Scalable Robust Models Under Adversarial Data Corruption

Xuchao Zhang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Chang-Tien Lu (Chair)
Narendran Ramakrishnan
Ing-Ray Chen
Chandan Reddy
Arnold P. Boedihardjo

Feb 21, 2019
Falls Church, Virginia

Keywords: robust modeling, scalable algorithm, adversarial data corruption

Scalable Robust Models Under Adversarial Data Corruption

Xuchao Zhang

(ABSTRACT)

The presence of noise and corruption in real-world data can be inevitably caused by accidental outliers, transmission loss, or even adversarial data attacks. Unlike traditional random noise usually assume a specific distribution with low corruption ratio, the adversarial data corruption can be arbitrary, unbounded and do not follow any specific distribution.

This dissertation focuses on the development of methods for scalable robust models under the adversarial data corruption assumptions. Six methods are proposed, including robust regression via heuristic hard-thresholding, online and distributed robust regression with adversarial noises, self-paced robust learning for leveraging clean labels in noisy data, and robust regression via online feature selection with adversarial noises. Moreover, I extended the self-paced robust learning method to its distributed version for the scalability of the proposed algorithm. Last, a robust multi-factor personality prediction model is proposed to hand the correlated data noises.

For the first method, existing solutions for robust regression lack rigorous recovery guarantee of regression coefficients under the adversarial data corruption with no prior knowledge of corruption ratio. The proposed contributions of our work include: (1) Propose efficient algorithms to address the robust least-square regression problem; (2) Design effective approaches to estimate the corruption ratio; (3) Provide a rigorous robustness guarantee for regression coefficient recovery; and (4) Conduct extensive experiments for performance evaluation.

For the second method, existing robust learning methods typically focus on modeling the entire dataset at once; however, they may meet the bottleneck of memory and computation as more and more datasets are becoming too large to be handled integrally. The proposed contributions of our work for this task include: (1) Formulate a framework for the scalable robust least-squares regression problem; (2) Propose online and distributed algorithms to handle the adversarial corruption; (3) Provide a rigorous robustness guarantee for regression coefficient recovery; and (4) Conduct extensive experiments for performance evaluations.

For the third method, the presence of data corruption in user-generated streaming data, such as social media, motivates a new fundamental problem that learns reliable regression coefficient when features are not accessible entirely at one time. The proposed contributions of our work for this task include: (1) Propose a robust online feature substitution method to address the robust online feature selection problem; (2) Prove that our algorithm has a restricted error bound compared to the optimal solution; and (3) Conduct extensive empirical experiments in both synthetic and real-world data sets.

For the fourth method, leveraging the prior knowledge of clean labels in noisy data is actually a crucial issue in practice, but existing robust learning methods typically focus more on eliminating noisy data. However, the data collected by "weak annotator" or crowd-sourcing can

be too noisy for existing robust methods to train an accurate model. The proposed contributions of our work for this task include: (1) Formulating a framework to leverage the clean labels in noisy data; (2) Proposing a self-paced robust learning algorithm to train models under the supervision of clean labels; (3) Providing a theoretical analysis for the convergence of the proposed algorithm; and (4) Conducting extensive experiments for performance evaluations.

For the fifth method, existing self-paced learning approaches typically focus on modeling the entire dataset at once; however, this may introduce a bottleneck in terms of memory and computation, as today's fast-growing datasets are becoming too large to be handled integrally. The proposed contributions of our work for this task include: (1) Reformulate the self-paced problem into a distributed setting; (2) A distributed self-paced learning algorithm based on consensus ADMM is proposed to solve the SPL problem in a distributed setting; (3) A theoretical analysis is provided for the convergence of our proposed DSPL algorithm; and (4) Extensive experiments have been conducted utilizing both synthetic and real-world data based on a robust regression task.

For the last method, personality prediction in multiple factors, such as openness and agreeableness, is growing in interest especially in the context of social media, which contains massive online posts or likes that can potentially reveal an individual's personality. However, the data collected from social media inevitably contains massive amounts of noise and corruption. The proposed contributions of our work for this task include: (1) Propose a novel robust multi-factor personality prediction model to address the correlated noises in multiple factors; (2) Prove that our algorithm benefits from strong guarantees in terms of convergence rates and coefficient recovery; and (3) Conduct extensive experiment on synthetic and real dataset.

Scalable Robust Models Under Adversarial Data Corruption

Xuchao Zhang

(GENERAL AUDIENCE ABSTRACT)

Social media has experienced a rapid growth during the past decade. Millions of users of sites such as Twitter have been generating and sharing a wide variety of content including texts, images, and other metadata. In addition, social media can be treated as a social sensor that reflects different aspects of our society. Event analytics in social media have enormous significance for applications like disease surveillance, business intelligence, and disaster management. Social media data possesses a number of important characteristics including dynamics, heterogeneity, noisiness, timeliness, big volume, and network properties. These characteristics cause various new challenges and hence invoke many interesting research topics, which will be addressed here.

This dissertation focuses on the development of five novel methods for social media-based spatiotemporal event detection and forecasting. The first of these is a novel unsupervised approach for detecting the dynamic keywords of spatial events in targeted domains. This method has been deployed in a practical project for monitoring civil unrest events in several Latin American regions. The second builds on this by discovering the underlying development progress of events, jointly considering the structural contexts and spatiotemporal burstiness. The third seeks to forecast future events using social media data. The basic idea here is to search for subtle patterns in specific cities as indicators of ongoing or future events, where each pattern is defined as a burst of context features (keywords) that are relevant to a specific event. For instance, an initial expression of discontent gas price increases could actually be a potential precursor to a more general protest about government policies. Beyond social media data, in the fourth method proposed here, multiple data sources are leveraged to reflect different aspects of the society for event forecasting. This addresses several important problems, including the common phenomena that different sources may come from different geographical levels and have different available time periods. The fifth study is a novel flu forecasting method based on epidemics modeling and social media mining. A new framework is proposed to integrate prior knowledge of disease propagation mechanisms and real-time information from social media.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Chang-Tien Lu, for his support during my 4 years of Ph.D. study and related research. I have been incredibly fortunate to benefit from his guidance and invaluable insights on doing research. Beyond that, he has been a mentor and guide in my career and life. I will never forget his support and encouragement during the hardest part of my PhD years. I could not have imagined having a better advisor for my Ph.D. study.

I am very thankful to all my committee members: Prof. Naren Ramakrishnan, Prof. Ing-Ray Chen, Prof. Chandan Reddy, Dr. Arnold P. Boedihardjo, who have provided not only important guidance for the completion of my dissertation, but also for my research work during our collaborations. Thanks go to Dr. Naren Ramakrishnan and Dr. Arnold P. Boedihardjo for offering their fantastic leadership of our group's major project, which has provided me with support throughout my 4-year research project. Their deep expertise in writing and publishing has greatly benefited my research. Thanks also go to Dr. Chandan Reddy and Dr. Ing-Ray Chen for providing insightful comments and valuable suggestions from my preliminary proposal, through my research defense, and for my final defense. Their knowledge of the field suggested new research directions that I might not otherwise have recognized.

I would like to especially thank the enormous support and continuous help I have received from Prof. Liang Zhao over the last four years. His professional advises and enormous passion for research have been always an excellent example for me.

I would like to express my sincere thanks to my friends in the Spatial Data Management Laboratory: Ting Hua, Kaiqun Fu, Zhiqian Chen, Taoran Ji, Lei Zhang, Shuo Lei, Lulwah Alkulaib, Jianfeng Chen, Abdulaziz Alhamadani, to name but a few - with whom I have shared unique moments thoughout this time. Thanks for being around and being there for me whenever I needed you all. In the course of my Ph.D. study, they made the journey enjoyable with many happy memories.

Finally, I would like to dedicate this dissertation with special thanks to my wife, Yingwen Shao, my parents, Ming Zhang and Xiudi Jin, my parents-in-law, Weimin Shao and Xiaohe Hua, and my daughter, Nicole Zhang. These amazing people have provided me with enormous support and love throughout writing this thesis and my life.

Contents

List of Figures

List of Tables

Chapter 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 [117] can produce millions of data records every second in monitoring air quality, energy consumption, and traffic congestion. More challenging, the presence of noise and corruption in real-world data can be inevitably caused by accidental outliers [93], transmission loss [96], or even adversarial data attacks [18]. As the most popular statistical approach, the traditional least-squares regression method is vulnerable to outlier observations [69] and not scalable to large datasets [72]. By considering both robustness and scalability in a least-squares regression model, we study scalable robust least-squares regression ($SRLR$) to handle the problem of learning a reliable set of regression coefficients given a large dataset with several adversarial corruptions in its response vector. A commonly adopted model from existing robust regression methods [5] [122] assumes that the observed response is obtained from the generative model $\boldsymbol{y} = X^T\boldsymbol{\beta}_* + \boldsymbol{u}$, where $\boldsymbol{\beta}_*$ is the true regression coefficients that we wish to recover and $\boldsymbol{u}$ is the corruption vector with arbitrary values. However, in the $SRLR$ problem, our goal 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 loaded into a single machine. Due to the ubiquitousness of data corruptions 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 [133], and image processing [79].

This research focuses on the development of scalable robust models under adversarial data corruption, including including robust regression via heuristic hard-thresholding, online and distributed Robust Regression under adversarial data corruption. These tasks have a wide array of applications. Some of them are described as follows.

Scalable robust regression under adversarial data corruption is to recover the models in massive data sets when the data noises can be arbitrary and unbounded. This area is very close to the classic research on robust regression and online learning.

- **Anomaly and Outlier Detection:** Outlier detection, also known as anomaly detection, has been well studied in a wide range of practical applications [8,37,81]. Literature on this work can be broadly classified into two types [39]: parametric methods [23] and non-parametric methods [103]. Parametric outlier detection methods explicitly assume the probabilistic or distribution model for the training data, while the non-parametric methods do not make any prior assumption of the data distribution. In parametric outlier detection, some work utilized heavy-tailed distributions [129] such as Student t-distribution and Poisson distribution, to model the outliers, while others detected outliers based on Gaussian distribution [39, 101] 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 [56,92]. 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 [10,11,47] generally have a stronger capability of modeling outliers by investigating not only the neighbors but the local densities. For instance, Breuning et al. [11] 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.

- **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 [17] [62] [91], but these methods, modeling the corruption on stochastic distributions, cannot be applied to data that may exhibit malicious corruption [18]. Some studies assume the adversarial corruption in the data, but most of them lack the strong guarantee of regression coefficients recovery under the arbitrary corruption assumption [18] [72]. Chen et al. [18] 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. [72] proposed a sub-sampling algorithm for large-scale corrupted linear regression, but their recovery result is not close to an exact recovery [5]. To pursue exact recovery results for robust regression problem, some studies focused on L_1 penalty based convex formulations [109] [82]. However, these methods imposed severe restrictions on the data distribution such as row-sampling from an incoherent orthogonal matrix [82].

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 [99] 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. [19] require the upper bound of the outliers number, which is also difficult to estimate. Bhatia et al. [5] 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. [122] 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 [28] [68] to update estimates incrementally. For instance, Duchi et al. [28] 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 [30] and online passive aggressive algorithms [21] 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 [70] [6], most approaches such as MapReduce [24] 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 [26] [49] and those that utilize directly on the probability distributions [22] [29] [3]. For instance, Delage et al. [25] 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. [98] 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 adversarial corruption. Recently, Feng et al. [33] proposed an online robust learning approach that gives 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.

1.1 Research Issues

This research aims to investigate and develop robust models for adversarial data corruption in large-scale data set. The major research issues are stated as follows:

1.1.1 Robust regression via heuristic hard-thresholding

The presence of data noise and corruptions recently 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, several 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 paper proposes a novel Robust least-squares 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 based on adaptive corruption estimation variation ($RACT$) 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.

1.1.2 Online and distributed robust regressions under adversarial data corruption

In today's era of big data, robust least-squares regression becomes a more challenging problem when considering the adversarial corruption 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 paper 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. Furthermore, an online version of the distributed algorithm is proposed to incrementally update the existing estimates with new incoming data. 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.

1.1.3 Robust regression via online feature selection under adversarial noises

The presence of data corruption in user-generated streaming data, such as social media, motivates a new fundamental problem that learns reliable regression coefficient when features are not accessible entirely at one time. Until now, several important challenges still cannot be handled concurrently: 1) corrupted data estimation when only partial features are accessible; 2) online feature selection when data contains adversarial corruption; and 3) scaling to a massive dataset. This paper proposes a novel RObust regression algorithm via Online Feature Selection (*RoOFS*) that concurrently addresses all the above challenges. Specifically, the algorithm iteratively updates the regression coefficients and the uncorrupted set via a robust online feature substitution method. We also prove that our algorithm has a restricted error bound compared to the optimal solution. Extensive empirical experiments in both synthetic and real-world data sets demonstrated that the effectiveness of our new method is superior to that of existing methods in the recovery of both feature selection and regression coefficients, with very competitive efficiency.

1.1.4 Self-paced robust learning for leveraging clean labels in noisy data

The success of training accurate models strongly depends on the availability of precisely labeled datasets with a sufficient amount of data samples. However, many real-world datasets contain erroneously labeled samples that substantially hinders the learning of accurate models. Meanwhile, well-labeled data is usually expensive to obtain and contains a limited amount for training. In this paper, we consider the problem of training a robust model using large-scale noisy data in conjunction with a small set of clean labels. To leverage the information contained in the clean data, we propose a novel self-paced robust learning algorithm (*SPRL*) that trains the model in a process from more reliable (clean) data instances to less reliable (noisy) ones under the supervision of well-labeled data. The self-paced learn-

ing process hedges the risk of selecting corrupted data into the training set. Moreover, theoretical analysis on the convergence of the proposed algorithm is provided under mild assumptions. Extensive experiments on synthetic and real datasets demonstrate that our approach is superior in effectiveness and robustness to existing methods.

1.1.5 Distributed self-paced learning in alternating direction method of multiplier

Self-paced learning (*SPL*) mimics the cognitive process of humans, who generally learn from easy samples to hard ones. One key issue in *SPL* is the training process required for each instance weight depends on the other samples and thus cannot easily be run in a distributed manner in a large-scale dataset. In this paper, we reformulate the self-paced learning problem into a distributed setting and propose a novel Distributed Self-Paced Learning method (*DSPL*) to handle large scale datasets. Specifically, both the model and instance weights can be optimized in parallel for each batch based on a consensus alternating direction method of multipliers. We also prove the convergence of our algorithm under mild conditions. Extensive experiments on both synthetic and real datasets demonstrate that our approach is superior to those of existing methods.

1.1.6 Robust multi-factor personality prediction with correlated data corruption

Personality prediction in multiple factors, such as openness and agreeableness, is growing in interest especially in the context of social media, which contains massive online posts or likes that can potentially reveal an individual's personality. However, the data collected from social media inevitably contains massive amounts of noise and corruption. To address it, traditional robust methods still suffer from several important challenges, including 1) existence of correlated corruption among multiple factors, 2) difficulty in estimating the corruption ratio in multi-factor data, and 3) scalability to massive datasets. This paper proposes a novel robust multi-factor personality prediction model that concurrently addresses all the above challenges by developing a distributed robust regression algorithm. Specifically, the algorithm optimizes regression coefficients of each factor in parallel with a heuristically estimated corruption ratio and then consolidates the uncorrupted set from multiple factors in two strategies: global consensus and majority voting. We also prove that our algorithm benefits from strong guarantees in terms of convergence rates and coefficient recovery, which can be utilized as a generic framework for the multi-factor robust regression problem with correlated corruption property. Extensive experiment on synthetic and real dataset demonstrates that our algorithm is superior to those of existing methods in both effectiveness and efficiency.

1.2 Contributions

The major proposed research contributions can be stated as follows:

Robust regression via heuristic hard-thresholding

- **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 six 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.

Online and distributed robust regressions under adversarial data corruption

- **Formulating a framework for the SRLR problem.** A framework is proposed for scalable robust least-squares regression problem where the entire data with adversarial corruption is too large to store in memory all at once. Specifically, given a large dataset with adversarial corruptions, a reliable set of regression coefficients is learned with limited memory.

- **Proposing online and distributed algorithms to handle the adversarial 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.

- **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 enough.

- **Conducting extensive experiments for performance evaluations.** The proposed method was 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.

Robust regression via online feature selection under adversarial noises

- **Design of an efficient algorithm to simultaneously address the problem of data corruption and online feature**. The algorithm *RoOFS* is proposed to recover the regression coefficients and uncorrupted set efficiently. Unlike using entire features, our approach alternately estimates the data corruption and selects the feature set via a robust online feature substitution method.

- **Theoretical analysis of the algorithm**. We prove that our method yields a solution with a restricted error bound compared to ground truth coefficients under the Subset Restricted Strong Convexity ($SRSC$) property.

- **Demonstration of empirical effectiveness and efficiency**. Our proposed algorithm was evaluated with 6 competing methods in both robust regression and online feature selection literatures. The results showed that our approach consistently outperforms existing methods in coefficients recovery and uncorrupted set estimation, delivering a competitive running time.

Self-paced robust learning for leveraging clean labels in noisy data

- **Formulating a framework to leverage the clean labels in noisy data.** A framework is proposed to utilize clean labels in a large-scale noisy dataset. Specifically, the clean labels are assumed to contain a limited size of data while the noisy dataset may

contain an extremely large amount of data corruption. The approaches in solving robust regression and classification tasks are presented, which demonstrates the proposed framework can be generally used in various tasks.

- **Proposing a self-paced robust learning algorithm to train models under the supervision of clean labels.** The proposed self-paced algorithm learns the data samples from clean to noisy under the supervision of clean labels, which helps to hedge the risk of involving corrupted data into the training set. Furthermore, the algorithm learns the training data in an order that are dynamically determined by the feedback of the learner itself without additional prior knowledge, which makes it more extensively utilized in practice.

- **Providing a theoretical analysis for the convergence of the proposed algorithm.** We prove that our self-paced robust learning algorithm converges under the assumption that the loss function selected for the estimated model has a finite lower bound. Specifically, the objective function of our algorithm can be monotonically decreased in accord with the increasing learning pace parameter until it reaches the lower bound.

- **Conducting extensive experiments for performance evaluations.** The proposed method was evaluated on both synthetic data and real-world datasets in robust regression and classification tasks with different corruption and data-size settings. The results demonstrate that the proposed approaches consistently outperform existing methods along multiple metrics.

Distributed self-paced learning in alternating direction method of multiplier

- **Formulating the distributed self-paced problem.** We reformulate the self-paced problem into a distributed setting. Specifically, an auxiliary variable is introduced to decouple the dependency of the model parameters for each data batch

- **Proposing a distributed self-paced learning algorithm based on consensus ADMM.** A distributed self-paced learning algorithm based on consensus ADMM is proposed to solve the *SPL* problem in a distributed setting. The algorithm optimizes the model parameters for each batch in parallel and consolidates their values in each iteration.

- **Providing the theoretical analysis for the convergence of the proposed algorithm.** A theoretical analysis is provided for the convergence of our proposed *DSPL* algorithm. The proof shows that our new algorithm will converge under mild assumptions, e.g., the loss function can be non-convex.

- **Conducting extensive experiments to evaluate the proposed algorithm.** Extensive experiments have been conducted utilizing both synthetic and real-world data

based on a robust regression task. The results demonstrate that the proposed approaches consistently outperform existing methods for multiple data settings.

Robust multi-factor personality prediction with correlated data corruption

- **Formulating a model for robust multi-factor personality prediction.** The proposed model considers correlation of multiple personality factors by utilizing the correlated corruption property. Based on this property, each personality factor learns the estimated corruption pattern from the others to improve overall performance.
- **Proposing a distributed robust algorithm for the multi-factor regression problem.** The optimization of the proposed multi-factor model is a non-convex discrete optimization problem, which is technically challenging. By optimizing individual factor in parallel, the uncorrupted set is combined from each factor following two strategies: global consensus and majority voting.
- **Providing a strong recovery guarantee under the multi-factor problem setting.** We prove that our RMFP algorithm with global consensus strategy converges at a geometric rate and recovers coefficients of each factor exactly under the assumption of Subset Strong Convexity and Subset Strong Smoothness properties. Specifically, we prove that our algorithm ensures a rigorous error bound of the regression coefficient compared to ground truth.
- **Conducting extensive experiments for performance evaluations.** The proposed method was evaluated on both synthetic data and real-world datasets with various corruption settings. The results demonstrate that the proposed approach runs efficiently and consistently outperforms the best of the existing methods along multiple metrics.

1.3 Organization of the Dissertation

The remainder of this research proposal is organized as follows. Chapter 2 defines the robust regression algorithm via heuristic corruption thresholding, provide the theoretical error bound of the proposed algorithm, and presents experiments results and discussions. Chapter 3 describes the proposed both online and distributed robust regression model to handle the adversarial noises, prove the theoretical property of the proposed algorithms and then demonstrate the performance of proposed methods. Chapter 4 presents the online feature selection algorithm with adversarial noises, and analyzes the convergence of the proposed algorithm, and then shows the experimental results of proposed algorithm. Chapter 5 proposes a self-paced robust learning framework, and then present the evaluation results of the proposed method. Chapter 6 proposes the distributed self-paced learning framework

which is a distributed extension of self-paced robust learning method. Chapter 7 presents a robust model application in multi-factor personality prediction. Chapter 8 summarizes the work carried out, lists the associated publications, and suggests directions for future research.

Chapter 2

Robust Regression via Heuristic Corruption Thresholding

This chapter presents a novel heuristic-hard-thresholding based approach to handle the regression coefficients recovery problem under adversarial data corruption. First, the introduction of this chapter is presented in Section 2.1. Then, Section 2.2 reviews the background and related work in robust regression models and outlier detection methods. Section 2.3 gives a formal problem formulation. The proposed *RHCT* algorithm and its adaptive variation, *RACT*, are presented in Section 2.4. Section 2.5 presents the proof for the recovery guarantees. In Section 2.6, the experimental results on both synthetic and real-world datasets are analyzed, and the paper concludes with a summary of our work in Section 2.7.

2.1 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 [69], 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 [102, 134], economics [93], bioinformatics [63] and image processing [79, 110].

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 [5]. 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 [82] 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 [110] or row-sampled from an incoherent orthogonal matrix [82]. 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 [73] 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) [117] 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 paper 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 six 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.

2.2 Related Work

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

2.2.1 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 [41,62]; however, these methods cannot be applied to data that exhibits malicious corruption [19]. Some work discovers regression coefficients with adversarial data corruption [5, 19, 48], but most of them [48, 99] lack the theoretical guarantee of regression coefficients recovery. To theoretically guarantee the recovery performance, Chen et al. [19] 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. [73] proposed a sub-sampling algorithm for large-scale corrupted linear regression; however, the recovery result they provide is not close to an exact recovery [5]. To pursue the exact recovery results for the *RLSR* problem, some work focused on L_1 penalty based convex formulations [82, 109]. However, these methods imposed severe restrictions on the data distribution such as row-sampling from an incoherent orthogonal matrix [82]. Although robust regression methods, such as M-estimator [41] and least-trimmed squares [94], have been shown to be effective in many applications, their success relies on the proper choices of threshold parameters, such as corruption ratio [5] and regularization parameter [99], which are difficult to determine manually. For instance, Chen et al. [19] 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 [99] rely on a regularization parameter to control the size of the uncorrupted set based on soft-thresholding. Recently, Bhatia et al. [5] 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.

