varepsilon}_{\mathrm{TP}_{t+1}}\|_2 + \lambda\left\|X^T_{\mathrm{FN}_{t+1}}(\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1})\right\|_2 + \lambda\|\boldsymbol{\varepsilon}_{\mathrm{FN}_{t+1}}\|_2 \\
\overset{(e)}{\leq} & \varphi_{\alpha,\gamma}\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}\|_2 + (1+\lambda+\sqrt{\lambda^2-1})\|\boldsymbol{\varepsilon}\|_2 \\
\overset{(f)}{\leq} & \frac{2\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}}(\|\boldsymbol{u}_{S_t}\|_2 + \|\boldsymbol{\varepsilon}\|_2) + (1+2\lambda)\|\boldsymbol{\varepsilon}\|_2 \\
\leq & \frac{2\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}}\|\boldsymbol{u}_{S_t}\|_2 + \left[\frac{2\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}} + (1+2\lambda)\right]\|\boldsymbol{\varepsilon}\|_2.
\end{aligned}$$

Let $\varphi_{\alpha,\gamma} = \sqrt{\kappa_\alpha} + (\lambda + \sqrt{\lambda^2-1})\sqrt{\kappa_{1-\gamma}}$, inequality *(e)* follows from $|\mathrm{FP}_{t+1}| \leq \alpha n$, $|\mathrm{TP}_{t+1}| \leq (1-\gamma)n$, $|\mathrm{FN}_{t+1}| \leq (1-\gamma)n$ and $\|\boldsymbol{\varepsilon}_{S_{t+1}}\|_2 \leq \|\boldsymbol{\varepsilon}\|_2$. Inequality *(f)* follows from the fact in inequality *(b)*. Let $\eta = \frac{2\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}}$. When $\frac{\varphi_{\alpha,\gamma}}{\sqrt{\zeta_\alpha}} < \frac{1}{2}$, we have $\eta < 1$. Replacing the coefficients with η, we get

$$\begin{aligned}
\|\boldsymbol{u}_{S_{t+1}}\|_2 \leq & \eta\|\boldsymbol{u}_{S_t}\|_2 + (\eta + 2\lambda + 1)\|\boldsymbol{\varepsilon}\|_2 \\
\leq & \eta\|\boldsymbol{u}_{S_t}\|_2 + (2+2\lambda)\|\boldsymbol{\varepsilon}\|_2 \\
\leq & \eta^t\|\boldsymbol{u}\|_2 + (2+2\lambda)\sum_{i=1}^{t}\eta^{i-1}\|\boldsymbol{\varepsilon}\|_2 \\
\leq & \eta^t\|\boldsymbol{u}\|_2 + \frac{(2+2\lambda)}{1-\eta}\|\boldsymbol{\varepsilon}\|_2.
\end{aligned}$$

Using the inequality for $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}\|_2$ again gives us

$$\begin{aligned}
\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_{t+1}\|_2 \leq & \frac{2}{\sqrt{\zeta_\alpha}}(\|\boldsymbol{u}_{S_t}\|_2 + \|\boldsymbol{\varepsilon}\|_2) \\
\leq & \frac{2\eta^t}{\sqrt{\zeta_\alpha}}\|\boldsymbol{u}\|_2 + \frac{2(3+2\lambda-\eta)}{\sqrt{\zeta_\alpha}(1-\eta)}\|\boldsymbol{\varepsilon}\|_2.
\end{aligned}$$

Let $C = \frac{2(3+2\lambda-\eta)}{1-\eta}$, after $t = O(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}\|_2}{\sqrt{n}\epsilon})$ iterations, Algorithm 2 yields an ϵ-accurate solution $\boldsymbol{\beta}^t$ with $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_t\|_2 \leq \epsilon + \frac{C\|\boldsymbol{\varepsilon}\|_2}{\sqrt{n}}$. □

REFERENCES

[1] 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.

[2] Joel W. Branch, Chris Giannella, Boleslaw Szymanski, Ran Wolff, and Hillol Kargupta. 2013. In-network outlier detection in wireless sensor networks. *Knowledge and Information Systems* 34, 1 (2013), 23–54.

[3] Markus Breunig, Hans-Peter Kriegel, Raymond Ng, and Jörg Sander. 1999. Optics-of: Identifying local outliers. *Principles of Data Mining and Knowledge Discovery* (1999), 262–270.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. In *Proceedings of the ACM Sigmod Record*, Vol. 29. ACM, 93–104.

[5] Kai Chen, Qi Lv, Yao Lu, and Yong Dou. 2017. Robust regularized extreme learning machine for regression using iteratively reweighted least squares. *Neurocomputing* 230 (2017), 345–358.

[6] Yudong Chen, Constantine Caramanis, and Shie Mannor. 2013. Robust sparse regression under adversarial corruption. In *Proceedings of the 30th International Conference on Machine Learning*. 28, 3 (2013), 774–782.

[7] Gaudenz Danuser and Markus Stricker. 1998. Parametric model fitting: From inlier characterization to outlier detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 3 (1998), 263–280.

[8] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2014), 2250–2267.