2.2.2 Anomaly and Outlier Detection

Outlier detection, also known as anomaly detection, has been well studied in a wide range of practical applications [8, 37, 81]. Literature on this work can be broadly classified into two types [39]: parametric methods [23] and non-parametric methods [103]. Parametric outlier detection methods explicitly assume the probabilistic or distribution model for the given data, while the non-parametric methods do not make any specific assumption on the data distribution. In parametric outlier detection, some work utilized heavy-tailed distributions [129] such as Student t-distribution and Poisson distribution, to model the outliers, while others detected outliers based on Gaussian distribution [39, 101] 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 approaches relying on distribution 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 [56, 92]. 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 [10,11,47] generally have a stronger capability of modeling outliers by investigating not only the neighbors but the local densities. For instance, Breuning et al. [11] 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.

2.3 Problem Formulation

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

$$\boldsymbol{y} = \boldsymbol{y}^* + \boldsymbol{u} + \boldsymbol{\varepsilon} \tag{2.1}$$

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:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \arg\min_{\boldsymbol{\beta}, S} \|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\|_2^2 \\ s.t.\ \ & S \subset [n],\ |S| \geq \mathcal{G}(\boldsymbol{\beta}) \end{aligned} \tag{2.2}$$

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 2.4.

The optimization problem in Equation (7.3) 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

Table 2.1: 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

squares function satisfies the *Subset Strong Convexity (SSC)* and *Subset Strong Smoothness (SSS)* properties, which are defined as follows:

Definition 1 (**SSC and SSS Properties**). *The least-squares function* $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\|_2^2$ *satisfies* $2\zeta_\alpha$*-Subset Strong Convexity property and* $2\kappa_\alpha$*-Subset Strong Smoothness property if the following holds:*

$$\zeta_\alpha I \preceq \frac{1}{2}\nabla^2 f_S(\boldsymbol{\beta}) \preceq \kappa_\alpha I \quad \text{for } \forall S \in S_\alpha \tag{2.3}$$

Equation (7.8) *is equivalent to*

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

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 2.5 to prove the theoretical guarantee on the recovery of regression coefficients.

2.4 Proposed Methodology

We propose a robust regression algorithm via heuristic corruption thresholding, *RHCT*, in Section 2.4.1. To improve the efficiency of the corruption estimation, an adaptive corruption thresholding based algorithm, *RACT*, is proposed in Section 2.4.2.

2.4.1 Robust Regression via Heuristic Corruption Thresholding

In order to solve the problem in Equation (7.3) 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 (7.3) 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 k^{th} 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:

$$\hat{\tau} = \arg\min_{\tau} \mathcal{L}(\tau) \qquad s.t. \quad r_{\delta(\tau)} \leq \frac{2\tau r_{\delta(\tau_o)}}{\tau_o} \tag{2.4}$$

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

$$\mathcal{L}(\tau) := \frac{(r_{\delta(\tau)} + 1)/\tau}{(r_{\delta(n)} - r_{\delta(\tau)})/(n - \tau)} \tag{2.5}$$

The variable τ_o in the constraint is defined as follows:

$$\tau_o = \arg\min_{1 \leq \tau \leq n} \left| r^2_{\delta(\tau)} - \frac{\|r_{S_{\tau'}}\|_2^2}{\tau'} \right| \qquad s.t. \;\; \tau_o \in \mathbb{Z}^+ \tag{2.6}$$

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 2 (**Hard Thresholding Operator**). *Defining $\delta^{-1}_{\boldsymbol{r}}(i)$ as the position of the i^{th} element in residual vector $\boldsymbol{r}$'s ascending order of magnitude, the heuristic hard thresholding of $\boldsymbol{r}$ is defined as*

$$\mathcal{H}_{\hat{\tau}}(\boldsymbol{r}) = \{i \in [n] : \delta^{-1}_{\boldsymbol{r}}(i) \leq \hat{\tau}\} \tag{2.7}$$

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 2.5. 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 (7.1), where the corrupted samples have the residual $\boldsymbol{r} \approx \boldsymbol{u} + \boldsymbol{\varepsilon}$, but the residual

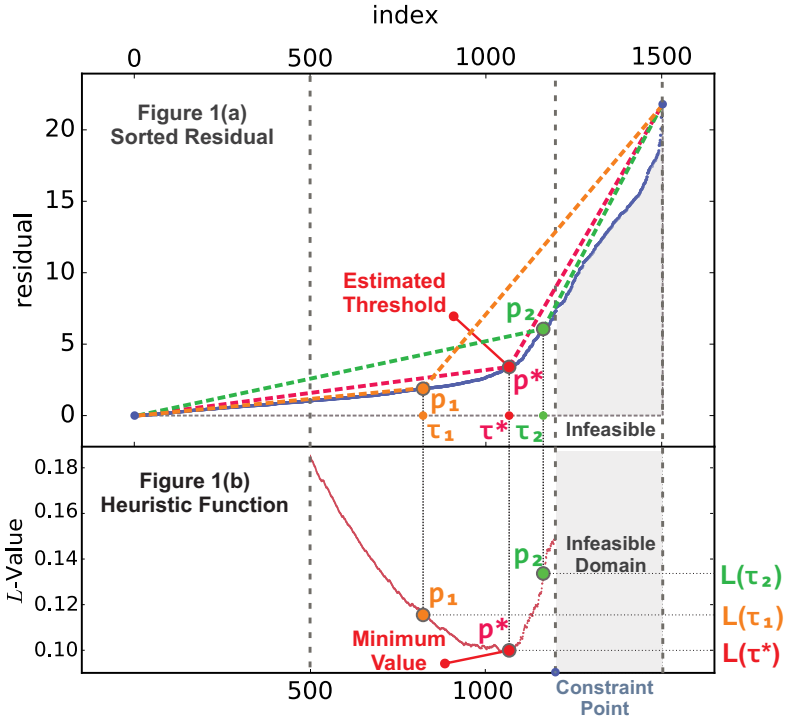


Figure 2.1: The blue line in Figure 1(a) indicates the values of residual vector $\boldsymbol{r}$ in ascending order, and the red point in Figure 1(b) shows the corresponding value of the heuristic function. τ_* is the estimated threshold with the minimum $\mathcal{L}$; τ_1 and τ_2 are the candidate threshold values in τ_*'s left- and right-hand sides, respectively.

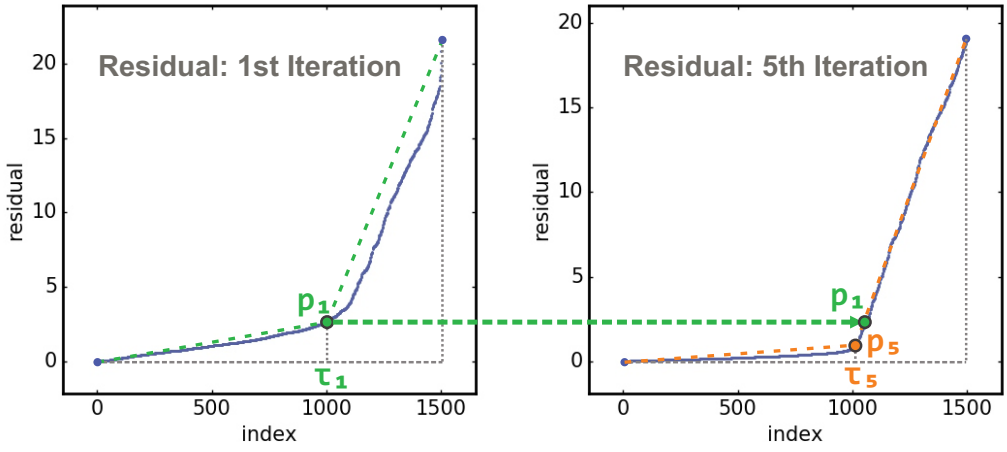


Figure 2.2: Residual $\boldsymbol{r}$ in ascending order for the 1st (left) and 5th (right) iterations of *RHCT* algorithm.

Algorithm 1 HEURISTIC CORRUPTION THRESHOLDING

Input: Residual vector $\boldsymbol{r}$, sample number n.
Output: Uncorrupted Set $\hat{S}$
$\boldsymbol{l} \leftarrow \mathbf{0}$, $i \leftarrow 1$
for $j = 1..n$ **do**
  $l_j \leftarrow \frac{(r_{\delta(j)}+1)/j}{(r_{\delta(n)}-r_{\delta(j)})/(n-j)}$ *// compute the heuristic value for each data sample.*
end
repeat
  $\tau \leftarrow \delta_{\boldsymbol{l}}^{-1}(i)$
    $\tau' \leftarrow \tau - \lceil n/2 \rceil$
    $\tau_o \leftarrow \arg\min_{1 \le k \le n} \left| r_{\delta(k)}^2 - \frac{\|\boldsymbol{r}_{S_{\tau'}}\|_2^2}{\tau'} \right| \quad s.t. \;\; k \in \mathbb{Z}^+$ *// compute the value of τ_o based on current τ.*
  if $r_{\delta(\tau)} \le \frac{2\tau r_{\delta(\tau_o)}}{\tau_o}$ **then**
    return $\mathcal{H}_\tau(\boldsymbol{r})$ *// return the heuristic hard thresholding $\mathcal{H}(\cdot)$ when constraint is satisfied.*
  end
  $i \leftarrow i + 1$
until $i \le \lceil n/2 \rceil$
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.*

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 2.1 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 (7.3). 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 (7.3).

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

Algorithm 2 RHCT ALGORITHM

Input: Corrupted training data $\{\boldsymbol{x}_i, y_i\}$, i = 1...n, tolerance ϵ
Output: solution $\hat{\boldsymbol{\beta}}$
$S_0 = [\text{n}]$, $t \leftarrow 0$
repeat
  $\boldsymbol{\beta}^{t+1} \leftarrow (X_{St}X_{St}^T)^{-1}X_{St}\boldsymbol{y}_{St}$ *// Update the regression coefficients $\boldsymbol{\beta}$ with estimated corruption set S_t.*
  for *$i = 1..n$* **do**
    $r_i^{t+1} \leftarrow |y_{S_ti} - \boldsymbol{x}_{S_ti}^T\boldsymbol{\beta}|$ *// Update the residual value for each data sample.*
  end
  $S_{t+1} \leftarrow HCT(\boldsymbol{r}^{t+1})$ *// calculate the estimated corruption set S_{t+1} based on residual vector $\boldsymbol{r}^{t+1}$.*
  $t \leftarrow t + 1$
until $\|\boldsymbol{r}_{S_{t+1}}^{t+1} - \boldsymbol{r}_{S_t}^t\|_2 < \epsilon n$
return $\boldsymbol{\beta}^{t+1}$

using all the data samples in Line 1. The algorithm continues until the change in the residual vector falls within a small range. Figure 7.1 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 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.

2.4.2 Adaptive Corruption Thresholding

The *RHCT* algorithm estimates the uncorrupted set S by computing the heuristic function $\mathcal{L}$ values in Equation (2.5) 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 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

$$\hat{\tau} := \underset{\lceil n/2 \rceil < \tau \leq n}{\arg\max}\ \tau \quad s.t.\ \ r_{\delta(\tau)} \leq \frac{2\tau r_{\delta(\tau_o)}}{\tau_o}, \tau \in \mathbb{Z}^+ \tag{2.8}$$

where $r_{\delta(k)}$ represents the k^{th} 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 (7.4), 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 (2.8). 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 (2.8) can be rewritten as $\Delta \leq 1$.

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 $\boldsymbol{r}_{\delta(\tau)} = X^T_{\delta(\tau)}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^t) + \boldsymbol{u}_{\delta(\tau)} + \boldsymbol{\varepsilon}_{\delta(\tau)}$ 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 (2.8) 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.

Algorithm 3 Adaptive Corruption Thresholding

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

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

The detailed steps of the adaptive corruption thresholding method are shown in Algorithm

3. The uncorrupted size τ_i of the i^{th} 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 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.

2.5 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 **??**.

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

Proof. Let $S_3 = \{\delta(i) : \tau_1 + 1 \leq i \leq \tau_2\}$. 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 $\|\boldsymbol{r}_{S_1}\|_2^2 \leq \|\boldsymbol{r}_{S_2}\|_2^2 + \frac{|S_3|}{|S_1|}\|\boldsymbol{r}_{S_1}\|_2^2 \leq \frac{|S_1|}{|S_1|+|S_3|}\|\boldsymbol{r}_{S_2}\|_2^2 = \frac{\tau_1}{\tau_2}\|\boldsymbol{r}_{S_2}\|_2^2 \leq \|\boldsymbol{r}_{S_2}\|_2^2$.

□

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

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

$$|S_* \cap S_t| \geq \tau_t - |[n] \setminus S_*| = \tau_t - (1 - \gamma)n \tag{2.9}$$

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

$$|S_* \cap S_t| \geq \tau_t - n + \frac{n}{n} = \tau_t - \frac{n}{2} \tag{2.10}$$

□

Lemma 3. *Let τ_t be the estimated uncorrupted threshold at the t^{th} iteration. If $\tau_t > \tau_* = \gamma n$, then $\|\boldsymbol{r}^t_{S_t}\|_2^2 \leq \left[1 + \frac{128(1-\gamma)}{2\gamma - 1}\right] \|\boldsymbol{r}^t_{S_*}\|_2^2$.*

Proof. To simplify the notation, we will omit all the subscripts t that signify the t^{th} 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 (7.4), we have the following properties:

$$\begin{aligned} r_\tau \leq & 2 \cdot \frac{\tau r_{\tau_o}}{\tau_o} \overset{(a)}{\leq} 8 \cdot r_{\tau_o} \\ r_\tau^2 \overset{(b)}{\leq} & \frac{64}{\tau'} \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \\ |S_t \setminus S_*| r_\tau^2 \overset{(c)}{\leq} & (1-\gamma) \cdot n \cdot \frac{64}{\tau'} \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \end{aligned} \tag{2.11}$$

Inequality *(a)* follows from $\tau_o \geq \tau/4$, and inequality *(b)* follows from the definition of τ_o in Equation (7.5) and the fact that $|S_* \cap S_t| \geq \tau'$ in Lemma 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

$$\begin{aligned} \|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 \leq & \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'} + 1\right] \|\boldsymbol{r}_{S_* \setminus S_t}\|_2^2 \\ & + \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'}\right] \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \\ \|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 + \|\boldsymbol{r}_{S_* \cap S_t}\|_2^2 \overset{(d)}{\leq} & \left[(1-\gamma) \cdot n \cdot \frac{64}{\tau'} + 1\right] \|\boldsymbol{r}_{S_*}\|_2^2 \\ \|\boldsymbol{r}_{S_t}\|_2^2 \overset{(e)}{\leq} & \left[1 + \frac{128(1-\gamma)}{2\gamma - 1}\right] \|\boldsymbol{r}_{S_*}\|_2^2 \end{aligned} \tag{2.12}$$

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 1. *Let $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^p$ 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 $t = \mathcal{O}\left(\log_{\frac{1}{\eta}} \frac{\mu \|\boldsymbol{u}\|_2}{\epsilon}\right)$ 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 t^{th} iteration of Algorithm 2 satisfies

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

Thus, the residual in the $(t+1)^{\text{th}}$ 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*, $\tau^{t+1} \le \tau_*$, and we have $\|\boldsymbol{r}_{S_{t+1}}^{t+1}\|_2^2 \le \|\boldsymbol{r}_{S_*}^{t+1}\|_2^2$ (see Lemma 6).

$$\begin{aligned}
\|\boldsymbol{u}_{S_{t+1}}\|_2^2 &= \|\boldsymbol{u}_{S_{t+1}} - X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 - \|X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 \\
&\quad + 2\boldsymbol{u}_{S_{t+1}}^TX_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t} \\
&\overset{(a)}{\le} \|X_{S_*}^TG_t\boldsymbol{u}_{S_t}\|_2^2 - \|X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 + 2\boldsymbol{u}_{S_{t+1}}^TX_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t} \\
&\overset{(b)}{\le} \frac{\kappa_{\alpha_1}^2}{\zeta_{1-\gamma}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\frac{\kappa_{\alpha_1}}{\zeta_{1-\gamma}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2
\end{aligned} \tag{2.13}$$

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

$$\|\boldsymbol{u}_{S_{t+1}}\|_2 \le (1+\sqrt{2})\frac{\kappa_{\alpha_1}}{\zeta_{1-\gamma}}\|\boldsymbol{u}_{S_t}\|_2 \tag{2.14}$$

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

$$\begin{aligned}
\|\boldsymbol{u}_{S_{t+1}}\|_2^2 &= \|\boldsymbol{u}_{S_{t+1}} - X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 - \|X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 \\
&\quad + 2\boldsymbol{u}_{S_t}^TX_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t} \\
&\overset{(c)}{\le} \lambda\|X_{S_*}^TG_t\boldsymbol{u}_{S_t}\|_2^2 - \|X_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t}\|_2^2 + 2\boldsymbol{u}_{S_t}^TX_{S_{t+1}}^TG_t\boldsymbol{u}_{S_t} \\
&\overset{(d)}{\le} \lambda\frac{\kappa_\gamma^2}{\zeta_{1-\alpha_2}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2 \\
&\overset{(e)}{\le} \lambda\frac{\kappa_\gamma^2}{\zeta_{1-\alpha_2}^2}\|\boldsymbol{u}_{S_t}\|_2^2 + 2\sqrt{\lambda}\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}_{S_t}\|_2\|\boldsymbol{u}_{S_{t+1}}\|_2
\end{aligned} \tag{2.15}$$

where $\alpha_2 = \max_t\{1 - \frac{\tau_t}{n}\}$. Inequality *(c)* follows from Lemma 2, and inequality *(d)* follows from the definition of the *SSC*/*SSS* properties, $|S_t| \le (1-\alpha_2)\cdot n$, and $|S_* \setminus S_{t+1}| \le \gamma \cdot n$. Inequality *(e)* follows from the fact that $\sqrt{\lambda} \ge 1$. Solving the quadratic equation in Equation (2.15) gives us

$$\|\boldsymbol{u}_{S_{t+1}}\|_2 \leq (1+\sqrt{2})\sqrt{\lambda}\frac{\kappa_\gamma}{\zeta_{1-\alpha_2}}\|\boldsymbol{u}\|_2 \tag{2.16}$$

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 $\mu = \max\left\{\frac{\sqrt{\kappa_{\alpha_1}}}{\zeta_{1-\gamma}}, \frac{\sqrt{\kappa_\gamma}}{\zeta_{1-\alpha_2}}\right\}$, $\eta_1 = \frac{(1+\sqrt{2})\kappa_{\alpha_1}}{\zeta_{1-\gamma}}$, 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 $t = \mathcal{O}\left(\log_{\frac{1}{\eta}}\frac{\mu\|\boldsymbol{u}\|_2}{\epsilon}\right)$, $\|\boldsymbol{\beta}^{t+1}-\boldsymbol{\beta}^*\|_2 \leq \epsilon$. □

2.6 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]

2.6.1 Experiment Setup

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 [5], the simulation samples were randomly generated according to the model in Equation (7.1) 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-τ_*)-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

[1] `https://github.com/xuczhang/RHCT_ACT`

[2] `https://www.airbnb.com/`

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.

Evaluation Metrics

For the synthetic data, performance of the regression coefficients recovery is measured by the standard L_2 error:

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

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 mean absolute error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ can be presented as follows.

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

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.

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 regularized L_1 algorithm for robust regression [109] [82]. For extensive L_1 minimization solvers, Yang et al. [113] showed that the *Homotopy* and *DALM* solvers outperform other proposed methods both in terms of recovery properties and running time. Both of the L_1 solver methods are parameter free. Another recently proposed hard thresholding method [5], *Torrent (abbr. Torr)*, developed for robust regression, was also compared to our method.

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

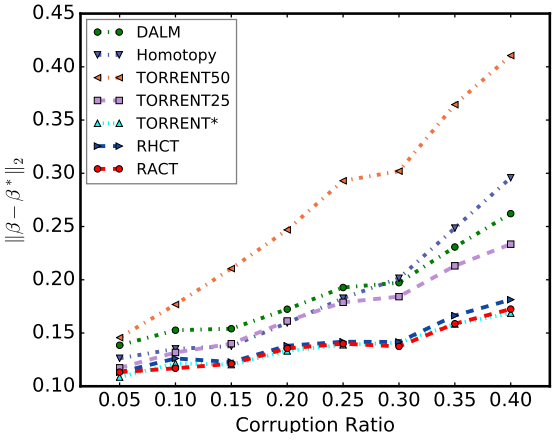
As the method requires a parameter for the corruption ratio, which is difficult to estimate in practice, we chose four versions with different parameter settings: *TORR**, *TORR25*, *TORR50*, and *TORR80*. *TORR** uses the true corruption ratio as its parameter, and the others apply parameters that are 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.

2.6.2 Recovery of Regression Coefficients

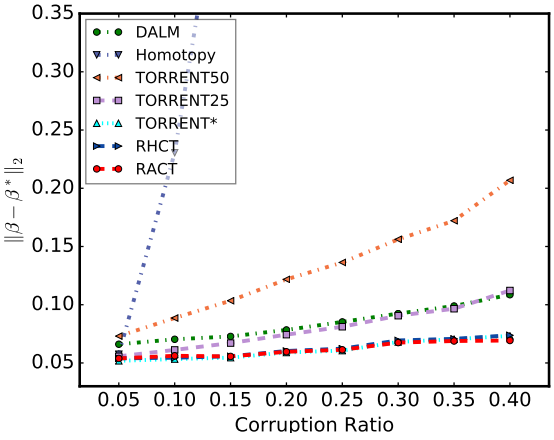
We selected six competing methods with which to evaluate the recovery performance of regression coefficients β: *OLS*, *DALM*, *Homotopy*, *TORR**, *TORR25*, and *TORR50*. As the recovery error for the *OLS* method is almost 10 times larger than those of the other methods, its result is not shown in Figure 7.2 in order to present the other results properly. Figures 7.2(a) and 7.2(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 7.2(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 7.2(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 7.2(e) and 7.2(f) show that *RHCT* and *RACT* perform equally as well as *TORR** without dense noise, with all achieving almost exact recovery of regression coefficients $\boldsymbol{\beta}$.