[9] Victoria Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22, 2 (2004), 85–126.
[10] Chao Huang and Dong Wang. 2016. Topic-aware social sensing with arbitrary source dependency graphs. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*. IEEE Press, 7.
[11] Chao Huang, Dong Wang, and Nitesh Chawla. 2017. Scalable uncertainty-aware truth discovery in big data social sensing applications for cyber-physical systems. *IEEE Transactions on Big Data*. 1–1. DOI: 10.1109/TBDATA.2017.266930
[12] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. 2006. Extreme learning machine: Theory and applications. *Neurocomputing* 70, 1–3 (2006), 489–501.
[13] Peter J. Huber. 1973. Robust regression: Asymptotics, conjectures and Monte Carlo. *Annals of Statistics* 1, 5 (1973), 799–821. https://projecteuclid.org/euclid.aos/1176342503.
[14] Peter J. Huber and Elvezio M. Ronchetti. 2009. *The Basic Types of Estimates*. John Wiley & Sons, Inc., 45–70. DOI: https://doi.org/10.1002/9780470434697.ch3
[15] Wen Jin, Anthony K. H. Tung, and Jiawei Han. 2001. Mining top-n local outliers in large databases. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*. ACM, New York, NY, 293–298. DOI: https://doi.org/10.1145/502512.502554
[16] Yoonsuh Jung, Seung Pil Lee, and Jianhua Hu. 2016. Robust regression for highly corrupted response by shifting outliers. *Statistical Modelling* 16, 1 (2016), 1–23.
[17] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. 2007. Outlier detection with kernel density functions. In *Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 61–75.
[18] Ruirui Li, Xinxin Huang, Shuo Song, Jia Wang, and Wei Wang. 2016. Towards customer trouble tickets resolution automation in large cellular services. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 479–480.
[19] 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 Advances in Neural Information Processing Systems*. 2726–2734.
[20] V. M. Lourenco, Ana M. Pires, and M. Kirst. 2011. Robust linear regression methods in association studies. *Bioinformatics* 27, 6 (2011), 815–821.
[21] RARD Maronna, R Douglas Martin, and Victor Yohai. 2006. *Robust Statistics*. John Wiley & Sons, Chichester.
[22] 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*. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.), Curran Associates, Inc., 415–423. Retrieved from http://papers.nips.cc/paper/5428-fast-and-robust-least-squares-estimation-in-corrupted-linear-models.pdf.
[23] Imran Naseem, Roberto Togneri, and Mohammed Bennamoun. 2012. Robust regression for face recognition. *Pattern Recognition* 45, 1 (2012), 104–118.
[24] Hong-Wei Ng and Stefan Winkler. 2014. A data-driven approach to cleaning large face datasets. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'14)*. IEEE, 343–347.
[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] Volker Roth. 2006. Kernel fisher discriminants for outlier detection. *Neural Computation* 18, 4 (2006), 942–960.
[27] Peter J. Rousseeuw and Annick M. Leroy. 2005. *Robust Regression and Outlier Detection*, Vol. 589. John Wiley & Sons.
[28] Peter J. Rousseeuw and Katrien van Driessen. 2006. Computing LTS regression for large data sets. *Data Mining and Knowledge Discovery* 12, 1 (Jan. 2006), 29–45. DOI: https://doi.org/10.1007/s10618-005-0024-4
[29] Yiyuan She and Art B. Owen. 2011. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association* 106, 494 (2011), 626–639. http://www.jstor.org/stable/41416397.
[30] Helge Erik Solberg and Ari Lahti. 2005. Detection of outliers in reference distributions: Performance of Horn's algorithm. *Clinical Chemistry* 51, 12 (2005), 2326–2332.
[31] Christoph Studer, Patrick Kuppinger, Graeme Pope, and Helmut Bolcskei. 2012. Recovery of sparsely corrupted signals. *IEEE Transactions on Information Theory* 58, 5 (2012), 3115–3130.
[32] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. 2006. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd International Conference on Very Large Data Bases*. VLDB Endowment, 187–198.
[33] John Wright and Yi Ma. 2010. Dense error correction via L1-minimization. *IEEE Transactions on Information Theory* 56, 7 (Jul. 2010), 3540–3560. DOI: https://doi.org/10.1109/TIT.2010.2048473
[34] John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. 2009. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 2 (2009), 210–227.

[35] Xian Wu, Yuxiao Dong, Jun Tao, Chao Huang, and Nitesh V. Chawla. 2017. Reliable fake review detection via modeling temporal and behavioral patterns. In *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data'17)*. IEEE, 494–499.

[36] Allen Yang, Arvind Ganesh, Shankar Sastry, and Yi Ma. 2010. *Fast L1-Minimization Algorithms and an Application in Robust Face Recognition: A Review*. Technical Report UCB/EECS-2010-13. EECS Department, University of California, Berkeley. Retrieved from http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-13.html.

[37] 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.

[38] 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 (ICDM'18)*. IEEE, 1440–1445.

[39] 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 IEEE International Conference on Data Mining (ICDM'17)*. IEEE, 625–634.

[40] 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 (IJCAI'17)*. 3434–3440. DOI: https://doi.org/10.24963/ijcai.2017/480

[41] Hao Zhu, Henry Leung, and Zhongshi He. 2013. A variational Bayesian approach to robust sensor fusion based on Student-t distribution. *Information Sciences* 221, Supplement C (2013), 201–214. DOI: https://doi.org/10.1016/j.ins.2012.09.017

[42] Abdelhak M Zoubir, Visa Koivunen, Yacine Chakhchoukh, and Michael Muma. 2012. Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts. *IEEE Signal Processing Magazine* 29, 4 (2012), 61–80.

Received January 2018; revised October 2018; accepted February 2019