2.6.3 Recovery of Uncorrupted Sets

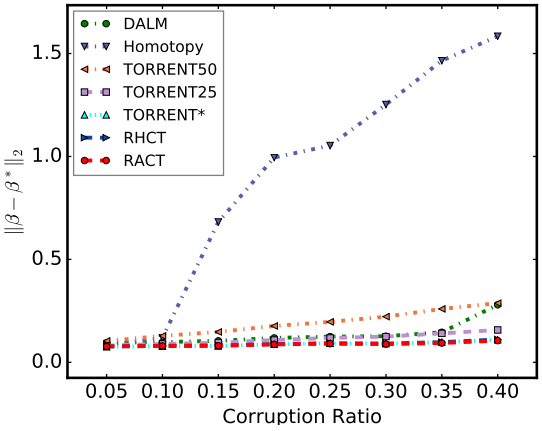
As most competing methods do not explicitly estimate uncorrupted sets, we compared our proposed methods with the *TORR* algorithm using a number of different parameter settings ranging from the true corrupted ratio up to a deviation of 80%. As the results show in Table 4.2, we found the following: 1) 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



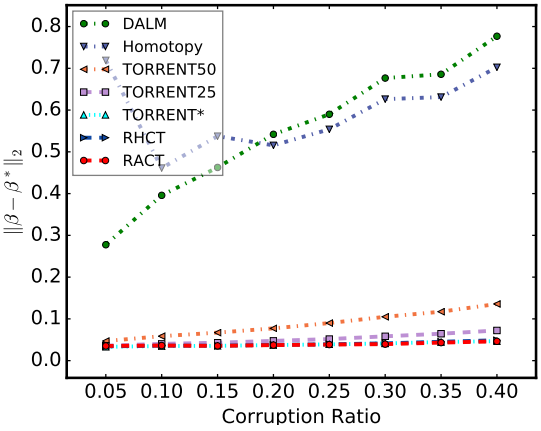
(a) p=100, n=1K, dense noise



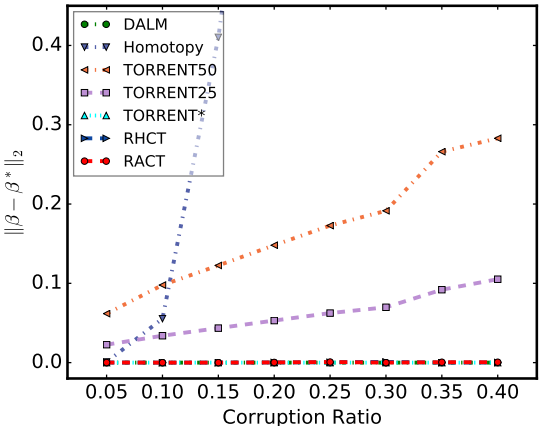
(b) p=100, n=4K, dense noise



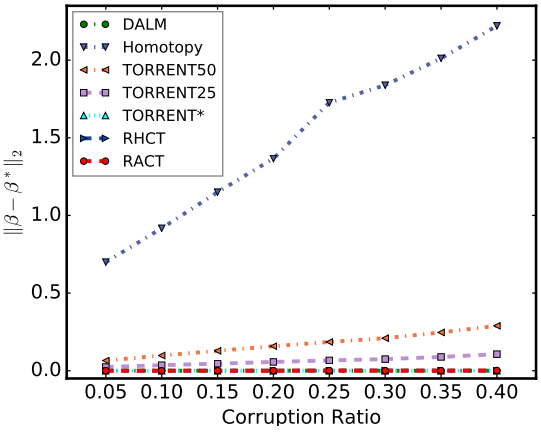
(c) p=200, n=4K, dense noise



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



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



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

Figure 2.3: Performance on regression coefficients recovery.

Table 2.2: 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**

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 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 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

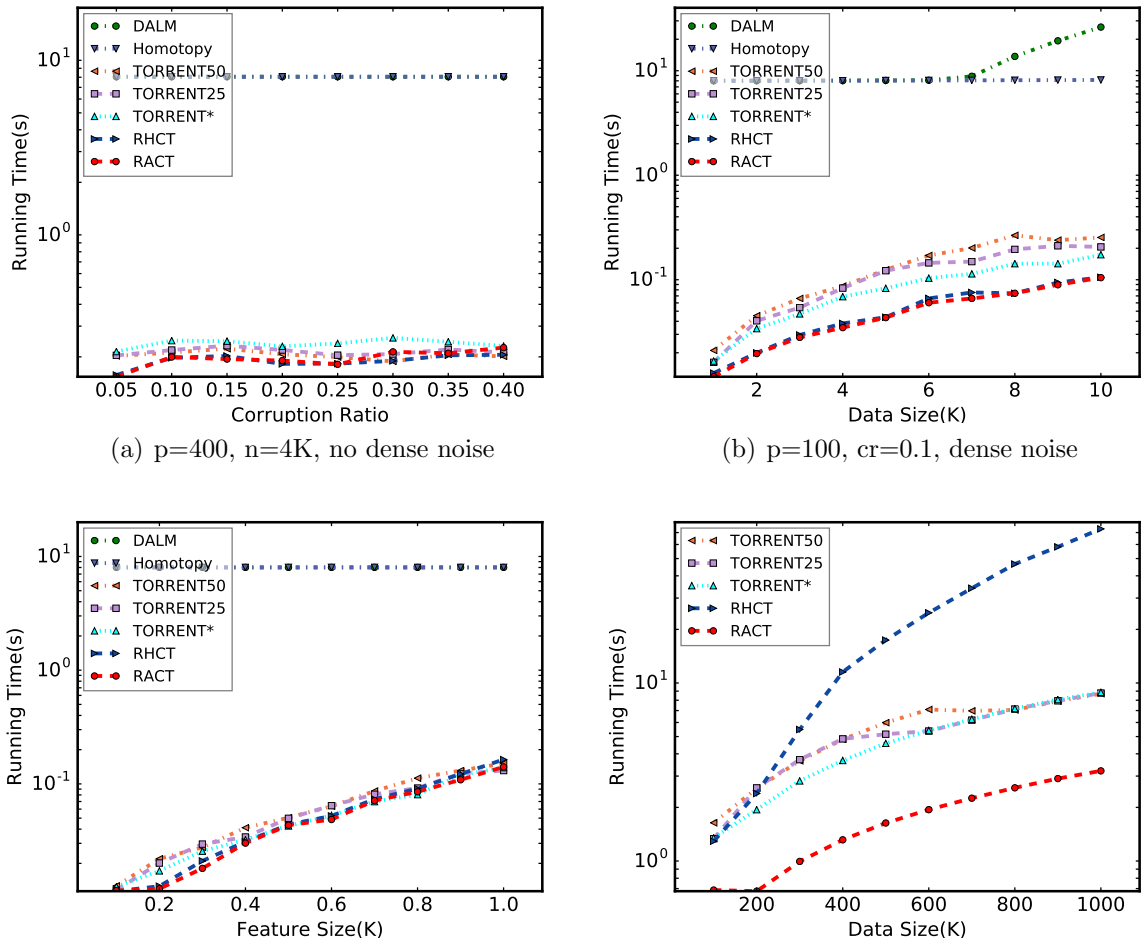
Table 2.3: Mean Absolute Error of Rental Price Prediction

	New York City (Corruption Ratio)					
	5%	**10%**	**20%**	**30%**	**40%**	**Avg.**
OLS	17.699±2.561	20.569±3.002	23.437±1.333	22.030±5.333	26.124±5.121	21.9718±3.47
DALM	58.187±12.457	53.274±19.055	63.658±27.091	84.866±21.100	67.230±27.965	61.443±21.5336
Homotopy	55.229±15.189	41.759±33.872	56.750±32.691	84.638±21.995	67.191±27.843	57.1134±26.318
TORR25	5.610±0.607	5.840±0.918	7.901±1.193	9.936±1.539	10.451±2.145	7.9476±1.2804
TORR50	8.135±1.107	9.051±1.705	11.960±1.653	13.994±1.431	14.598±1.761	11.5476±1.5314
RHCT	**2.824±0.001**	**2.824±0.001**	**2.825±0.002**	**2.827±0.002**	2.826±0.003	**2.8252±0.0018**
RACT	2.832±0.005	2.836±0.010	2.828±0.004	2.827±0.004	**2.824±0.001**	2.8294±0.0048
TORR*	**2.823±0.000**	**2.823±0.000**	**2.823±0.000**	**2.823±0.000**	**2.823±0.000**	**2.823±0.000**
	Los Angeles (Corruption Ratio)					
	5%	**10%**	**20%**	**30%**	**40%**	**Avg.**
OLS	22.860±1.897	27.326±7.038	31.130±5.173	37.234±6.473	42.833±4.290	32.2766±4.9742
DALM	46.700±12.764	54.572±6.245	72.702±2.590	72.011±13.405	48.056±15.309	58.8082±10.0626
Homotopy	40.682±12.510	51.578±4.661	71.229±2.371	71.690±13.777	43.499±22.270	55.7356±11.1178
TORR25	8.192±1.203	9.595±0.971	10.608±1.455	14.142±2.245	14.430±1.966	11.3934±1.568
TORR50	10.723±1.006	11.876±2.178	15.205±3.928	18.496±2.286	18.914±3.421	15.0428±2.5638
RHCT	**3.992±0.001**	**3.992±0.001**	**3.995±0.003**	4.065±0.024	4.067±0.087	4.0222±0.0232
RACT	4.077±0.067	4.022±0.028	4.005±0.018	**3.993±0.002**	**3.994±0.003**	**4.0182±0.0236**
TORR*	**3.988±0.000**	**3.988±0.000**	**3.988±0.000**	**3.988±0.000**	**3.989±0.000**	**3.9882±0.000**

noise, *RHCT* outperforms *RACT* in the no-dense-noise setting.

2.6.4 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 i^{th} sample data. We selected six competing methods with which to evaluate rental price prediction performance: *OLS*, *DALM*, *Homotopy*, *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 3.3 shows the mean absolute error 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) The *TORR** method outperforms all the other methods; however, the method



(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

Figure 2.4: Running time for different corruption ratios and data sizes

requires a corruption ratio parameter which is hard to estimate in practice. 3) 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. 4) Both the *DALM* and *Homotopy* methods perform even worse than the *OLS* method because their results are combined with results in small batches due to their scalability issues. It is possible for each small batch to contain a large amount of corrupted data.

2.6.5 Efficiency

To evaluate the efficiency of our proposed method, we compared the performance of all the competing methods for three different data settings: different corruption ratios, data sizes, and feature numbers in Figure 7.4(a)-7.4(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 2.4(d). In general, as Figure 7.4 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 7.4(a) shows that the corruption ratio has little impact on the 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 2.4(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) *RACT* outperforms all the competing methods when the data size is larger than 100,000 in Figure 2.4(d) because its corrupted set is adaptively estimated without computing heuristic values as in *RHCT*, which makes *RACT* less impacted by the data size.

2.7 Conclusion

In this paper, 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 subset strong convexity 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.

Chapter 3

Online and Distributed Robust Regression with Adversarial Noises

This chapter presents novel online and distributed robust regression approaches to handle the extremely large dataset with adversarial data corruption. First, Section 3.1 provides an introduction of the work. Then, Section 3.2 reviews background and related work, and Section 3.3 introduces the problem setup. The proposed online and distributed robust regression algorithms are presented in Section 3.4. Section 3.5 presents the proof of recovery guarantee in regression coefficients. The experiments on both synthetic and real-world datasets are presented in Section 3.6, and the paper concludes with a summary of the research in Section 3.7.

3.1 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 [117] can produce millions of data records every second in monitoring air quality, energy consumption, and traffic congestion. More challenging, the presence of noise and corruption in real-world data can be inevitably caused by accidental outliers [93], transmission loss [96], or even adversarial data attacks [18]. As the most popular statistical approach, the traditional least-squares regression method is vulnerable to outlier observations [69] and not scalable to large datasets [72]. By considering both robustness and scalability in a least-squares regression model, we study scalable robust least-squares regression ($SRLR$) to handle the problem of learning a reliable set of regression coefficients given a large dataset with several adversarial corruptions in its response vector. A commonly adopted model from existing robust regression methods [5] [122] assumes that the observed response is obtained from the generative model $\boldsymbol{y} = X^T\boldsymbol{\beta}_* + \boldsymbol{u}$, where $\boldsymbol{\beta}_*$ is the true regression coefficients that we wish to

recover and $\boldsymbol{u}$ is the corruption vector with arbitrary values. However, in the *SRLR* problem, our goal 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 loaded into a single machine. Due to the ubiquitousness of data corruptions 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 [133], and image processing [79].

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 paper presents a novel Distributed Robust Least-squares Regression (*DRLR*) method and its online version, named Online Roubst Least-squares Regression (*ORLR*) to handle the scalable robust regression problem in large datasets with adversarial 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. Also, we prove that both *DRLR* and *ORLR* preserve the overall robustness of regression coefficients in the entire dataset. The main contributions of this paper are as follows:

- **Formulating a framework for the SRLR problem.** A framework is proposed for scalable robust least-squares regression problem where the entire data with adversarial corruption is too large to store in memory all at once. Specifically, given a large dataset with adversarial corruptions, a reliable set of regression coefficients is learned with limited memory.
- **Proposing online and distributed algorithms to handle the adversarial 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.
- **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 enough.
- **Conducting extensive experiments for performance evaluations.** The proposed method was 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.

3.2 Related Work

The work related to this paper 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 [17] [62] [91], but these methods, modeling the corruption on stochastic distributions, cannot be applied to data that may exhibit malicious corruption [18]. Some studies assume the adversarial corruption in the data, but most of them lack the strong guarantee of regression coefficients recovery under the arbitrary corruption assumption [18] [72]. Chen et al. [18] 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. [72] proposed a sub-sampling algorithm for large-scale corrupted linear regression, but their recovery result is not close to an exact recovery [5]. To pursue exact recovery results for robust regression problem, some studies focused on L_1 penalty based convex

formulations [109] [82]. However, these methods imposed severe restrictions on the data distribution such as row-sampling from an incoherent orthogonal matrix [82].

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 [99] 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. [19] require the upper bound of the outliers number, which is also difficult to estimate. Bhatia et al. [5] 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. [122] 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 [28] [68] to update estimates incrementally. For instance, Duchi et al. [28] 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 [30] and online passive aggressive algorithms [21] 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 [70] [6], most approaches such as MapReduce [24] 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 [26] [49] and those that utilize directly on the probability distributions [22] [29] [3]. For instance, Delage et al. [25] 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. [98] 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 adversarial corruption. Recently, Feng et al. [33] proposed an online robust learning approach that gives 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.

3.3 Problem Formulation

In this section, the problem addressed by this research is formulated.

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 i^{th} batch. We assume the corresponding response vector $\boldsymbol{y}^{(i)} \in \mathbb{R}^{n \times 1}$ is generated using the following model:

$$\boldsymbol{y}^{(i)} = \left[\boldsymbol{X}^{(i)}\right]^T \boldsymbol{\beta}_* + \boldsymbol{u}^{(i)} + \boldsymbol{\varepsilon}^{(i)} \tag{3.1}$$

where $\boldsymbol{\beta}_* \in \mathbb{R}^{p \times 1}$ is the ground truth coefficients of the regression model and $\boldsymbol{u}^{(i)}$ is the adversarial corruption vector of the i^{th} mini-batch. $\boldsymbol{\varepsilon}^{(i)}$ represents the additive dense noise for the i^{th} mini batch, where $\varepsilon_j^{(i)} \sim \mathcal{N}(0, \sigma^2)$. The notations used in this paper are summarized in Table 7.2.

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:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \arg\min_{\boldsymbol{\beta}, S} \left\| \boldsymbol{y}_S - X_S^T \boldsymbol{\beta} \right\|_2^2 \\ s.t. \quad S &\in \left\{ \Omega\big(Z\big) \;\middle|\; \forall i \leq m, \forall j \leq |Z^{(i)}| : |Z_j^{(i)}| \geq h(\boldsymbol{r}^{(i)}) \right\} \end{aligned} \tag{3.2}$$

We define $Z^{(i)}$ as the estimated uncorrupted set for the i^{th} 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 i^{th} mini-batch. Also, the residual vector $\boldsymbol{r}^{(i)} \in \mathbb{R}^n$ of the i^{th} mini-batch is defined as $\boldsymbol{r}^{(i)} = \boldsymbol{y}^{(i)} - \left[X^{(i)}\right]^T \boldsymbol{\beta}$. Specifically, we use the notation $\boldsymbol{r}_Z^{(i)}$ to represent the $|Z^{(i)}|$-dimensional residual vector containing the components in $Z^{(i)}$. The constraint of $Z^{(i)}$ is determined by function $h(\cdot)$, which is designed to estimate the size of the uncorrupted set of each mini-batch according to the residual vector $\boldsymbol{r}^{(i)}$. The uncorrupted set of each mini-batch will be consolidated by function $\Omega(\cdot)$ in both online and distributed approaches. The details of the heuristic function $h(\cdot)$ and consolidation function $\Omega(\cdot)$ will be explained in Section 3.4.

The problem defined above is very 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

Table 3.1: Math Notations

Notations	Explanations
$X^{(i)} \in \mathbb{R}^{p\times n}$	collection of data samples of the i^{th} mini-batch
$\boldsymbol{y}^{(i)} \in \mathbb{R}^{n\times 1}$	response vector of the i^{th} mini-batch
$\boldsymbol{\beta}^{(i)} \in \mathbb{R}^{p\times 1}$	estimated regression coefficient of the i^{th} batch
$\boldsymbol{\beta}_*^{(i)} \in \mathbb{R}^{p\times 1}$	ground truth regression coefficient of the i^{th} batch
$\boldsymbol{u}^{(i)} \in \mathbb{R}^{n\times 1}$	corruption vector of the i^{th} batch
$\boldsymbol{r}^{(i)} \in \mathbb{R}^{n\times 1}$	residual vector of the i^{th} batch
$\boldsymbol{\varepsilon}^{(i)} \in \mathbb{R}^{n\times 1}$	dense noise vector of the i^{th} batch
$Z^{(i)} \subseteq [n]$	estimated uncorrupted set of the i^{th} 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

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.

3.4 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 method (*HRR*) is proposed in Section 3.4.1. Based on *HRR*, a new approach, *DRLR*, is presented in Section 3.4.2 to process multiple mini-batches in distributed manner. Furthermore, in Section 3.4.3, a novel online version of *DRLR*, namely *ORLR*, is proposed to incrementally update the estimate of regression coefficients with new incoming data.

3.4.1 Single-Batch Heuristic Robust Regression

In order to efficiently solve the single batch problem when $m = 1$ in Equation (7.3), we propose a robust regression algorithm, *HRR*, based on heuristic hard thresholding. The algorithm heuristically determines the uncorrupted set $Z^{(i)}$ for the i^{th} 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

$$h(\boldsymbol{r}^{(i)}) := \underset{\tau\in\mathbb{Z}^+,\tau\leq n}{\arg\max}\ \tau \quad s.t.\ \ r^{(i)}_{\varphi(\tau)} \leq \frac{2\tau r^{(i)}_{\varphi(\tau_o)}}{\tau_o} \tag{3.3}$$

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 ϵ

Output: solution $\hat{\boldsymbol{\beta}}$, $\hat{Z}$

$Z_0 = [\mathrm{n}]$, $t \leftarrow 0$

repeat

$\boldsymbol{\beta}^{t+1} \leftarrow (X_{Z_t} X_{Z_t}^T)^{-1} X_{Z_t} \boldsymbol{y}_{Z_t}$

$\boldsymbol{r}^{t+1} \leftarrow |\boldsymbol{y} - X^T \boldsymbol{\beta}^{t+1}|$

$Z_{t+1} \leftarrow \mathcal{H}(\boldsymbol{r}^{t+1})$, where $\mathcal{H}(\cdot)$ is defined in Equation (7.6).

$t \leftarrow t + 1$

until $\|\boldsymbol{r}^{t+1}_{Z_{t+1}} - \boldsymbol{r}^{t}_{Z_t}\|_2 < \epsilon n$

return $\boldsymbol{\beta}^{t+1}$, Z_{t+1}

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

$$\tau_o = \underset{\lceil n/2\rceil \le \tau \le n}{\arg\min} \left| \left(r^{(i)}_{\varphi(\tau)}\right)^2 - \frac{\|\boldsymbol{r}^{(i)}_{Z_{\tau'}}\|_2^2}{\tau'} \right| \tag{3.4}$$

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 always have a residual higher in magnitude than that of uncorrupted data. Moreover, the constraint in Equation (7.4) ensures the residual of the largest element τ in our estimation cannot be too much 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 (7.5), 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 (7.4) chooses the maximum uncorrupted set size.

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

Definition 3 (**Heuristic Hard Thresholding**). *Defining $\varphi^{-1}_{\boldsymbol{r}}(i)$ as the position of the i^{th} element in residual vector $\boldsymbol{r}$'s ascending order of magnitude, the heuristic hard thresholding of $\boldsymbol{r}$ is defined as*

$$\mathcal{H}(\boldsymbol{r}) = \{i \in [n] : \varphi^{-1}_{\boldsymbol{r}}(i) \le h(\boldsymbol{r})\} \tag{3.5}$$

The optimization of $Z^{(i)}$ is formulated as solving Equation (7.6), 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. It then estimates the uncorrupted set Z_{t+1} 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.

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

$$d_{i,j} = \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^{(j)}\|_2 \tag{3.6}$$

Based on the definition of estimate distance, we define the distance vector of the i^{th} mini-batch as $\boldsymbol{d}^{(i)} \in \mathbb{R}^{m\times 1}$, where m is the total number of batches and $\boldsymbol{d}_j^{(i)}$ represents the distance from the estimate of the i^{th} batch to the j^{th} 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 k^{th} smallest value in distance vector $\boldsymbol{d}^{(i)}$, respectively. For instance, if the 3^{rd} batch is the 5^{th} smallest distance in $\boldsymbol{d}^{(i)}$ with $d_3^{(i)} = 0.3$, then we have $\sigma_5(\boldsymbol{d}^{(i)}) = 0.3$ and $\delta_5(\boldsymbol{d}^{(i)}) = 3$.

Definition 5 (**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 i^{th} batch, the p^{th} batch is defined as pivot batch if it satisfies*

$$p = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)}) \tag{3.7}$$

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 dominating set as follows.

Definition 6 (**Dominating 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 dominating set Ψ is defined as follows:*

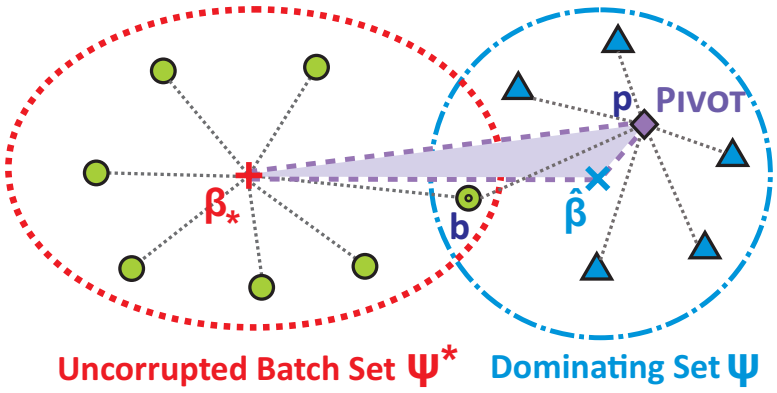


Figure 3.1: Example for Distributed Robust Least-squares Regression

$$\Psi = \left\{\delta_k(\boldsymbol{d}^{(p)}) | 1 \leq k \leq \tilde{m}\right\} \tag{3.8}$$

The dominating 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 $j \in \Psi$. The property will be used later in the proof of Lemma 5. Then we define the general robust consolidation of a set of regression coefficients as follows.

Definition 7 (**Robust Consolidation**). *Given a set of mini-batch estimates* $\{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ *and using* Ψ *to denote its dominating set, the robust consolidation of the given estimates* $\hat{\boldsymbol{\beta}}$ *is defined as follows:*

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{T} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2 \right\} \tag{3.9}$$

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 $\hat{\boldsymbol{\beta}}$

for *i = 1..m* **do**

  $\boldsymbol{\beta}^{(i)} \leftarrow \mathrm{HRR}(X^{(i)}, \boldsymbol{y}^{(i)})$

end

$p = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$ *// Optimize pivot batch p*

$\Psi = \left\{\delta_k(\boldsymbol{d}^{(p)}) | 1 \leq k \leq \tilde{m}\right\}$ *// Find dominating set Ψ*

$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{T} \sum_{i \in \Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2 \right\}$ *// Robust consolidation*

return $\hat{\boldsymbol{\beta}}$

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 dominating set Ψ in Line 4. Finally, all the batch estimates are combined via robust consolidation in Line 5. Figure 3.1 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 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 $d_{\hat{\boldsymbol{\beta}},p}$.

3.4.3 Online Robust Regression

The *DRLR* algorithm, proposed in Section 3.4.2, 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

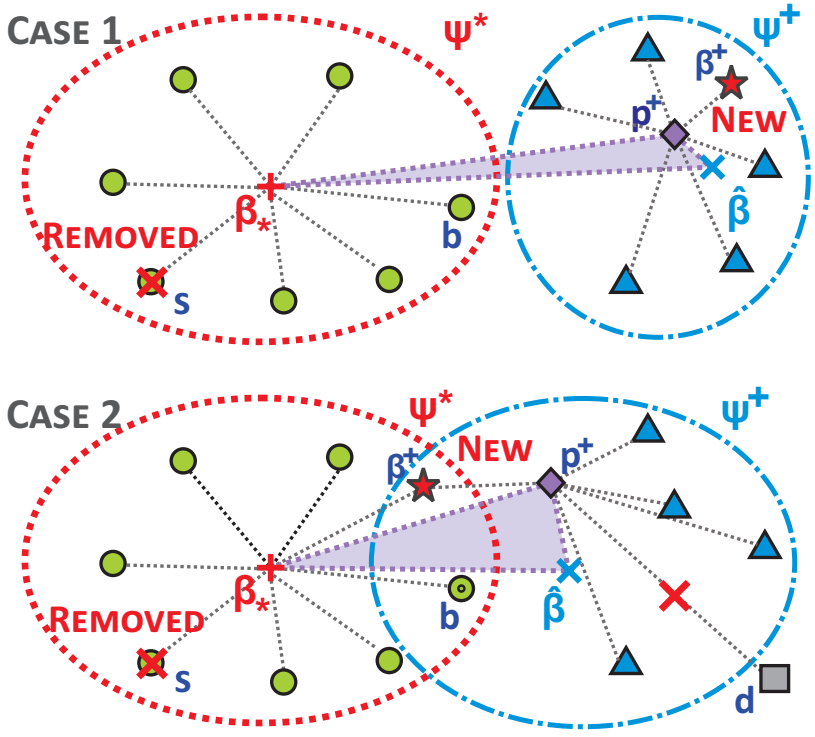


Figure 3.2: Examples for Online Robust Least-squares Regression

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 $\Pi = \{\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}\}$ and their corresponding Ψ.

Output: solution $\hat{\boldsymbol{\beta}}$, Π, Ψ

$\boldsymbol{\beta}^+ \leftarrow \mathrm{HRR}(X^+, \boldsymbol{y}^+)$

$s \leftarrow \min([m] \setminus \Psi)$ *// Select removed estimate s*

$\Pi^+ = \Pi \setminus \{\boldsymbol{\beta}^{(s)}\} \cup \{\boldsymbol{\beta}^+\}$

$p^+ = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$ *// Optimize new pivot batch p^+*

$\Psi^+ = \left\{\delta_k(\boldsymbol{d}^{(p^+)}) | 1 \le k \le \tilde{m}\right\}$ *// Find new dominating set Ψ^+*

$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{\frac{1}{\tilde{m}} \sum_{i\in\Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2\right\}$ *// Robust consolidation*

return $\hat{\boldsymbol{\beta}}, \Pi^+, \Psi^+$

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 dominating 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 3.2 shows two 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^+. Last but not least, the third case is $\boldsymbol{\beta}^+ \notin \{\Psi^* \cup \Psi^+\}$, which is not shown in Figure 3.2. The case is the same as Figure 3.1 except a new estimate is added outside of Ψ^* and Ψ. However, the change will not impact the result of $\hat{\boldsymbol{\beta}}$.

3.5 Theoretical Recovery Analysis

In this section, the recovery properties of regression coefficients for the proposed distributed and online algorithms are presented in Theorem 3 and 4, respectively. Before that, the recovery property of *HRR* is presented in Theorem 7.

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

$$\zeta_\gamma I \preceq \frac{1}{2}\nabla^2 f_S(\boldsymbol{\beta}) \preceq \kappa_\gamma I \quad \text{for } \forall S \in S_\gamma \tag{3.10}$$

Note that Equation (7.8) is equivalent to:

$$\zeta_\gamma \leq \min_{S\in S_\gamma} \lambda_{min}(X_S X_S^T) \leq \max_{S\in S_\gamma} \lambda_{max}(X_S X_S^T) \leq \kappa_\gamma \tag{3.11}$$

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

Theorem 2 (**HRR Recovery Property**). *Let* $X^{(i)} \in \mathbb{R}^{p\times n}$ *be the given data matrix of the* i^{th} *mini batch and the corrupted response vector* $\boldsymbol{y}^{(i)} = \left[\boldsymbol{X}^{(i)}\right]^T\boldsymbol{\beta}_* + \boldsymbol{u}^{(i)} + \boldsymbol{\varepsilon}^{(i)}$ *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* $t = \mathcal{O}\left(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}^{(i)}\|_2}{\sqrt{n\epsilon}}\right)$ *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 7 can be found in the supplementary material[1]. The theoretical analyses of regression coefficients recovery for Algorithm 2 and 3 are shown in the following.

Lemma 4. *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 i^{th} mini-batch. Then we have:

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

$$\sum_{i\in[m]\setminus\Psi_*}\gamma_i n \overset{(a)}{\leq}\sum_{i}^{m}\gamma_i n=\frac{\gamma_0}{2}\cdot m\cdot n$$
$$(\gamma_0 n+1)(m-|\Psi_*|)\overset{(b)}{\leq}\frac{\gamma_0}{2}\cdot m\cdot n$$

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

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

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

□

Lemma 5. *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* $\sigma_{\tilde{m}}(\boldsymbol{d}^{(p)})\leq 2\epsilon$*.*

Proof. Suppose k^{th} 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:

$$\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}^{(k)}\|_2-\|\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}_*\|_2\leq\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}_*\|_2\leq\epsilon$$
$$\|\boldsymbol{\beta}^{(i)}-\boldsymbol{\beta}^{(k)}\|_2\leq 2\epsilon$$

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 3 (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 1, we have $|\Psi_*|\geq\lfloor\frac{m}{2}\rfloor+1$. Because of Lemma 5, 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 $\Psi=\left\{\delta_k(\boldsymbol{d}^{(p)})|1\leq k\leq\tilde{m}\right\}$ 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 k^{th} 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.

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

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 4 (**ORLR Recovery Property**). *Given m mini-batch estimates of regression coefficients $\Pi=\{\boldsymbol{\beta}^{(1)},\ldots,\boldsymbol{\beta}^{(m)}\}$, their corresponding dominating 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

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

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 dominating 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

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

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.

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

Inequality *(f)* utilizes the fact that $\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}_*\|_2 \leq \epsilon$. Note that if $\tilde{m}$ is large enough, $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2 \precsim 5\epsilon$, which is the same as the error bound in Theorem 3.

□

3.6 Experiment

In this section, the proposed algorithms *DRLR* and *ORLR* are evaluated on both synthetic and real-world datasets. After the experiment setup has been introduced in Section 3.6.1, 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, in Section 3.6.2. 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 datasets used in the experiment can be downloaded here[2].

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

3.6.1 Experiment Setup

Datasets and Labels

Our dataset is composed of synthetic and real-world data. The simulation samples were randomly generated according to the model in Equation (7.1) for each mini-batch, sampling the regression coefficients $\boldsymbol{\beta}_* \in \mathbb{R}^p$ 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 $\boldsymbol{y}_*^{(i)} = \left[\boldsymbol{X}^{(i)}\right]^T \boldsymbol{\beta}_* + \boldsymbol{\varepsilon}^{(i)}$, 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 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 i^{th} 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.

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. In each dataset, there were 21 features after data preprocessing, including the number of beds and bathrooms, location, and average price in the area.

Evaluation Metrics

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

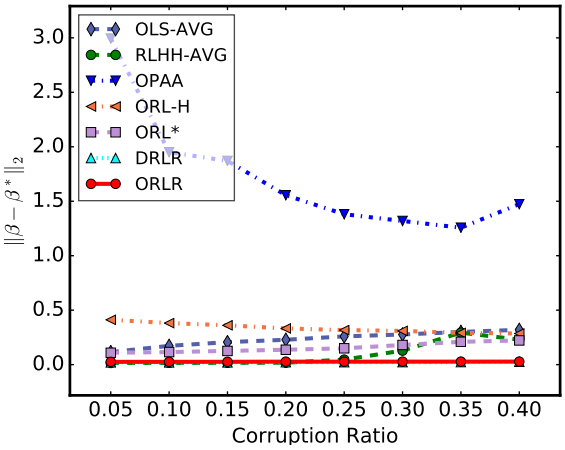
$$e = \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_*\|_2$$

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.

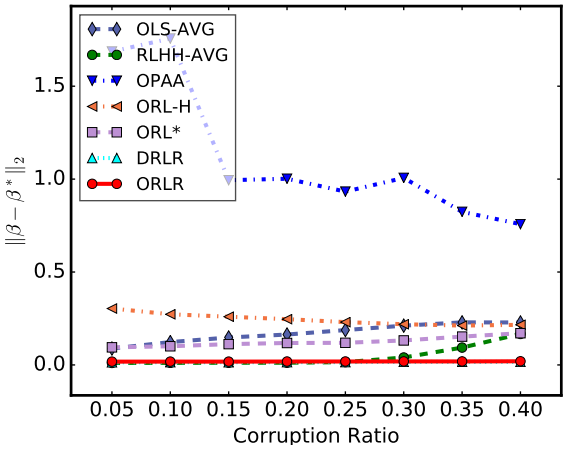
For the real-world dataset, we use the mean absolute error (MAE) to evaluate the performance of rental price prediction. Defining $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ as the predicted price and ground truth

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

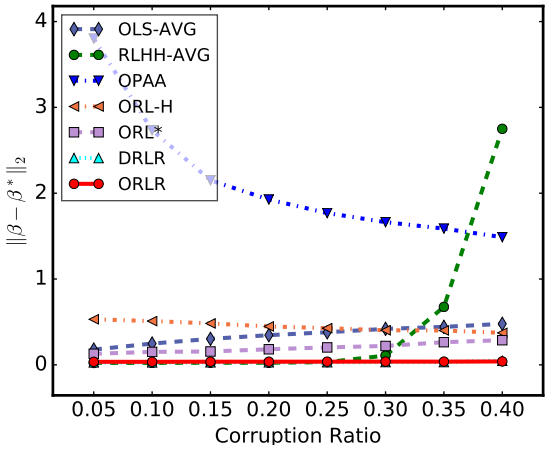
[4] `http://insideairbnb.com/get-the-data.html`



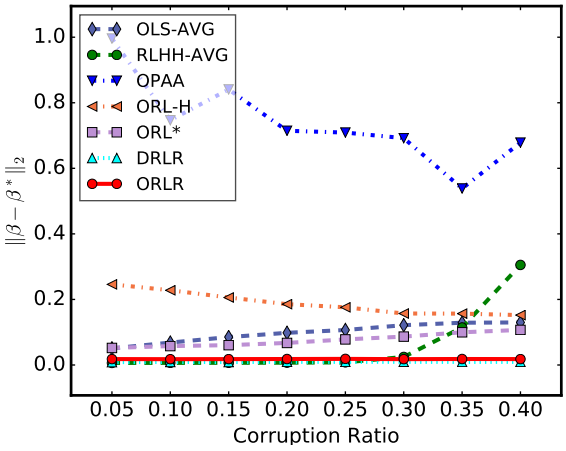
(a) p=100, n=5K, b=10, dense noise



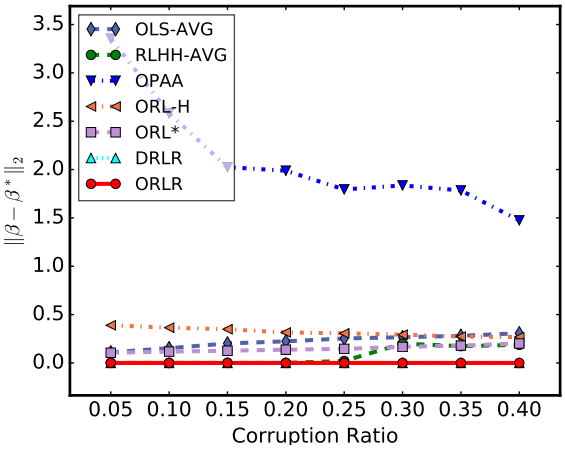
(b) p=100, n=10K, b=10, dense noise



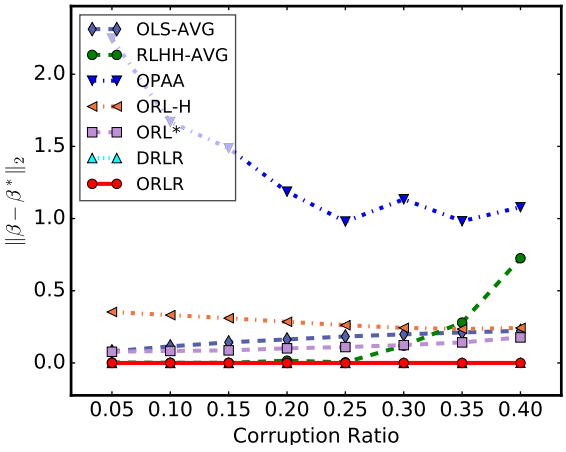
(c) p=400, n=10K, b=10, dense noise



(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

Figure 3.3: Performance on regression coefficients recovery for different corruption ratios in uniform distribution.

price, respectively, the mean absolute error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ can be presented as follows.

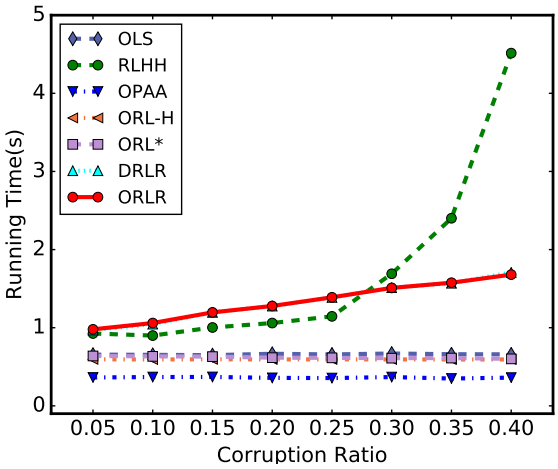
$$\text{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n} \left|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right|$$

Comparison Methods

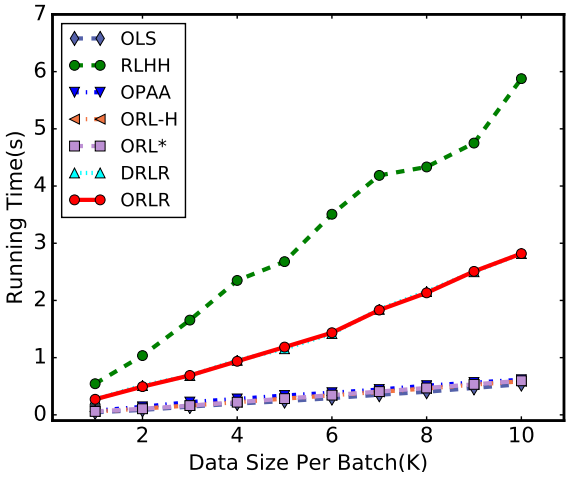
The following methods are included in the performance comparison presented here: The *averaged ordinary least-squares* (*OLS-AVG*) method takes the average over the regression coefficients of each mini-batch, which is computed by the ordinary least-squares method. *RLHH-AVG* applies a recently proposed robust method, *RLHH* [122], on each mini-batch and averages the regression coefficients of all the mini-batches. Different from *OLS-AVG*, *RLHH-AVG* can estimate the corrupted samples in each mini-batch by a heuristic method. The *online passive aggressive algorithm* (*OPAA*) [21] 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 *online robust learning* approach (*ORL*) [33], 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 [33] if it is unknown. For our proposed methods, we use *DRLR* and *ORLR* to evaluate our methods in both distributed and online settings. For *ORLR*, we set the number of previous mini-batch estimates to seven if not specified. All the results from comparison methods will be averaged over 10 runs.

Table 3.2: Performance on Regression Coefficients Recovery in Different Corrupted Mini-batches

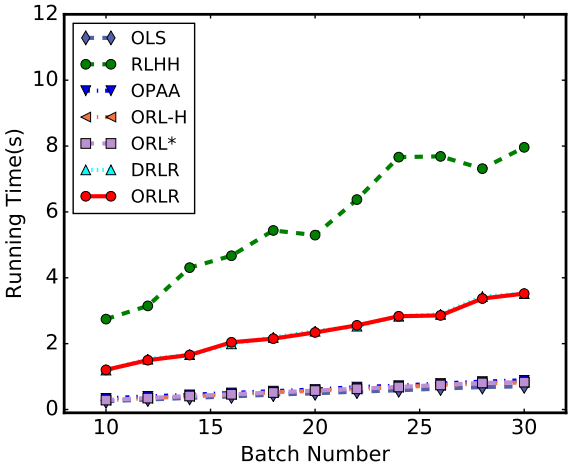
	0/20	**1/20**	**2/20**	**4/20**	**6/20**	**8/20**
OLS-AVG	0.126	0.133	0.147	0.169	0.193	0.208
RLHH-AVG	**0.011**	0.065	0.096	0.131	0.163	0.185
OPAA	1.537	1.577	1.385	1.573	1.539	1.483
ORL-H	0.346	0.362	0.358	0.392	0.417	0.442
ORL*	0.078	0.089	0.092	0.106	0.113	0.150
ORLR	0.025	**0.026**	**0.027**	**0.026**	**0.026**	**0.026**
DRLR	0.015	**0.015**	**0.015**	**0.015**	**0.015**	**0.015**



(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

Figure 3.4: Running time for different corruption ratios and data sizes

Table 3.3: Mean Absolute Error 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
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
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**

3.6.2 Performance

This section presents the recovery performance of the regression coefficients.

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 7.2 shows the performance of coefficients recovery for different corruption ratios in uniform distribution. Specifically, Figures 7.2(a) and 7.2(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* has 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* works well on mini-batches with fewer outliers, it cannot handle the case when the corrupted samples are arbitrarily distributed. 4) *OPAA* generally exhibits worse performance than the other algorithms because the incremental update for each data sample makes it very sensitive to outliers. Figures 7.2(c) and 7.2(d) show the similar performance when the number of features and batches increases. Figures 7.2(e) and 7.2(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 $\boldsymbol{\beta}_*$.

Performance on different corrupted mini-batches

Table B.1 shows the performance of regression coefficient recovery in different settings of corrupted mini-batches, ranging from zero to eight 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 B.1, 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.

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 i^{th} sample data. Table 3.3 shows the mean absolute error 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.

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 7.4 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. In addition, 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*.

3.7 Conclusion

In this paper, distributed and online robust regression algorithms, *DRLR* and *ORLR*, are proposed to handle the scalable least squares regression problem in the presence of adversarial corruption. 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.

Chapter 4

Robust Regression via Online Feature Selection with Adversarial Noises

This chapter presents a novel online feature selection algorithm when the dataset contains adversarial data corruption. The introduction of the work is provided in Section 4.1. Then, Section 4.2 reviews the related work in robust regression model and online feature selection categories. Section 4.3 gives a formal problem formulation. The proposed *RoOFS* algorithm is presented in Section 4.4. Section 4.5 presents the theoretical analysis of proposed algorithm. In Section 4.6, the experimental results are analyzed and the paper concludes with a summary of our work in Section 4.7.

4.1 Introduction

The presence of noise and data corruption in real-world data can be inevitably caused by various reasons such as experimental errors, accidental outliers, or even adversarial data attacks. In traditional robust regression problem, reliable regression coefficients are learned in the presence of adversarial data corruptions in its response vector. A commonly adopted model from existing methods assumes that the observed response is obtained from the generative model $\boldsymbol{y} = X^T\boldsymbol{\beta}^* + \boldsymbol{u}$, where $\boldsymbol{\beta}^*$ is the true regression coefficients we wish to recover and $\boldsymbol{u}$ is the corruption vector with adversarial values. In the problem setting, the data matrix X is assumed to contain all the features that can be accessed at any time and by arbitrarily many times.

Existing robust learning methods typically focus on modeling the entire dataset with all the features at once; however, they may meet the bottleneck in terms of computation and memory as more and more data sets are becoming However, the assumption is no longer suitable to the following scenarios in the applications that contain exponentially increasing user-generated contents: 1) *features are too many to be loaded entirely.* Features grows

dramatically fast and becomes extremely large in. For instance, over 4.7 million movies and televisions with 8.3 million reviews in IMDb[1] online movie and television review website, which makes it hard to load all the features entirely for any machine learning models using the movies as features. 2) *features are generated dynamically.* For example, people create and use new terms and hashtags all the time in Twitter, and the "Likes" [80] on newly-generated articles in Facebook can be considered as new features describing the interestingness of the user. 3) Therefore, it is necessary to address online features in traditional robust regression as a new fundamental problem; however, current methods either focus on robust regression or online feature learning separately.

To the best of our knowledge, our proposed approach is the first robust regression algorithm that can handle the online features with adversarial data corruptions. It is nontrivial to consider online features and adversarial data corruption simultaneously in robust regression because 1) robust methods usually estimate data corruption based on the entire data, but online features make the data can only be partially accessible at one time; and 2) online feature selection methods can only select features based on uncorrupted data. Simply using robust regression and online feature selection methods sequentially makes the recovery result of coefficients worse, which is presented in our experiments in Section 4.6. To address the above challenges, we proposed a new robust regression algorithm via online feature selection (*RoOFS*). The main contributions of our study are summarized as follows:

- *design of an efficient algorithm to simultaneously address the problem of data corruption and online feature.* The algorithm *RoOFS* is proposed to recover the regression coefficients and uncorrupted set efficiently. Unlike using entire features, our approach alternately estimates the data corruption and selects the feature set via a robust online feature substitution method.
- *theoretical analysis of the algorithm.* We prove that our method yields a solution with a restricted error bound compared to ground truth coefficients under the Subset Restricted Strong Convexity (*SRSC*) property.
- *demonstration of empirical effectiveness and efficiency.* Our proposed algorithm was evaluated with 6 competing methods in both robust regression and online feature selection literatures. The results showed that our approach consistently outperforms existing methods in coefficients recovery and uncorrupted set estimation, delivering a competitive running time.

4.2 Related Work

The work related to this paper is summarized in the categories of robust regression model and online feature selection as below.

[1]https://www.imdb.com/

4.2.1 Robust Regression Model

A large body of literature on robust regression problem has been established over the last few decades. Most of studies focus on handling stochastic noise in small amounts [62]; however, these methods cannot be applied to data that may exhibit malicious corruption [19]. To recover regression coefficients with adversarial data corruption, Chen et al. [19] proposed a robust algorithm based on trimmed inner product. McWilliams et al. [72] proposed a sub-sampling algorithm for large-scale corrupted linear regression, but their theoretical recovery boundaries are not close to the ground truth [5]. Some L_1 penalty based methods [82, 109] pursue strong recovery results for robust regression problem, but these methods depend on severe restrictions of the data distribution such as row-sampling from an incoherent orthogonal matrix [82]. Zhang et al. [120] proposed a distributed robust algorithm to handle the large-scale data set under adversarial data corruption.

Most research in this area requires the corruption ratio parameter, which is difficult to estimate under the assumption that the dataset can be adversarially attacked. For instance, She and Owen [99] rely on a regularization parameter to determine the size of the uncorrupted set based on soft-thresholding. Chen et al. [19] require the upper bound of the outliers number, which is also difficult to estimate when the data contain the adversarial data corruption. Bhatia et al. [5] proposed a hard-thresholding algorithm with a strong guarantee of coefficient recovery under mild assumption on input data. However, the corruption ratio parameter is required by the algorithm and its recovery error can be more than doubled in size if the parameter is far from the true value. Recently, Zhang et al. [123] proposed a heuristic hard-thresholding based methods that learns the optimal uncorrupted set. However, all these approaches are based on batch feature selection under the assumption that all features can be accessed entirely at any time, which is infeasible to apply in massive and fast growing feature set.

4.2.2 Online Feature Selection

Online feature selection methods [46, 108, 116] relaxes the requirement of batch selection and fit the scenarios that feature cannot be accessed entirely at one time. Statistical online feature selection algorithms [111,115,128] select features via certain statistical quantity such as mutual information, but these methods lack of specific objectives and usually have sub-optimal solutions for some certain tasks. Optimization based approaches [86,130] use target oriented objective functions solved by some specific optimization techniques. These methods usually require the regression coefficient $\boldsymbol{\beta}$ be sparse, i.e., $\|\boldsymbol{\beta}\|_0 \leq \mu$. Grafting [87] and its variation [130] relax the hard constraint of feature set into L_1 penalty, which makes it a convex problem. However, the parameter of L_1 norm [95] is difficult to determine because the usual cross validation strategy is unavailable for the online feature selection scenario [107]. Yang et al. [114] proposed a limited-memory substitution algorithm based on the L_0 norm constraint. Although the hard constraint leads to an NP-hard problem, a

Table 4.1: Math Notations

Notations	Explanations
$p, n \in \mathbb{R}$	number of entire features and data samples
$p_t \in \mathbb{R}$	number of features in t^{th} time interval
$\mu \in \mathbb{R}$	ratio of feature sparsity, where $\|\boldsymbol{\beta}\|_0 = \mu$
$X_t \in \mathbb{R}^{p_t \times n}$	data samples containing features in the t^{th} time interval
$X \in \mathbb{R}^{p \times n}$	data samples containing the entire features
$\boldsymbol{\beta}, \boldsymbol{\beta}^* \in \mathbb{R}^{p \times 1}$	estimated and ground truth regression coefficient
$\boldsymbol{u} \in \mathbb{R}^{n \times 1}$	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 $\boldsymbol{r} = \|\boldsymbol{y} - X^T\boldsymbol{\beta}\|$
$S \subseteq [n]$	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 set

theoretical guarantee for the error bound of their local optimal solution is provided. However, none of these online feature methods can handle the adversarial data corruption.

4.3 Problem Formulation

In this study, we consider the problem of robust regression with adversarial data corruption in the feature selection scenario in which only a few features are accessible at each time. Given data matrix $X_t \in \mathbb{R}^{p_t \times n}$ where p_t is the number of features available in the t^{th} time interval, and n are the number of data samples. The data matrix for all the time intervals is represented as $X = \{X_t\}_{t=1}^{\mathcal{T}}$. We assume the corresponding response vector $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$ is generated using the following model:

$$\boldsymbol{y} = X^T\boldsymbol{\beta}^* + \boldsymbol{u} + \boldsymbol{\varepsilon} \tag{4.1}$$

where $\boldsymbol{\beta}^*$ represents the μ-sparse ground truth coefficients of the regression model i.e., $\|\boldsymbol{\beta}^*\|_0 \leq \mu$ and $\boldsymbol{u}$ is the unbounded corruption vector introduced by adversarial data attacks. $\boldsymbol{\varepsilon} \in \mathbb{R}^{n \times 1}$ represents the additive dense noise, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Different from the corruption vector $\boldsymbol{u}$ that can be arbitrarily distributed, the dense noise ε_i follows normal distribution with zero mean and a relatively small variance σ. The notations used in this paper is summarized in Table 7.2.

The goal of our problem is to learn a new robust regression problem with online feature selection, which is to recover the regression coefficients $\boldsymbol{\beta}^*$ and simultaneously determine the uncorrupted point set $\hat{S}$ with sequentially accessible features. The problem is formally defined as follows:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \arg\min_{\boldsymbol{\beta}, S} \|\boldsymbol{y}_S - X_S^T \boldsymbol{\beta}\|_2^2 \\ s.t.\ \ & S \subset [n],\ |S| \geq \mathcal{G}(\boldsymbol{\beta}),\ \|\boldsymbol{\beta}\|_0 \leq \mu \end{aligned} \tag{4.2}$$

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 ground truth set of uncorrupted points. 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 notation $\Psi = supp(\boldsymbol{\beta})$ is used to represent the set of selected features, resulting in $|\Psi| \leq \mu$. Similarly, we use X_Ψ to signify the rows of X are restricted to indices in Ψ and $X_{\Psi,S}$ to restrict both the rows and columns in set Ψ and S. The function $\mathcal{G}(\cdot)$ determines the size of uncorrupted data according to the regression coefficients $\boldsymbol{\beta}$, which is explained in Section 4.4. It is worth mentioning that the features of data matrix X in Equation (7.3) cannot be loaded entirely, but they can be accessed partially for each time interval. Therefore, the joint optimization of $\boldsymbol{\beta}$ and S in our problem are very challenging because it amounts to a non-convex discrete optimization problem under the assumption that data matrix X cannot be access entirely at one time.

4.4 The Proposed Methodology

To solve the problem in Equation (7.3) efficiently with the guarantee on the strong recovery of regression coefficients, we propose a novel robust regression algorithm with online feature selection, *RoOFS*. The algorithm is only allowed to access part of features at each time, which are defined as the newly incoming features in our problem. One naive solution to handle the sequentially incoming features is to retain all the features in the memory and then apply traditional robust feature selection methods. However, the solution has two major drawbacks: 1) the feature set can be too large to be retained in the memory, and 2) the algorithm becomes slower and slower when the feature set increases. Therefore, we proposed a new "robust online substitution" method to decide the retained feature set based on an adaptively estimated corrupted set. The procedure of robust online substitution is defined as follows:

- Update coefficients of retained features Ψ based on the estimated uncorrupted set S as follows: $\boldsymbol{\beta}_\Psi := \boldsymbol{\beta}_\Psi - \eta\ X_{\Psi,S}^T(X_{\Psi,S}^T\boldsymbol{\beta}_\Psi - \boldsymbol{y}_S)$, where η is the step length.
- Retain the top μ largest (in magnitude) elements in $\boldsymbol{\beta}$ and set the rest to zero. Then all the non-zero features will be kept in the retained feature set Ψ.
- Compute the residual vector $\boldsymbol{r} \in \mathbb{R}^{n\times 1}$ with the updated coefficients $\boldsymbol{\beta}$, then estimate the uncorrupted feature set S via a thresholding operator $\mathcal{H}_\tau(\boldsymbol{r})$, where τ is the estimated size of uncorrupted set.

The procedure will be repeatedly executed until the residual vector $\boldsymbol{r}$ converges. The thresholding operator $\mathcal{H}_\tau(\cdot)$ is formally defined as follows:

Definition 9 (**Thresholding Operator**). *Defining $\varphi_{\boldsymbol{v}}^{-1}(i)$ as the position of the i^{th} element in input vector $\boldsymbol{v}$'s ascending order of magnitude and τ as the threshold parameter, the thresholding operator of $\boldsymbol{v}$ is defined as*

$$\mathcal{H}_\tau(\boldsymbol{v}) = \{i \in [n] : \varphi_{\boldsymbol{v}}^{-1}(i) \leq \tau\} \tag{4.3}$$

To estimate the uncorrupted set S, the thresholding operator $\mathcal{H}_\tau(\cdot)$ generally requires two inputs: residual vector $\boldsymbol{r}$ and the size of uncorrupted set τ. The residual vector $\boldsymbol{r}$ can be computed with coefficients $\boldsymbol{\beta}$ as follows:

$$\boldsymbol{r} = |\boldsymbol{y} - X^T\boldsymbol{\beta}| \tag{4.4}$$

For the size of uncorrupted set, two general cases are discussed. The first case is that the size can be estimated by users based on their prior knowledge on the data. For instance, if we know the data corruption happens rarely, then we can estimate the uncorrupted size as 95% of the entire data. However, it is hard to obtain prior knowledge on the data in the real-world. Thus, in the second case, we propose a method to adaptively estimate the uncorrupted size based on the residual vector $\boldsymbol{r}$. The method follows an intuition that when the coefficient $\boldsymbol{\beta}$ is close to $\boldsymbol{\beta}^*$, the residuals of uncorrupted samples are smaller than those of corrupted samples in strong possiblity. The intuition can be explained by the generative model in Equation (7.1), 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 $\boldsymbol{\varepsilon}$.

The estimation of uncorrupted size can be formalized to solve the following problem:

$$\hat{\tau} := \underset{\lceil n/2 \rceil < \tau \leq n}{\arg\max}\ \tau \quad s.t.\ \ r_{\varphi(\tau)} \leq \frac{2\tau r_{\varphi(\tau_o)}}{\tau_o}, \tau \in \mathbb{Z}^+ \tag{4.5}$$

where $r_{\varphi(k)}$ represents the k^{th} 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 $r^2_{\varphi(\tau_o)}$ has the closest value to $\frac{\|\boldsymbol{r}_{\mathcal{H}_{\tau'}(\boldsymbol{r})}\|_2^2}{\tau'}$, where $\tau' = \tau - \lceil n/2 \rceil$ and $\mathcal{H}_{\tau'}(\boldsymbol{r})$ represent the position set containing the smallest τ' elements in residual $\boldsymbol{r}$. The problem in Equation (7.4) can be solved by searching from n to $\lceil n/2 \rceil + 1$ and return the first value $\hat{\tau}$ which satisfies the constraint. It is important to note that the estimation method in Equation (7.4) requires the coefficients $\boldsymbol{\beta}$ to be close to $\boldsymbol{\beta}^*$. Thus, we optimize the uncorrupted set S along with coefficient $\boldsymbol{\beta}$ until both of them converge.

The details of *RoOFS* algorithm are presented in Algorithm 4. In Line 3, the algorithm receives data matrix X_{Ψ^k} with the incoming feature set Ψ^k at time k. The new feature set Ψ^k is combined into the retained feature set Ψ in Line 4. For each incoming feature set, the algorithm iteratively optimizes the regression coefficients $\boldsymbol{\beta}$ and the uncorrupted

ALGORITHM 4: RoOFS Algorithm

Input: Corrupted training data $\{\boldsymbol{x}_i, y_i\}$, $i = 1...\text{n}$, feature ratio μ, tolerance ϵ
Output: solution $\hat{\boldsymbol{\beta}}$
$\boldsymbol{\beta}^0 \leftarrow \mathbf{0}$, $\Psi = \emptyset$, $S_0 = [\text{n}]$, $t \leftarrow 0$, $k \leftarrow 0$
repeat
  Receive features X_{Ψ^k} from the pool $\bar{\Psi}$ with index set Ψ^k
  $\Psi = \Psi \cup \Psi^k$
  repeat
    $\boldsymbol{\beta}_{\Psi}^{t+1} \leftarrow \boldsymbol{\beta}_{\Psi}^{t} - \eta X_{\Psi,S_t}^T (X_{\Psi,S_t}^T \boldsymbol{\beta}_{\Psi}^{t} - \boldsymbol{y}_{S_t})$
    if $|\Psi| > \mu$ **then**
      $\Omega = \arg\min_{\Omega \in \Psi} \|\boldsymbol{\beta}_{\Omega}\|_1 \quad s.t.\ |\Omega| = |\Psi| - \mu$
      $\boldsymbol{\beta}_{\Omega} = 0$
      $\Psi = \Psi \setminus \Omega$
    end
    $\boldsymbol{r} = |\boldsymbol{y} - X^T \boldsymbol{\beta}|$
    $S_{t+1} \leftarrow \mathcal{H}_{\tau}(\boldsymbol{r}^{t+1})$, where τ is the estimated uncorrupted size.
    $t \leftarrow t + 1$
  until $\|\boldsymbol{r}_{S_{t+1}}^{t+1} - \boldsymbol{r}_{S_t}^{t}\|_2 < \epsilon n$
  $k \leftarrow k + 1$
until *No more features;*
return $\boldsymbol{\beta}^{t+1}$, S_{t+1}

set S until the value of residual vector $\boldsymbol{r}_{S_t}^t$ is converged in Line 14. Specifically, in Line 6, regression coefficients $\boldsymbol{\beta}$ are updated to a better fit for the current estimated feature set Ψ and uncorrupted set S_t. In Line 8, feature set Ω that contains features with $|\Psi| - \mu$ smallest weights in $\boldsymbol{\beta}$ is selected. Then features in Ω are removed from the retained feature set Ψ and the weights in $\boldsymbol{\beta}_{\Omega}$ are reset to zero in Lines 9 and 10. The residual vector $\boldsymbol{r}$ is updated in Line 11, while the uncorrupted set S_{t+1} is estimated in Line 12 by the thresholding operator. Finally, both coefficients $\boldsymbol{\beta}$ and uncorrupted set S are returned in Line 17.

4.5 Theoretical Analysis

In this section, we show that the local optimal solution of our algorithm obtains a restricted error bound compared to ground truth solution. To prove the theoretical properties of our algorithm, we require that the least squares function satisfies the *Subset Restricted Strong Convexity* (SRSC), which is defined as follows:

Definition 10 (**SRSC Property**). *The least squares function* $f_S(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T \boldsymbol{\beta}\|_2^2$ *satisfies Subset Restricted Strong Convexity (SRSC) Property if the following holds for* $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in$

Ω_μ *and* $\forall S \in \mathcal{S}_\gamma$*:*

$$f_S(\boldsymbol{\beta}_1) - f_S(\boldsymbol{\beta}_2) \geq \nabla^T f_S(\boldsymbol{\beta}_2)(\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2) + \frac{\varphi_\mu}{2}\|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2\|_2^2 \tag{4.6}$$

To provide the local optimality property of our solution, the following two lemmas are first proved.

Lemma 6. *For a given least squares function* $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - X^T\boldsymbol{\beta}\|_2^2$*, let residual vector* $\boldsymbol{r} = \boldsymbol{y} - X^T\boldsymbol{\beta}$ *and* $\delta(k)$ *be the k-th position of the ascending order in vector* $\boldsymbol{r}$*, i.e.* $r_{\delta(1)} \leq r_{\delta(2)} \leq ... \leq r_{\delta(n)}$*. For any* $1 \leq \tau_1 < \tau_2 \leq n$ *and* $\forall \boldsymbol{\beta}^t \in \Omega_m$*, let* $S_1 = \{\delta(i)|1 \leq i \leq \tau_1\}$ *and* $S_2 = \{\delta(i)|1 \leq i \leq \tau_2\}$*. We then have* $f_{S_1}(\boldsymbol{\beta}^t) \leq f_{S_2}(\boldsymbol{\beta}^t)$*.*

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

Lemma 7. *Let* $\tau_* = \gamma n$ *be the true number of uncorrupted samples and* τ_t *be the estimated uncorrupted threshold at the t-th iteration. If* $\tau_t \leq \tau_*$*, then* $f_{S_t}(\boldsymbol{\beta}^t) \leq f_{S_*}(\boldsymbol{\beta}^t)$*. If* $\tau_t > \tau_*$*, then* $f_{S_t}(\boldsymbol{\beta}^t) \leq \lambda f_{S_*}(\boldsymbol{\beta}^t)$*, where* $\lambda = \left[1 + \frac{128(1-\gamma)}{2\gamma-1}\right]$*.*

Proof. To simplify the notation, the subscripts t that signify the t-th iteration will be omitted and the residual vector $\boldsymbol{r}$ is assumed to be sorted in ascending order of magnitude.

We will discuss the τ_t value in two different conditions compared to the value of τ_*. In the first condition that $\tau_t \leq \tau_*$, let $S_t = \{\delta(i)|1 \leq i \leq \tau_t\}$ and $S_* = \{\delta(i)|1 \leq i \leq \tau_*\}$, we have $f_{S_t}(\boldsymbol{\beta}^t) \leq f_{S_*}(\boldsymbol{\beta}^t)$ according to Lemma 6. When $\tau_t > \tau_*$, we have the following properties according to the constraint specified in equation (7.4).

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

The inequality *(a)* follows the definition of τ_o and the fact that $|S_* \cap S_t| \geq \tau'$. The inequality *(b)* follows $|S_t \setminus S_*| \leq (1-\gamma) \cdot n$ and $\|\boldsymbol{r}_{S_t \setminus S_*}\|_2^2 \leq |S_t \setminus S_*| r_\tau^2$. Then we have

$$\begin{aligned} f_{S_t \setminus S_*}(\boldsymbol{\beta}) &\leq \left[64(1-\gamma) \cdot \frac{n}{\tau'} + 1\right] f_{S_* \setminus S_t}(\boldsymbol{\beta}) \\ &\quad + \left[64(1-\gamma) \cdot \frac{n}{\tau'}\right] f_{S_* \cap S_t}(\boldsymbol{\beta}) \\ f_{S_t \setminus S_*}(\boldsymbol{\beta}) + f_{S_* \cap S_t}(\boldsymbol{\beta}) &\overset{(c)}{\leq} \left[64(1-\gamma) \cdot \frac{n}{\tau'} + 1\right] f_{S_*}(\boldsymbol{\beta}) \\ f_{S_t}(\boldsymbol{\beta}) &\overset{(d)}{\leq} \left[1 + \frac{128(1-\gamma)}{2\gamma - 1}\right] f_{S_*}(\boldsymbol{\beta}) \end{aligned}$$

The inequality *(c)* follows $f_{S_*}(\boldsymbol{\beta}) = f_{S_*\setminus S_t}(\boldsymbol{\beta}) + f_{S_*\cap S_t}(\boldsymbol{\beta})$ and the inequality *(d)* follows $\tau' = \tau_t - \frac{n}{2}$. □

Theorem 5. *Assume that least squares function $f_S(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\|_2^2$ satisfies Subset Restricted Strong Convexity (SRSC) Property for $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \Omega_\mu$ and $\forall S \in \mathcal{S}_\gamma$, then we have*

$$f_{\hat{S}}(\hat{\boldsymbol{\beta}}) - f_{S^*}(\boldsymbol{\beta}^*) \leq \frac{\alpha\lambda}{1+\alpha} f_{S^*}(\mathbf{0}) + \left(\frac{\lambda}{1+\alpha} - 1\right) f_{S^*}(\boldsymbol{\beta}^*) \tag{4.7}$$

where $\alpha = \left(\frac{1}{\eta\cdot\varphi_\mu}\right)^2$ and $\lambda = \left[1 + \frac{128(1-\gamma)}{2\gamma-1}\right]$. Specifically, when the uncorrupted set size is less than ground truth, $\lambda = 1$.

Proof. As function $f_S(\boldsymbol{\beta})$ satisfies SRSC property, we have

$$f_S(\boldsymbol{\beta}_1) - f_S(\boldsymbol{\beta}_2) \geq \langle \nabla f_S(\boldsymbol{\beta}_2), \boldsymbol{\beta}_1 - \boldsymbol{\beta}_2 \rangle + \frac{\varphi_\mu}{2}\|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2\|_2^2 \tag{4.8}$$

for $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \Omega_\mu$ and $\forall S \in \mathcal{S}_\gamma$. Let $\text{supp}(\boldsymbol{\beta}_1) = \Omega_1$, then we have

$$\begin{aligned}
&f_S(\boldsymbol{\beta}_1) - f_S(\boldsymbol{\beta}_2) \\
&\quad \geq \min_{\text{supp}(\boldsymbol{\beta})\subseteq\Omega_1} \left\{ \langle \nabla f_S(\boldsymbol{\beta}_2), \boldsymbol{\beta} - \boldsymbol{\beta}_2 \rangle + \frac{\varphi_\mu}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}_2\|_2^2 \right\} \\
&\quad \overset{(a)}{=} -\frac{1}{2\varphi_\mu} \left\| \left[\nabla f_S(\boldsymbol{\beta}_2)\right]_{\Omega_1} \right\|_2^2 \overset{(b)}{\geq} -\frac{|\Omega_1|}{2\varphi_\mu} \left\| \left[\nabla f_S(\boldsymbol{\beta}_2)\right]_{\Omega_1} \right\|_\infty^2 \\
&\quad \geq -\frac{\mu}{2\varphi_\mu} \left\| \nabla f_S(\boldsymbol{\beta}_2) \right\|_\infty^2
\end{aligned} \tag{4.9}$$

The equation (a) solves the minimum value of $\min_{\text{supp}(\boldsymbol{\beta})\subseteq\Omega_1}\{\cdot\}$ by setting its gradient to zero, and the inequality (b) follows $|\Omega_1| \leq \mu$. Let $\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_2$ be ground truth coefficient $\boldsymbol{\beta}^*$ and estimated solution $\hat{\boldsymbol{\beta}}$ respectively, and set S be ground truth uncorrupted set S^*, then we have

$$f_{S^*}(\hat{\boldsymbol{\beta}}) - f_{S^*}(\boldsymbol{\beta}^*) \leq \frac{\mu}{2\varphi_\mu} \left\| \nabla f_{S^*}(\hat{\boldsymbol{\beta}}) \right\|_\infty^2 \leq \frac{\mu}{2\varphi_\mu} \left(\frac{1}{\eta}\hat{\boldsymbol{\beta}}_{\min}\right)^2$$

where $\hat{\boldsymbol{\beta}}_{\min} = \min_i |\hat{\boldsymbol{\beta}}_i|$. According to SRSC property, we have

$$f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}}) \geq \langle \nabla f_{S^*}(\hat{\boldsymbol{\beta}}), -\hat{\boldsymbol{\beta}} \rangle + \frac{\varphi_\mu}{2}\|\hat{\boldsymbol{\beta}}\|_2^2$$

Because of $\langle \nabla f_{S^*}(\hat{\boldsymbol{\beta}}), -\hat{\boldsymbol{\beta}} \rangle \geq 0$, we have

$$\mu\hat{\boldsymbol{\beta}}^2_{\min} \leq \|\hat{\boldsymbol{\beta}}\|_2^2 \leq \frac{2}{\varphi_\mu}\Big[f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}})\Big]$$
$$2\varphi_\mu \cdot \eta^2\Big[f_{S^*}(\hat{\boldsymbol{\beta}}) - f_{S^*}(\boldsymbol{\beta}^*)\Big] \overset{(c)}{\leq} \frac{2}{\varphi_\mu}\Big[f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}})\Big]$$

The inequality (c) follows the Equation (4.9). Let α be $\left(\frac{1}{\eta\varphi_\mu}\right)^2$, then we have

$$f_{S^*}(\hat{\boldsymbol{\beta}}) - f_{S^*}(\boldsymbol{\beta}^*) \leq \alpha\Big[f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}})\Big]$$
$$f_{S^*}(\hat{\boldsymbol{\beta}}) \leq \frac{\alpha}{1+\alpha}\Big[f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}})\Big] + f_{S^*}(\boldsymbol{\beta}^*)$$

According to Lemma 2, we have

$$\frac{1}{\lambda}f_{\hat{S}}(\hat{\boldsymbol{\beta}}) \leq f_{S^*}(\boldsymbol{\beta}^*) \leq \frac{\alpha}{1+\alpha}\Big[f_{S^*}(\mathbf{0}) - f_{S^*}(\hat{\boldsymbol{\beta}})\Big] + f_{S^*}(\boldsymbol{\beta}^*)$$
$$f_{\hat{S}}(\hat{\boldsymbol{\beta}}) - f_{S^*}(\boldsymbol{\beta}^*) \leq \frac{\alpha\lambda}{1+\alpha}f_{S^*}(\mathbf{0}) + \left(\frac{\lambda}{1+\alpha} - 1\right)f_{S^*}(\boldsymbol{\beta}^*)$$

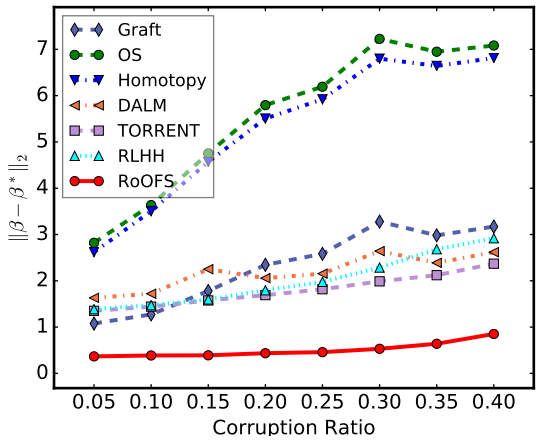
□

Since the value of $f_{S^*}(\mathbf{0})$ and $f_{S^*}(\boldsymbol{\beta}^*)$ are both constants and $f_{S^*}(\boldsymbol{\beta}^*)$ is close to 0, the error bound of our solution is depended on the value of $\frac{\alpha\lambda}{1+\alpha}$. When the ratio of data corruption γ is close to one, λ is close to one according to its definition. In addition, the value of α is smaller when the parameter φ_μ of the SRSC property is smaller. Therefore, the error of our solution can be close to zero when both φ_μ and γ are large enough.

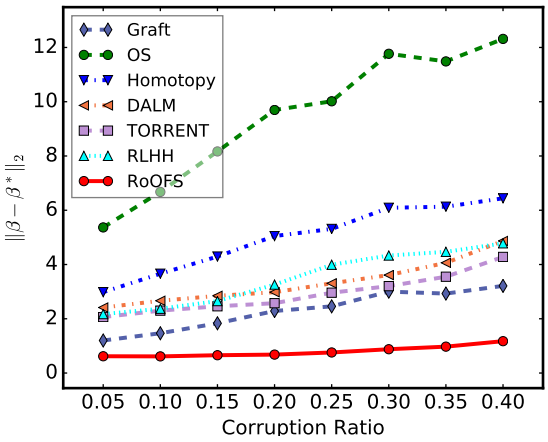
4.6 Experimental Results

In this section, we report the extensive experimental evaluation performed to verify the robustness, effectiveness of feature selection, and efficiency of the proposed method. All the experiments were conducted on a 64-bit machine with 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 experiment can be downloaded here[2].

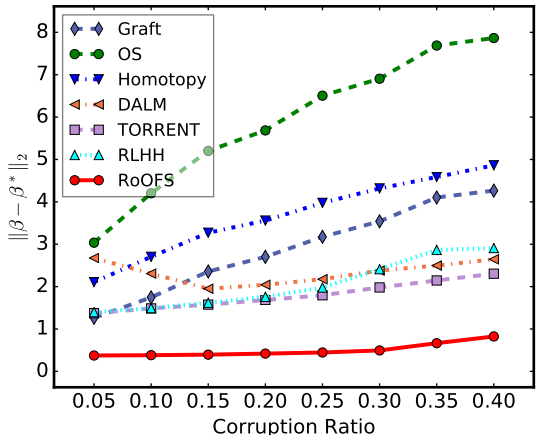
[2] `https://goo.gl/C4HQjo`



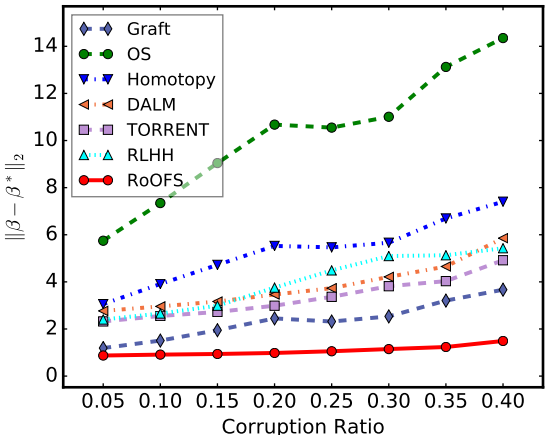
(a) p=2K, n=1K, μ/p=20%, dense



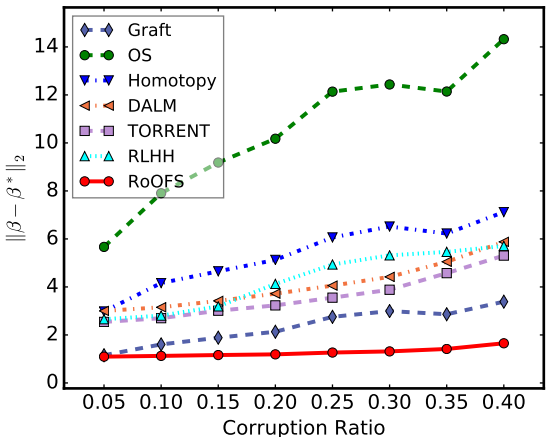
(b) p=4K, n=1K, μ/p=20%, dense



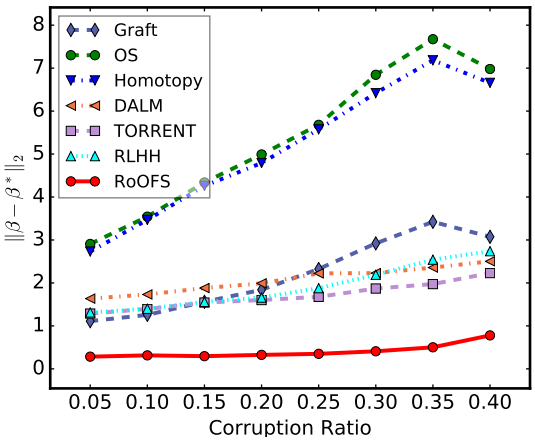
(c) p=4K, n=2K, μ/p=20%, dense



(d) p=4K, n=1K, μ/p=40%, dense



(e) p=4K, n=1K, μ/p=80%, dense



(f) p=2K, n=1K, μ/p=20%, no dense

Figure 4.1: Performance on regression coefficients recovery for different corruption ratios in uniform distribution.

Table 4.2: F1 Scores for the Performance on Uncorrupted Set Recovery.

	p=2K, n=1K, μ/p=20%				**p=2K, n=2K, μ/p=20%**				**p=4K, n=2K, μ/p=20%**			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
Homotopy	0.980	0.912	0.849	0.682	0.977	0.923	0.854	0.834	0.970	0.923	0.845	0.775
DALM	0.976	0.915	0.865	0.825	0.973	0.921	0.885	0.924	0.962	0.946	0.926	0.898
TORR*	0.983	0.950	0.927	0.893	0.983	0.960	0.919	0.934	0.978	0.954	0.934	0.916
TORR25	0.965	0.899	0.842	0.762	0.961	0.909	0.828	0.770	0.958	0.905	0.848	0.752
RLHH	0.979	0.945	0.933	0.901	0.978	0.966	0.936	0.914	0.980	0.959	0.940	0.896
RoOFS	**0.991**	**0.986**	**0.974**	**0.933**	**0.993**	**0.991**	**0.976**	**0.946**	**0.993**	**0.988**	**0.975**	**0.923**
	p=2K, n=1K, μ/p=60%				**p=2K, n=1K, μ/p=20% (nd)**				**p=4K, n=2K, μ/p=20% (nd)**			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
Homotopy	0.979	0.932	0.829	0.708	0.972	0.923	0.853	0.717	0.985	0.913	0.868	0.789
DALM	0.975	0.939	0.863	0.826	0.965	0.910	0.886	0.842	0.984	0.951	0.937	0.889
TORR*	0.979	0.957	0.937	0.870	0.974	0.950	0.935	0.896	0.988	0.960	0.947	0.904
TORR25	0.952	0.912	0.833	0.690	0.952	0.911	0.859	0.758	0.968	0.908	0.864	0.745
RLHH	0.975	0.959	0.928	0.845	0.973	0.959	0.935	0.907	0.983	0.965	0.940	0.912
RoOFS	**0.982**	**0.984**	**0.962**	**0.910**	**0.989**	**0.993**	**0.985**	**0.947**	**0.994**	**0.991**	**0.988**	**0.933**

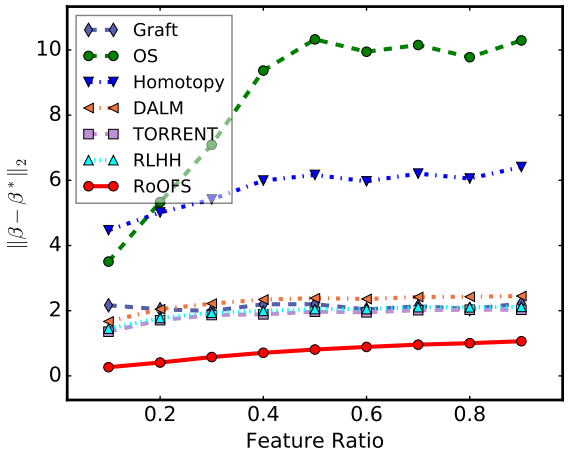
4.6.1 Datasets and Metrics

To demonstrate the performance of our proposed method, comprehensive experiments are performed in synthetic datasets whose simulation samples were randomly generated according to the model in Equation (7.1). Specifically, we sample the regression coefficients $\boldsymbol{\beta}^* \in \mathbb{R}^p$ as a random unit norm vector with feature ratio constraint $\|\boldsymbol{\beta}\|_0 = \mu$. The data matrix X was drawn independently and identically distributed from $\boldsymbol{x}_i \sim \mathcal{N}(\mathbf{0}, I_p)$ and the uncorrupted response variables were generated as $y_i^* = \boldsymbol{x}_i^T \boldsymbol{\beta}^*$. The set of uncorrupted samples S was selected as a uniformly random τ_*-sized subset of $[n]$. The response vector $\boldsymbol{y}$ containing corrupted samples was generated as $\boldsymbol{y} = \boldsymbol{y}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where the corruption vector $\boldsymbol{u}$ was sampled from the uniform distribution $\big[-5\|\boldsymbol{y}^*\|_\infty, 5\|\boldsymbol{y}^*\|_\infty\big]$ and the additive dense noise was $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. For the real-world data set, we applied our methods on the IMDb reviews data set for the review score prediction. The data set contains 50,000 popular movie reviews with the review score from 1 to 10 provided by the IMDb website. The adversarial data corruption vector $\boldsymbol{u}$ was appended to its original review score, where $\boldsymbol{u}$ was also sampled from the range $\big[-5\|\boldsymbol{y}^*\|_\infty, 5\|\boldsymbol{y}^*\|_\infty\big]$ randomly.

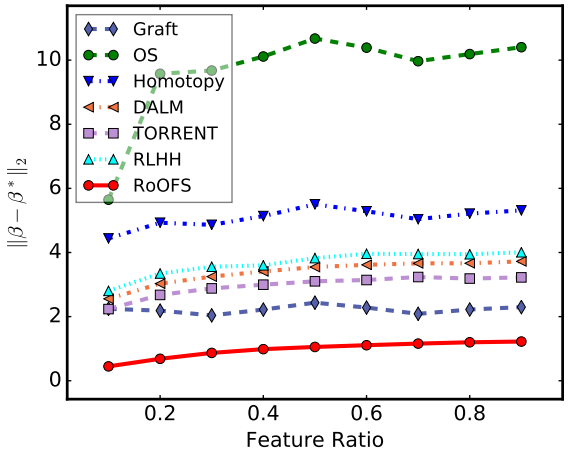
Following the setting in [5] [123], we measured the performance of the regression coefficients recovery using the standard L_2 error $e = \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_2$, where $\hat{\boldsymbol{\beta}}$ represents the recovered coefficients for each method and $\boldsymbol{\beta}^*$ 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. Similarly, the F1 score is also used to measure the effectiveness of feature selection by comparing the selected feature set with actual ones. To compare the scalability of each method, the CPU running time for each of the competing methods was also measured.

4.6.2 Comparison Methods

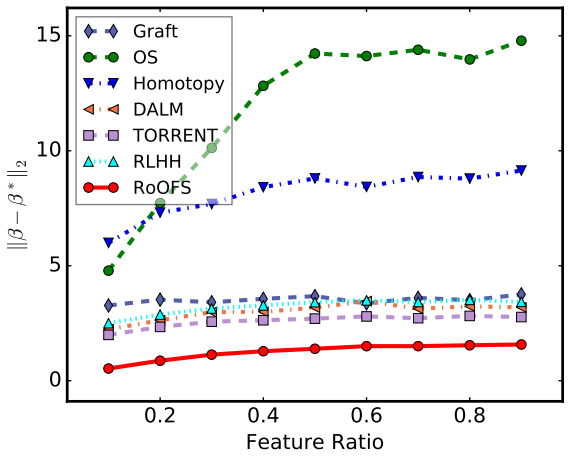
The following methods are included in the performance comparison presented here: *Grafting* [87]. The *Grafting* method is an online version of L_1 regularization approach to selects features. *Online Substitution* (*OS*) [114] is a parameter-free online feature selection algorithm with limited-memory. Both *Grafting* and *OS* cannot handle the adversarial data corruption and train models without considering data corruption. We also compared our method to the robust regression methods [109] [82]. *Homotopy* and *DALM* are two L_1 based solvers that outperform other L_1 methods both in terms of recovery properties and running time [113]. A hard thresholding method, *TORRENT (abbreviated "TORR")* [5], 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 two versions of parameter settings: *TORR** and *TORR25*. *TORR** uses the true corruption ratio as its parameter, and *TORR25* applies parameter that is uniformly distributed across the range of $\pm 25\%$ off the true value. Another recently proposed heuristic hard thresholding method, *RLHH* [123], is also compared in our experiment. The method is a parameter-free approach, where the



(a) p=2K, corruption ratio=20%



(b) p=4K, corruption ratio=20%



(c) p=2K, corruption ratio=40%

Figure 4.2: Performance on regression coefficients recovery for different ratios of feature ratio (n=1K, dense noise).

data corruption is estimated by a heuristic hard thresholding method. As all these robust methods are not designed for online feature selection, we run them individually in different feature batches and select features with largest μ weights in regression coefficients when $\|\boldsymbol{\beta}\|_0 = \mu$.

4.6.3 Recovery of regression coefficients

We selected 6 competing methods with which to evaluate the recovery performance of regression coefficients $\boldsymbol{\beta}$: *Grafting*, *OS*, *Homotopy*, *DALM*, *TORR*, *RLHH*. Figures 7.2(a) and 7.2(b) show the recovery performance for different feature numbers when the data size is fixed. The results show that 1) the proposed method, *RoOFS*, outperforms all the competing methods in all the setting of corruption ratios, and 2) The performance of *RoOFS* is very resistant to the corruption data because the error of *RoOFS* method increases much more slowly than others when corruption ratio increases from 5% to 40%. Figures 7.2(b) and 7.2(c) show that when data size increases, we have similar conclusion on the performance except the overall error is decreased since more data is applied. Figures 7.2(d) and 7.2(e) show that the result of coefficient recovery remains the same when the number of selected features increase. Figure 7.2(f) shows that almost all the methods without the dense noise setting perform more than 50% better than that in dense noise settings. Specifically, the error of *RoOFS* is close to zero which means it can almost exactly recover the ground true regression coefficients without the dense noise setting.

Figure 4.2 shows the recovery performance of regression coefficients in different ratios of feature sparsity. In general, the performance of *RoOFS* method outperforms all the other competing methods in all the data settings. Figure 4.2(a) and 4.2(b) show that 1) when the feature ratio increases, the recovery error of *RoOFS* method grows linearly with a small slope, which means our approach can be fitted into different settings of feature sparsity, and 2) the *RoOFS* method performs constantly well when the feature number increases from 2K to 4K. In addition, Figure 4.2(a) and 4.2(c) show that the *RoOFS* method is robust to corrupted data, because the error is not significantly impacted when the corruption ratio increases from 20% to 40%.

4.6.4 Recovery of Uncorrupted Set

As the online feature selection methods *Grafting* and *OS* do not explicitly estimate uncorrupted sets, we compared our proposed method with the robust methods: *Homotopy*, *DALM*, *TORR*, and *RLHH*. For the *TORR* algorithm, we use two parameter settings of *TORR** and *TORR25* for 0% and 25% deviation of true corrupted ratio, respectively. Table 4.2 shows the following: 1) *RoOFS* outperforms all the other methods up to 14.9% in different settings of data sizes, feature numbers and ratios of feature sparsity. 2) When increasing the corruption ratio, the F1 scores decrease for all the methods. Also, the F1 scores slightly increase 0.5%

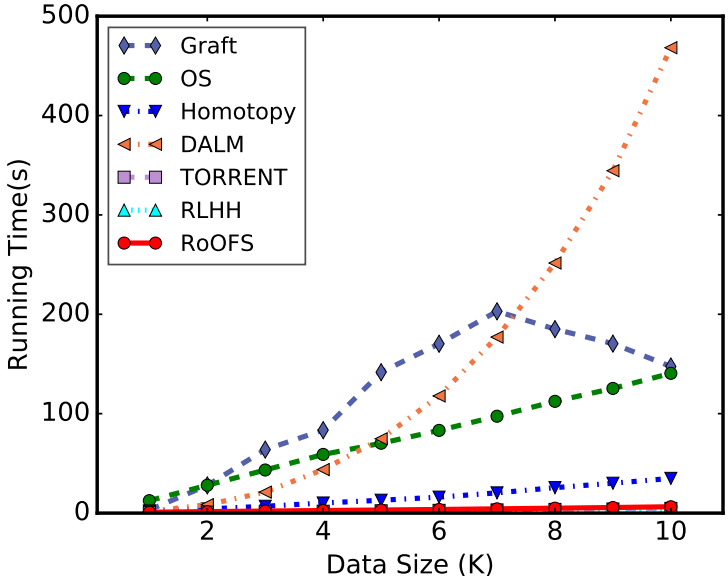
in average when data size become two times larger, which indicates the number of features has few influence on the estimation of uncorrupted set. 3) The result of *TORR* methods is highly dependent on the corruption ratio parameter: the results of *TORR** is up to 26.1% better than *TORR25*. It is important to note that the true corruption ratio parameter used in *TORR** cannot be estimated exactly in practice. 4) Without the dense noise settings, the F1 scores increase less than 1% compared to the F1 score based on the same setting with dense noise, which shows that dense noise has small impact on the performance of uncorrupted set recovery.

4.6.5 Performance of Feature Selection

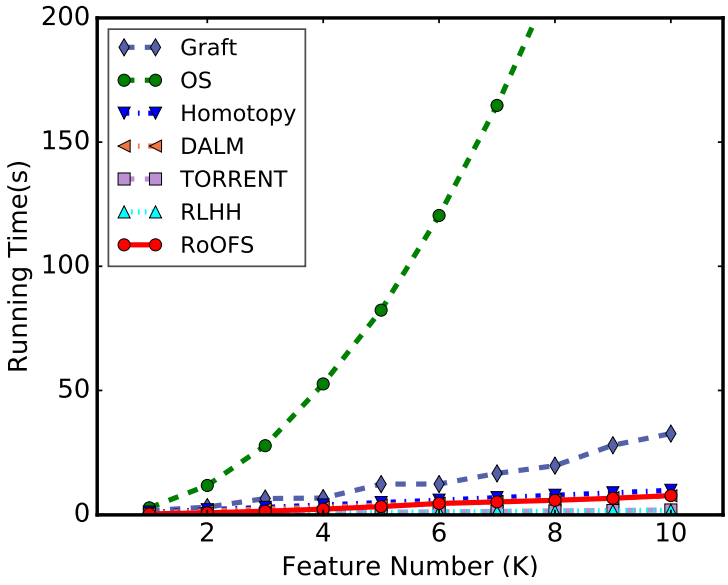
We selected all the six competing methods to evaluate the performance of feature selection in different settings including data sizes, feature numbers, and dense noises. For each data setting, we chose different ratios of feature sparsity (also known as μ/p) ranging from 10% to 60%. Table 4.3 shows the following: 1) the F1 scores of *RoOFS* method is up to 69.2% better than other methods, especially when the feature ratio is less than 40%. 2) Although the F1 scores of most methods such as *Grafting* and *TORR* are above 0.6 when the ratio is larger than 50%, the performance degraded significantly when the ratio decreased to 10%. However, the F1 score of *RoOFS* method is constantly higher than 0.85 in all the ratios of features. 3) *OS* method is very competitive in the task of feature selection; however, it still has lower F1 scores in all the settings when the ratio is less than 50%. 4) The setting of dense noise does not have significant impact on the performance of feature selection, since the F1 score without dense noise is only less than 1% larger than that in dense noise setting.

4.6.6 Performance in real-world data

To evaluate the robustness of our proposed methods in a real-world dataset, we compared the performance of sentiment prediction in different corruption settings, ranging from 5% to 40%. The dataset was first proposed by Maas et al. [66] as a benchmark for sentiment analysis. It consists of movie reviews retrieved from IMDB website. For each movie review, it contains several sentences. The total 100K movie reviews are divided into three parts: 25K labeled training samples, 25K labeled test samples and 50K unlabeled training samples. The unlabeled data were designed as the additional corruption to the dataset: the score of sentiment were random number between one to ten. Table 4.4 shows the mean absolute error of sentiment prediction in the IMDB datasets. From the result, we can conclude: 1) *RoOFS* method outperform all the other methods in different corruption settings. 2) Although the absolute error of the other methods such as *Homotopy* and *DALM* are above 4, the performance varied significantly when the ratio changed because these methods highly dependent on the parameters and it's hard to estimate the feature sparsity ratio and true corruption ratio in the real-world data. However, the performance of *RoOFS* method is



(a) p=2K, μ/p=20%, cr=10%



(b) n=1K, μ/p=20%, cr=10%

Figure 4.3: Running time for different data and feature sizes.

Table 4.3: F1 Score on Performance of Feature Selection (cr=30%).

	p=10K, n=10K, dense						**p=20K, n=10K, dense**					
	10%	20%	30%	40%	50%	60%	10%	20%	30%	40%	50%	60%
Grafting	0.130	0.201	0.302	0.543	0.759	0.789	0.111	0.399	0.642	0.773	0.844	0.895
OS	0.706	0.649	0.626	0.643	0.810	0.678	0.611	0.611	0.606	0.689	0.836	**0.975**
Homotopy	0.116	0.217	0.304	0.406	0.498	0.667	0.109	0.202	0.417	0.625	0.750	0.833
DALM	0.130	0.219	0.309	0.418	0.516	0.597	0.114	0.215	0.390	0.404	0.502	0.603
TORR	0.275	0.297	0.368	0.446	0.525	0.647	0.320	0.338	0.395	0.458	0.535	0.623
RLHH	0.193	0.261	0.338	0.416	0.510	0.647	0.322	0.336	0.390	0.461	0.536	0.628
RoOFS	**0.911**	**0.910**	**0.876**	**0.870**	**0.895**	**0.891**	**0.876**	**0.842**	**0.832**	**0.848**	**0.881**	0.906
	p=10K, n=5K, dense						**p=10K, n=10K, no dense**					
	10%	20%	30%	40%	50%	60%	10%	20%	30%	40%	50%	60%
Grafting	0.114	0.206	0.426	0.641	0.765	0.836	0.142	0.224	0.293	0.527	0.698	0.797
OS	0.657	0.596	0.618	0.688	0.840	**0.961**	0.688	0.682	0.647	0.649	0.655	0.688
Homotopy	0.107	0.207	0.304	0.395	0.500	0.667	0.126	0.203	0.307	0.405	0.500	0.667
DALM	0.113	0.210	0.311	0.396	0.504	0.602	0.140	0.226	0.308	0.407	0.504	0.609
TORR	0.312	0.336	0.391	0.461	0.532	0.627	0.475	0.448	0.472	0.521	0.579	0.646
RLHH	0.314	0.334	0.388	0.463	0.530	0.624	0.476	0.458	0.487	0.531	0.585	0.646
RoOFS	**0.873**	**0.830**	**0.837**	**0.859**	**0.883**	0.909	**0.917**	**0.922**	**0.898**	**0.900**	**0.889**	**0.900**

constantly above 3.10 in all the ratios of corruption. 3) It is true that *OS* has a very competitive performance in all the corruption settings because the deviation of corruption is small, which is less than 50% from the labeled data. But the running time of *OS* is too high to train the data which has 10k features. 4) When increasing the corruption ratio, the absolute error of *RoOFS* method decreased.

4.6.7 Efficiency

To evaluate the efficiency of our proposed method, we compared the performances of all the competing methods for two difference settings: data sizes and feature numbers. For *Grafting* and *OS* methods, the online features are handled individually due to their design. For the other methods, a hundred features are handled together as a batch. As Figure 7.4 shows, we found the following: 1) *RoOFS* algorithm has a very competitive efficiency compared to the thresholding based methods, *TORR* and *RLHH*, and significantly outperforms other four methods. 2) The running time of *RoOFS* algorithm increases linearly when both data size

Table 4.4: Mean Absolute Error of Sentiment Prediction.

	p=10K, n=10K					
	5%	10%	20%	30%	40%	Avg
Grafting	3.162	3.162	3.162	3.162	3.162	3.162
OS	3.111	3.078	3.078	3.113	3.108	3.098
Homotopy	4.157	3.925	3.894	3.828	3.540	3.869
DALM	3.573	3.311	3.523	3.459	3.252	3.424
TORR	5.090	3.928	4.596	5.147	4.218	4.596
RLHH	5.448	4.198	3.716	4.269	4.626	4.451
RoOFS	**3.074**	**3.073**	**3.072**	**3.070**	**3.066**	**3.071**

and feature number increase, which indicates that our algorithm can be scaled to massive datasets. 3) The running time of *DALM* method increases exponentially when data size increases, however, its efficiency has rarely impacted by increasing the number of features. 4) The efficiency of *Grafting* method fluctuates largely on the different data sizes, which indicates that its running time depends on the data size and content of data.

4.7 Conclusion

In this paper, a novel robust regression algorithm via online feature selection, *RoOFS*, is proposed to recover the regression coefficients and the uncorrupted set under the assumption that features cannot be accessed entirely at one time. To achieve this, we designed a robust online substitution method to alternately estimate the optimal uncorrupted set and substitute the retained feature set with newly updated features. We demonstrate that our algorithm can recover regression coefficients with a restricted error bound compared to ground truth. Extensive experiments on massive simulation data demonstrated that the proposed algorithm outperforms other competing methods in both effectiveness and efficiency.

Chapter 5

Self-Paced Robust Learning for Leveraging Clean Labels in Noisy Data

This chapter presents a novel self-paced robust model to leverage the clean labels for training a large but noisy dataset. The work is introduced in Section 5.1. Then, Section 5.2 reviews related work, and Section 5.3 introduces the background and problem setup. The proposed self-paced robust learning algorithm is presented in Section 5.4. In particular, the convergence analysis is shown in Section 5.4.2 and Section 5.4.3 gives two motivative examples of robust regression and classification tasks. Experiments on both synthetic and real-world datasets are presented in Section 5.5, and the paper concludes with a summary of the research in Section 5.6.

5.1 Introduction

The availability of well-labeled data is becoming a key factor for training a machine learning model with high complexity, such as deep neural networks [57]. However, collecting a large-scale dataset with clean labels usually requires cumbersome verification work by human beings, which is time-consuming and expensive. Usually it is much easier to obtain a small set of data precisely labeled by human experts and collect large quantities of noisy labels by using crowd-sourcing [7] or "weak annotators" based on heuristics [9]. For example, to collect the training images of certain keywords, one possible solution using a "weak annotator" is to search with the keyword in an image search engine such as *Google Images*[1]. However, the images collected by searching for a polysemous word such as "apple" will show not only the fruit , but also the logo or products of *Apple* company. Although it is easy and cheap

[1] `https://images.google.com/`

for a so-called "weak annotator" to collect a massive amount of training data, the collected training samples will inevitably contain a large amount of noise.

Leveraging the prior knowledge of clean labels in noisy data is actually a crucial issue in practice, but existing robust learning methods [72, 122] typically focus more on eliminating noisy data. However, the data collected by "weak annotator" or crowd-sourcing can be too noisy for existing robust methods to train an accurate model. Moreover, existing work that utilize additional clean labels are usually designed for some specific problems such as image classification [105, 120]. These methods typically utilize clean labels in large-scale noisy data based on their additional domain knowledge [45, 59]; however, these approaches are difficult to handle extremely noisy data and relied on their domain knowledge [59] heavily, which makes them difficult be used in more general problems. For those seeking to address these issues, the major challenges can be summarized as follows. 1) *Infeasibility to train an accurate model using clean or noisy data individually.* Training models in a small amount of well-labeled data will usually cause the overfitting problem and result in an inaccurate model. On the other hand, weakly labeled data may contain a large amount of corrupted data, e.g., more than 50% data can be corrupted, which makes it hard for most of the robust learning approaches to resist the noise. 2) *Difficulty to learn models from noisy data under the supervision of a small amount of clean labels.* Models trained in a small amount of clean data is highly vulnerable to be overfitting and bias. If the biased model is directly applied to identify the uncorrupted samples in noisy data, some corrupted samples can be mistakenly included in the training set. 3) *Lack of domain knowledge to utilize clean labels to extract the uncorrupted samples from noisy data.* Most of the existing work using additional clean labels depends on extra domain knowledges, such as utilizing the information of label relations in a knowledge graph [59] for image classification task. However, these domain knowledge is usually hard to obtain in practice and prevents these methods to be more extensively used in general situations.

To address all these technical challenges, this paper presents a novel self-paced robust learning approach, named *SPRL*, to leverage the clean labels in noisy data by a self-paced training process based on samples in a nearly optimal sequence on the noisiness. The main contributions of this paper are as follows: 1) *Formulating a framework to leverage the clean labels in noisy data.* A framework is proposed to utilize clean labels in a large-scale noisy dataset. Specifically, the clean labels are assumed to contain a limited size of data while the noisy dataset may contain an extremely large amount of data corruption. The approaches in solving robust regression and classification tasks are presented, which demonstrates the proposed framework can be generally used in various tasks. 2) *Proposing a self-paced robust learning algorithm to train models under the supervision of clean labels.* The proposed self-paced algorithm learns the data samples from clean to noisy under the supervision of clean labels, which helps to hedge the risk of involving corrupted data into the training set. Furthermore, the algorithm learns the training data in an order that are dynamically determined by the feedback of the learner itself without additional prior knowledge, which makes it more extensively utilized in practice. Furthermore, 3) *Providing a theoretical analysis for*

the convergence of the proposed algorithm. We prove that our self-paced robust learning algorithm converges under the assumption that the loss function selected for the estimated model has a finite lower bound. Specifically, the objective function of our algorithm can be monotonically decreased in accord with the increasing learning pace parameter until it reaches the lower bound. 4) *Conducting extensive experiments for performance evaluations.* The proposed method was evaluated on both synthetic data and real-world datasets in robust regression and classification tasks with different corruption and data-size settings. The results demonstrate that the proposed approaches consistently outperform existing methods along multiple metrics. To the best of our knowledge, this is the first work to train models from noisy dataset by utilizing clean labels in a self-paced learning process.

5.2 Related Work

The work related to this paper is summarized in three categories below.

5.2.1 Self-Paced Learning

In recent years, self-paced learning [53] has received widespread attention for various applications in machine learning, such as image classification [44], event detection [42] and object tracking [104, 118]. Inspired by the learning processes used by humans and animals [4], self-paced learning (*SPL*) [53] considers training data in a meaningful order, from easy to hard, to facilitate the learning process. Unlike standard curriculum learning [4], which learns the data in a predefined curriculum design based on prior knowledge, *SPL* learns the training data in an order that is dynamically determined by feedback from the learning process itself, which means it can be more extensively utilized in practice. Furthermore, a wide assortment of *SPL*-based methods [64, 88] have been developed, including self-paced curriculum learning [44], self-paced learning with diversity [43], multi-view [112] and multi-task [50, 58] self-paced learning. In addition, several researchers have conducted theoretical analyses of self-paced learning. Meng et al. [76] provides a theoretical analysis of the robustness of *SPL*, revealing that the implicit objective function of *SPL* has a similar configuration to a non-convex regularized penalty. Such regularization restricts the contributions of noisy examples to the objective, and thus enhances the learning robustness. Ma et al. [65] proved that the learning process of *SPL* always converges to critical points of its implicit objective under mild conditions. However, none of the existing self-paced learning approaches can be applied to our problem of leveraging clean labels in noisy data.

5.2.2 Robust Learning

A large body of literature on the robust learning problem has been established over the last few decades. Most of the studies aim to directly learn from noisy labels and focus on noise-robust algorithms. For instance, Chen et al. [19] proposed a robust algorithm based on trimmed inner product. McWilliams et al. [72] proposed a sub-sampling algorithm for large-scale corrupted linear regression. Bhatia et al. [5] and Zhang et al. [122] proposed hard-thresholding based methods with strong guarantees of coefficient recovery under a mild assumption on datasets. Another group of methods focuses on removing or correcting mislabeled data. For example, some work utilized heavy-tailed distributions [129] such as Student t-distribution and Poisson distribution, to model the mislabeled data, while others detected these outliers based on Gaussian distribution [39, 101] under the assumption that outliers have a small probability of occurrence in the population. Some methods do not assume any prior knowledge on the data distribution based on kernel functions [56,92]. These approaches utilize kernel functions to approximate the actual density distribution and declare the instances lying in the low probability area of the kernel density function as outliers. However, all these approaches typically jointly learn the clean and noisy data together, but cannot fully leverage the information contained in the clean set.

5.2.3 Weakly-Supervised Learning

Recently, some work in weakly-supervised learning [75] utilized additional clean labels in learning a noisy dataset. For instance, Azadi et al. [120] proposed an auxiliary image regularization to train a deep convolutional neural network in noisy labeled image data, in which a limited number of training examples are supplied with clean labels. In order to classify images from weakly labeled data, Li et al. [59] used not only a small clean dataset, but some other "side" information of label relations in a knowledge graph. Veit et al. [105] used millions of images with noisy annotations in conjunction with a small set of cleanly-annotated images to learn effective image representations, while Jiang et al. [45] designed a curriculum paradigm to learn the instance weights in corrupted labels to prevent deep convolutional neural networks from overfitting. Compared to existing work utilizing the clean dataset, our methods are different in two ways. First, we consider the learning process from clean to noisy data in a self-paced manner, which hedges the risk of training corrupted data samples. Moreover, our model learns the instance weight determined by the feedback of the learner itself without using any additional prior knowledge, which means our methods can be applied to more general problems in practice.

5.3 Preliminary

First, we give an introduction to self-paced learning in Section 5.3.1. Then we describe the problem formulation in Section 5.3.2.

5.3.1 Background: Self-Paced Learning

Inspired by the learning process and cognitive mechanism of humans, self-paced learning (*SPL*) provides a strategy for learning models in a progression of easier-to-harder data instances. Given a set of n training instances along with their labels $\mathcal{D} = \big\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\big\}$, the general form of the objective function for self-paced learning is given by

$$\arg\min_{\boldsymbol{w},\boldsymbol{v}} \sum_{i=1}^{n} v_i \mathcal{L}\big(y_i, f(x_i, \boldsymbol{w})\big) + \psi(\boldsymbol{w}) + \lambda r(\boldsymbol{v}), \tag{5.1}$$

where $\mathcal{L}$ is the loss function to measure the error between label y_i and the estimated value from model f. $\psi(\boldsymbol{w})$ and $r(\boldsymbol{v})$ are the regularization terms for model parameters $\boldsymbol{w}$ and the instance weights $\boldsymbol{v}$, respectively. The parameter λ represents the "age" of the *SPL* algorithm to control the learning pace. In the process of self-paced learning, the value of λ is gradually increased to learn new samples. When λ is small, the "easier" samples with smaller losses are considered. As λ grows, more "harder" samples with larger losses will be gradually added into the training set.

With this setting, the Equation (5.1) is a bi-convex optimization problem over $\boldsymbol{w}$ and $\boldsymbol{v}$, which can be efficiently solved by the alternate convex search (*ACS*) method [35]. Given a fixed $\boldsymbol{v}$, the solution for $\boldsymbol{w}$ can be obtained using any off-the-shelf solver, and when $\boldsymbol{v}$ is fixed, the analytical solution for $\boldsymbol{v}$ can be given as follows:

$$v_i = \begin{cases} 1, & \text{if } \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) < \lambda \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

An intuitive explanation for this alternative search strategy in the perspective of robust learning can be described as follows: 1) When updating instance weights $\boldsymbol{v}$ with a fixed $\boldsymbol{w}$, an instance whose loss is smaller than the threshold λ is taken as an "uncorrupted" sample, and it will be selected in the training set with its instance weight equal to 1; otherwise, the instance will not be selected. 2) When updating $\boldsymbol{w}$ with a fixed $\boldsymbol{v}$, the model is trained on the selected "uncorrupted" training set in the last step. If λ is small, only a small proportion of samples with small losses will be trained. But the size of the selected training samples is not large enough to train an accurate model. In case λ is large, some corrupted samples might be included into the training set, which will lead to an inaccurate model. Therefore, the performance of self-paced learning will be highly impacted by the choice of parameter λ.

5.3.2 Problem Formulation

In the setting of self-paced robust learning for leveraging clean labels in noisy data, we consider the samples to be provided in two parts with different qualities: a small set of well-labeled samples with little data corruption $\mathcal{D}_s = \big\{(\boldsymbol{x}_1, y_1) \ , \ldots, (\boldsymbol{x}_k, y_k)\big\}$ and a weakly labeled dataset $\mathcal{D}_w = \{(\boldsymbol{x}_{k+1}, y_{k+1}), \ldots, (\boldsymbol{x}_n, y_n)\}$ in large size, where $\boldsymbol{x}_i \in \mathbb{R}^p$ represents the i^{th} sample data and y_i is the corresponding label. We assume the well-labeled data contains k samples and the weakly labeled data has $n-k$ samples, where n is much larger than k ($n \gg k$).

The goal of our study is to infer the model parameter $\boldsymbol{w} \in \mathbb{R}^p$ based on the uncorrupted data samples $\mathcal{D}^+ = \mathcal{D}_s \cup \mathcal{D}_w^+$, where the set $\mathcal{D}_w^+$ represents the uncorrupted data samples in the weakly labeled dataset $\mathcal{D}_w$, in which the samples are correctly labeled. All the samples in $\mathcal{D}_s$ are uncorrupted and correctly labeled. Therefore, our purpose is to fully utilize the uncorrupted data samples in both the clean set $\mathcal{D}_s$ and the noisy set $\mathcal{D}_w$, which can be formalized as follows:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^p}{\arg\min} \sum_{i\in\mathcal{D}_s\cup\mathcal{D}_w^+} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) + \psi(\boldsymbol{w}), \tag{5.3}$$

where $\mathcal{L}$ is the loss function to measure the error between label y_i and the estimated value from model f. For example, the linear model $f(\boldsymbol{x}_i, \boldsymbol{w})\big) = \boldsymbol{x}_i^T\boldsymbol{w}$ and squared loss $\|y_i - f(\boldsymbol{x}_i, \boldsymbol{w})\big)\|_2^2$ can be applied to a regression problem. The regularization term $\psi(\boldsymbol{w})$ restricts the complexity of model parameters $\boldsymbol{w}$. The notations used in this paper are summarized in Table 7.2.

The problem defined in Equation (5.3) is challenging in the following two aspects. First, the uncorrupted set $\mathcal{D}_w^+$ in the weakly labeled set $\mathcal{D}_w$ is unknown. Simply ignoring the whole noisy data $\mathcal{D}_w$ is not a proper solution because the amount of well-labeled data is not large enough to train an accurate model. Second, the weakly labeled data $\mathcal{D}_w$ can be extremely noisy or even contain adversarial data corruption, which makes the uncorrupted samples in $\mathcal{D}_w$ difficult to estimate. In the next section, we present a self-paced-learning approach to address these challenges.

5.4 Proposed Method

In Section 5.4.1, we propose the *SPRL* algorithm to utilize clean labels in training noisy datasets. After that, the convergence analysis of our algorithm is presented in Section 5.4.2. Finally, we give two motivative examples for robust regression and classification tasks in Section 5.4.3.

Table 5.1: Math Notations

Notations	**Explanations**
p	number of feature in data matrix X
k	number of samples in the clean dataset
n	number of samples in the entire dataset
$X, \boldsymbol{y}$	data matrix and its corresponding label vector
$\boldsymbol{w}$	parameters of estimated model
$\tilde{\boldsymbol{w}}$	model parameter trained in the clean set
$\boldsymbol{v}$	instance weight vector
λ	parameter to control the learning pace
μ	step size of parameter λ
$\mathcal{L}$	loss function of estimated model

5.4.1 SPRL Algorithm

In order to utilize the prior knowledge on the high-quality data $\mathcal{D}_s$ on the weakly labeled data $\mathcal{D}_w$, we reformulate our objective function $\mathcal{J}$ as follows:

$$\begin{aligned}\underset{\boldsymbol{w}\in\mathbb{R}^n,\boldsymbol{v}\in[0,1]}{\arg\min}\ \mathcal{J}(\boldsymbol{w},\boldsymbol{v};\lambda) &= \sum_{i=1}^{k}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i,\boldsymbol{w})\big) \\ + \sum_{i=k+1}^{n} v_i\mathcal{L}\big(y_i, f(\boldsymbol{x}_i,\boldsymbol{w})\big) &+ \|\boldsymbol{w}\|_2^2 + \theta\|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|_2^2 - \lambda\sum_{i=k+1}^{n} v_i,\end{aligned} \tag{5.4}$$

where $\mathcal{L}$ represents the loss function and variable v_i is denoted as the weight of the i^{th} data instance, which is allowed to take any value in the interval $[0, 1]$. The first term is the total loss of the clean set $\mathcal{D}_s$, in which we assume all the samples in clean set are included into our training set. The second term represents the total loss of noisy set $\mathcal{D}_w$, where the samples are selected by the instance weight vector $\boldsymbol{v}$. The term $\|\boldsymbol{w}\|_2^2$ is to control the complexity of model parameters $\boldsymbol{w}$. The variable $\tilde{\boldsymbol{w}}$ represents the model weights trained by the clean set $\mathcal{D}_s$. Since the clean set is assumed to contain few data corruption, we can minimize the following objective function without considering its data noises:

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\arg\min}\sum_{i=1}^{k}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i,\boldsymbol{w})\big) + \psi(\boldsymbol{w}), \tag{5.5}$$

where $\psi(\boldsymbol{w})$ is the regularization term to control the complexity of $\tilde{\boldsymbol{w}}$. The term $\theta\|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|_2^2$ is to control the difference between the estimated model and the model $\hat{\boldsymbol{w}}$ trained in the clean set only. If a larger parameter θ is given, the algorithm prefers to give a similar estimated model as the $\hat{\boldsymbol{w}}$. The last term $\lambda\sum_{i=k+1}^{n} v_i$ is the regularization term for instance weights $\boldsymbol{v}$, where parameter λ controls the learning pace.

The problem defined in Equation (5.4) can be solved based on alternate convex search (ACS) method [35]. When model parameter $\boldsymbol{w}$ is fixed, the variable $\boldsymbol{v}$ in the iteration t+1 can be

ALGORITHM 5: SPRL ALGORITHM

Input: $X \in \mathbb{R}^{p\times n}$, $\boldsymbol{y} \in \mathbb{R}^n$, $\theta \in \mathbb{R}$, $\lambda^0 \in \mathbb{R}$, $\lambda_\infty \in \mathbb{R}$, $\mu \in \mathbb{R}$
Output: solution $\boldsymbol{w}^{(t+1)}$, $\boldsymbol{v}^{(t+1)}$
$\tilde{\boldsymbol{w}} \leftarrow \arg\min_{\boldsymbol{w}} \sum_{i=1}^{k} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) + \psi(\boldsymbol{w})$
Initialize $\boldsymbol{w}^0 = \tilde{\boldsymbol{w}}$, $\varepsilon > 0$, $t \leftarrow 0$
repeat
  for $i = k+1 \dots n$ **do**
    $v_i^{t+1} \leftarrow \mathbb{1}\Big(\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^t)\big) < \lambda^t\Big)$
  end
  Update $\boldsymbol{w}^{t+1}$ by Equation (5.8) with fixed $\boldsymbol{v}^{t+1}$ and $\tilde{\boldsymbol{w}}$.
  $\lambda^{t+1} \leftarrow \lambda^t * \mu$
  if $\lambda^{t+1} > \lambda_\infty$ **then**
    $\lambda^{t+1} \leftarrow \lambda_\infty$
  $t \leftarrow t+1$
until $\|\mathcal{J}(\boldsymbol{w}^{t+1}, \boldsymbol{v}^{t+1}; \lambda^{t+1}) - \mathcal{J}(\boldsymbol{w}^t, \boldsymbol{v}^t; \lambda^t)\|_2 < \varepsilon$
return $\boldsymbol{w}^{t+1}$, $\boldsymbol{v}^{t+1}$

solved by the following sub-problem:

$$v_i^{t+1} = \arg\min_{v_i \in [0,1]} \sum_{i=k+1}^{n} v_i \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^t)\big) - \lambda^t \sum_{i=k+1}^{n} v_i, \tag{5.6}$$

where λ^t represents the value of λ in the t^{th} iteration. A closed-form solution of $\boldsymbol{v}^{t+1}$ can be given as follows:

$$v_i^{t+1} = \begin{cases} 1, & \text{if } \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^t)\big) < \lambda^t \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

When we fix the value of variable $\boldsymbol{v}^{t+1}$, variable $\boldsymbol{w}^{t+1}$ can be solved by the following sub-problem:

$$\begin{aligned} \boldsymbol{w}^{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^p} \sum_{i=1}^{k} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) + \sum_{i=k+1}^{n} v_i^{t+1} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) \\ + \|\boldsymbol{w}\|_2^2 + \theta \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2^2 \end{aligned} \tag{5.8}$$

The sub-problem can be easily solved by an off-the-shelf optimizer when the loss function is convex. Two specific examples of squared loss and hinge loss are given in Section 5.4.3. In traditional self-paced learning, the value of variable λ is increased until the objective function is converged. However, in robust learning, when λ grows, more corrupted samples with larger losses will be gradually added into the training set. Even a few corrupted samples can lead to an extremely large loss if the value of λ is too large. Therefore, in order to keep the corrupted data out of our training set, we introduce a threshold parameter λ_∞ to control the size of training set.

The details of the *SPRL* algorithm are presented in Algorithm 5. In Line 1, the model parameter $\tilde{\boldsymbol{w}}$ is trained in the clean set. In Lines 2, the variable $\boldsymbol{w}^0$ is initialized as the value of $\tilde{\boldsymbol{w}}$. The variable $\boldsymbol{v}^{t+1}$ is updated in Line 5 with variable $\boldsymbol{w}^t$ fixed. After which, the model parameter $\boldsymbol{w}^{t+1}$ is optimized by Equation (5.8) in Line 6 with the variables $\boldsymbol{v}^{t+1}$ fixed. In Lines 8-11, the parameter λ is enlarged to include more data instances in the training set, where λ_∞ is the threshold parameter for λ and μ is the step size. The algorithm will be stopped when the objective function is converged in Line 13. The convergence of the algorithm will be proved in Section 5.4.2.

5.4.2 Convergence Analysis

In this section, we will present the theoretical analysis on the convergence of the proposed algorithm.

First, the assumption on the loss function $\mathcal{L}$ is as follows.

Assumption 1 (Lower Bound)**.** *The loss function $\mathcal{L}$ in problem* (5.4) *has a lower bound $\mathcal{B}$ as follows:*

$$\mathcal{B} = \min_{\boldsymbol{w}} \mathcal{L}\big(y, f(\boldsymbol{x}, \boldsymbol{w})\big) > -\infty \tag{5.9}$$

This assumption can be easily satisfied by most loss functions; e.g., least-squares loss and hinge loss have their lower bound $\mathcal{B} = 0$. Then we can get the following property of the objective function:

Lemma 1. *The objective function $\mathcal{J}$ in Equation* (5.4) *is lower bounded as follows:*

$$\lim_{t\to\infty} \mathcal{J}(\boldsymbol{w}^t, \boldsymbol{v}^t; \lambda^t) > -\infty \tag{5.10}$$

Proof. The objective function $\mathcal{J}$ has the following property:

$$\begin{aligned} &\mathcal{J}(\boldsymbol{w}^t, \boldsymbol{v}^t; \lambda^t) \\ &\quad \overset{(a)}{\geq} \sum_{i=1}^{k} \mathcal{B} + \sum_{i=k+1}^{n} v_i^t \mathcal{B} + \|\boldsymbol{w}^t\|_2^2 + \theta\|\boldsymbol{w}^t - \tilde{\boldsymbol{w}}\|_2^2 - \lambda^t \sum_{i=k+1}^{n} v_i^t \\ &\quad \overset{(b)}{\geq} k\mathcal{B} + \sum_{i=k+1}^{n} v_i^t \mathcal{B} - (n-k)\cdot\lambda_\infty \end{aligned} \tag{5.11}$$

Inequality (a) follows from $\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^t)\big) \geq \mathcal{B}$ and $v_i^t \geq 0$. Inequality (b) follows from $\|\boldsymbol{w}^t\|_2^2 \geq 0$, $\theta\|\boldsymbol{w}^t - \tilde{\boldsymbol{w}}\|_2^2 \geq 0$, and λ_∞ is the maximum threshold of parameter λ. When all the v_i^t are equal to 1, $\lambda^t \sum_{i=k+1}^{n} v_i^t$ reaches its minimum value $(n-k)\cdot\lambda_\infty$. When lower bound

$B \geq 0$, we have $\sum_{i=k+1}^{n} v_i^t \mathcal{B} \geq 0$; otherwise, when $B < 0$, we have $\sum_{i=k+1}^{n} v_i^t \mathcal{B} \geq (n-k) \cdot \mathcal{B}$. Therefore, we have

$$\begin{aligned} \mathcal{J}(\boldsymbol{w}^t, \boldsymbol{v}^t; \lambda^t) &\geq k\mathcal{B} + \min\left\{0, (n-k) \cdot \mathcal{B}\right\} - (n-k) \cdot \lambda_\infty \\ &= k\mathcal{B} + (n-k) \cdot \left(\min\left\{0, \cdot \mathcal{B}\right\} - \lambda_\infty\right) \end{aligned} \tag{5.12}$$

Since $\mathcal{B} > -\infty$ and λ_∞ is constant, we have $\mathcal{J}(\boldsymbol{w}^t, \boldsymbol{v}^t; \lambda^t) > -\infty$ for $\forall t = 1 \ldots \infty$. □

Theorem 1. *When Assumption 1 is satisfied, Algorithm 5 converges with the following property:*

$$\lim_{t \to \infty} \left\| \mathcal{J}^{t+1} - \mathcal{J}^t \right\|_2 = 0 \tag{5.13}$$

The proof of Theorem 7 can be found in Appendix A.1.

5.4.3 Motivative Examples

In this section, we discuss two special cases of our self-paced robust learning approach: robust linear regression and robust binary classification based on the support vector machine (SVM).

Robust Linear regression

For robust linear regression, the objective function can be represented as follows:

$$\begin{aligned} \underset{\boldsymbol{w} \in \mathbb{R}^n, \boldsymbol{v} \in \{0,1\}^n}{\arg\min} \; & \mathcal{J}(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{k} \|y_i - \boldsymbol{x}_i^T \boldsymbol{w}\|_2^2 \\ + \sum_{i=k+1}^{n} & v_i \|y_i - \boldsymbol{x}_i^T \boldsymbol{w}\|_2^2 + \|\boldsymbol{w}\|_2^2 + \theta \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2^2 - \lambda \sum_{i=k+1}^{n} v_i, \end{aligned} \tag{5.14}$$

where the squared loss is chosen as the loss function $\mathcal{L}$ and linear regression $f(\boldsymbol{x}_i, \boldsymbol{w}) = \boldsymbol{x}^T \boldsymbol{w}$ is applied as its model f. The closed-form solutions for both $\boldsymbol{w}$ and $\boldsymbol{v}$ are as follows:

$$\begin{aligned} \boldsymbol{w} &= \left(\sum_{i=1}^{k} \boldsymbol{x}_i \boldsymbol{x}_i^T + \sum_{i=k+1}^{n} v_i \boldsymbol{x}_i \boldsymbol{x}_i^T + \big(\theta + 1\big) I \right)^{-1} \\ &\qquad \left(\sum_{i=1}^{k} \boldsymbol{x}_i y_i + \sum_{i=k+1}^{n} v_i \boldsymbol{x}_i y_i + \theta \tilde{\boldsymbol{w}} \right) \\ \boldsymbol{v} &= \mathbb{1}\left[\|\boldsymbol{x}_i^T \boldsymbol{w} - y_i\|_2^2 < \lambda \right] \end{aligned} \tag{5.15}$$

Robust Binary Classification

For the robust binary classification problem, the objective function is represented as follows based on the support vector machine:

$$\begin{aligned}\underset{\boldsymbol{w}\in\mathbb{R}^n,\boldsymbol{v}\in\{0,1\}^n}{\arg\min}\ \mathcal{J}(\boldsymbol{w},\boldsymbol{v};\lambda) &= \sum_{i=1}^{k}\max\big(0,1-y_i\langle\boldsymbol{w},\boldsymbol{x}_i\rangle\big)\\ &+\sum_{i=k+1}^{n}v_i\max\big(0,1-y_i\langle\boldsymbol{w},\boldsymbol{x}_i\rangle\big)+\|\boldsymbol{w}\|_2^2\\ &+\theta\|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|_2^2-\lambda\sum_{i=k+1}^{n}v_i\end{aligned}\tag{5.16}$$

A primal estimated sub-gradient SVM solver, Pegasos [97], is used to update the variable $\boldsymbol{w}$ based on its primal problem. In the Pegasos algorithm, the number of iterations required to obtain a solution of accuracy ϵ is $\mathcal{O}(1/\epsilon)$. The update of variable $\boldsymbol{w}$ is shown as follows:

$$\boldsymbol{w}^{t+1}=\boldsymbol{w}^t-\eta_t\bigg(\big(2+2\theta\big)\boldsymbol{w}^t-2\theta\tilde{\boldsymbol{w}}-\sum_{i\in\mathcal{A}^t}y_i\boldsymbol{x}_i\bigg),\tag{5.17}$$

where the step size $\eta^t=1/(\alpha t)$ and α is a fixed parameter to control the size of the step. The set $\mathcal{A}^t$ for the t^{th} iteration is defined as:

$$\mathcal{A}^t=\{i\in[n]:y_i\langle\boldsymbol{w}^t,\boldsymbol{x}_i\rangle<1,\ v_i^t=1\}\tag{5.18}$$

The set $\mathcal{A}^t$ excludes all the data samples that either have a zero value with loss $\max\big(0,1-y_i\langle\boldsymbol{w},\boldsymbol{x}_i\rangle\big)$ or are disabled by self-paced learning in the current iteration. The closed-form solution for variable $\boldsymbol{v}$ can be presented as follows:

$$\boldsymbol{v}^{t+1}=\mathbb{1}\big[\max(0,1-y_i\langle\boldsymbol{w}^{t+1},\boldsymbol{x}_i\rangle)<\lambda\big]\tag{5.19}$$

5.5 Experiment

In this section, the proposed *SPRL* algorithm is evaluated on synthetic and real-world datasets in robust regression and classification tasks. After the experiment setup has been introduced in Section 5.5.1, we present results on the robust regression task compared against several existing methods on both synthetic and real-world datasets in Section 5.5.2, along with performance results for the binary classification task in Section 5.5.3. The analysis on parameter λ is presented in Section 5.5.4. 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

of memory. Details of both the source code and datasets used in the experiments can be downloaded here[2].

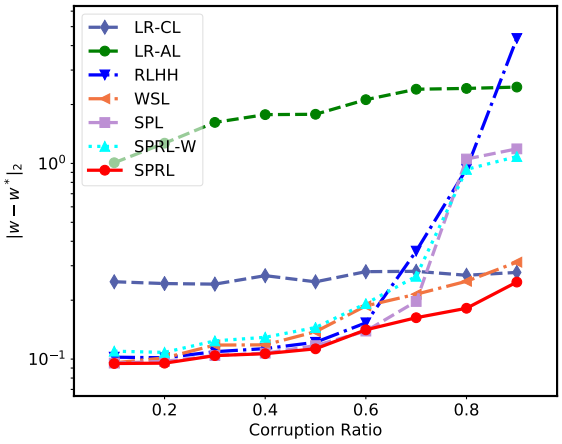
5.5.1 Experiment Setup

Datasets and Labels

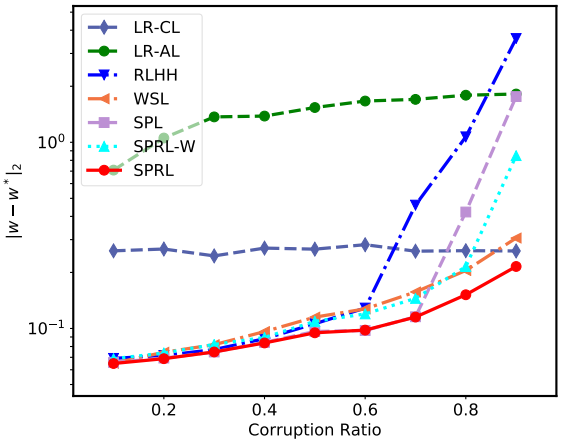
Our datasets consist of synthetic and real-world data. The simulation samples for linear regression were randomly generated according to the model $\boldsymbol{y} = X^T\boldsymbol{w}^* + \boldsymbol{u} + \boldsymbol{\varepsilon}$, where $\boldsymbol{w}_*$ is the ground truth coefficients and $\boldsymbol{u}$ is the adversarial corruption vector. $\boldsymbol{\varepsilon}$ represents the additive dense noise, where $\varepsilon_j \sim \mathcal{N}(0, \sigma^2)$. We sampled the ground-truth regression coefficients $\boldsymbol{w}_* \in \mathbb{R}^p$ as a random unit norm vector. The covariance data X was drawn independently and identically distributed from $\boldsymbol{x}_i \sim \mathcal{N}(0, I_p)$ where I_p is identity matrix, and the uncorrupted response variables were generated as $\boldsymbol{y}_* = X^T\boldsymbol{w}_* + \boldsymbol{\varepsilon}$. The corrupted response vector was generated as $\boldsymbol{y} = \boldsymbol{y}_* + \boldsymbol{u}$, where the corruption vector $\boldsymbol{u}$ was sampled from the uniform distribution $[-5\|\boldsymbol{y}_*\|_\infty, 5\|\boldsymbol{y}_*\|_\infty]$. The set of uncorrupted points was selected as a uniformly random subset of $[n]$. Similarly, the authentic samples for the binary classification task were generated according to the model $\boldsymbol{y}_* = \text{sign}(X^T\boldsymbol{w}_* + \boldsymbol{\varepsilon})$, where $\boldsymbol{\varepsilon}$ is the additive dense noise. The labels of corrupted samples were set as the opposite values in $\boldsymbol{y}_*$.

For the real-world datasets, we chose the *BlogFeedback* dataset [13] for the robust regression task to predict the number of blog comments in the upcoming 24 hours. The data originates from blog posts with 280 feature attributes, including the total number of comments before base time and the number of comments in the first 24 hours after the publication of the blog post. We chose the first 21,000 data samples as the training set, in which the clean set contains 1,000 samples and the noisy set contains the remaining 20,000 samples. The additional data corruption was sampled from the uniform distribution $[-0.5y_i, 0.5y_i]$, where y_i denotes the blog feedback number of the i^{th} sample data. For the classification task, we chose the Large Movie Review dataset [67] collected from the IMDb website[3] for sentiment classification of movie reviews. The first 12,000 data samples were used as training set, of which 2,000 samples were used as the clean set and the remaining samples as the noisy set. Additional data corruption was also added into the noisy set by reversing their labels. The other 10,000 data samples were chosen as the testing set. The features are represented as the tf-idf value of each word. One thousand features with the largest tf-idf values were chosen in our experiment.

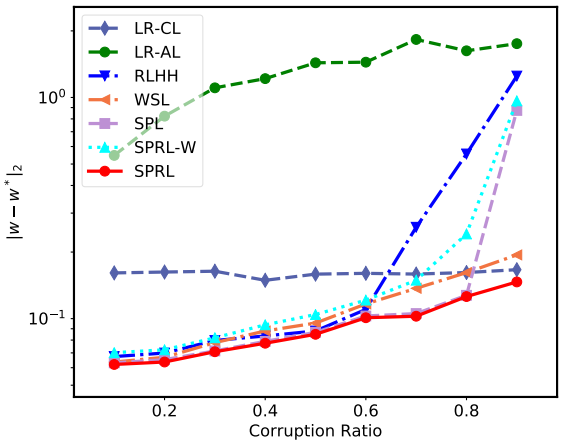
[2] `https://goo.gl/p7eKmZ`
[3] `http://www.imdb.com`



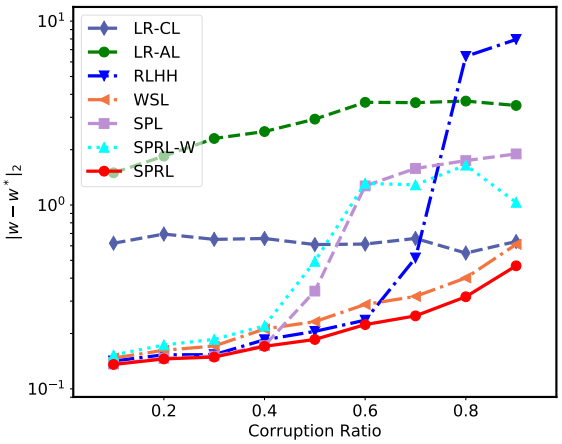
(a) p=400, k=1K, n=6K, dense noise



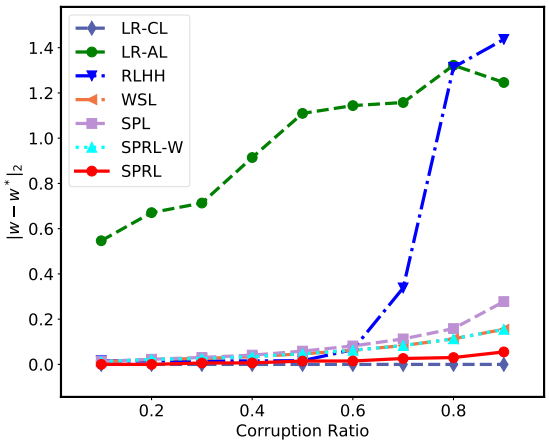
(b) p=400, k=1K, n=11K, dense noise



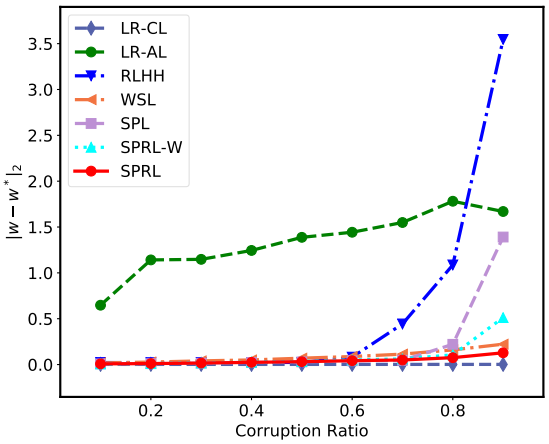
(c) p=400, k=2K, n=12K, dense noise



(d) p=800, k=1K, n=6K, dense noise



(e) p=200, k=1K, n=11K, no dense noise



(f) p=400, k=1K, n=11K, no dense noise

Figure 5.1: Performance on Regression Coefficient Recovery for Different Corruption Ratios in Uniform Distribution.

Table 5.2: Mean Absolute Error of Blog Feedback Prediction

	Corruption Ratio					
	10%	**30%**	**50%**	**70%**	**90%**	**Avg.**
LR-CL	1.159	1.161	1.153	1.164	1.173	1.162
LR-AL	7.254	17.116	10.459	17.226	8.334	12.0778
WSL	0.981	1.280	2.562	2.154	1.375	1.6704
SPL	0.973	1.189	3.666	4.382	4.525	2.947
SPRL-W	**0.919**	2.627	2.493	4.547	5.797	3.2766
SPRL	0.971	**1.107**	**1.036**	**1.053**	**1.046**	**1.0426**

Evaluation Metrics

For the robust regression task, we measured the performance of the regression coefficients recovery using the L_2 error $e = \|\hat{\boldsymbol{w}} - \boldsymbol{w}^*\|_2$ in the synthetic data, where $\hat{\boldsymbol{w}}$ represents the estimated coefficients for each compared method and $\boldsymbol{w}_*$ is the ground truth regression coefficients. For the BlogFeedback dataset, we use the root-mean-squared-error (*RMSE*) to evaluate the performance of blog feedback prediction. Defining $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}_*$ as the predicted feedback number and ground truth number, respectively, the root-mean-squared-error can be presented as $\text{RMSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}_*) = \frac{1}{n}\|\hat{\boldsymbol{y}} - \boldsymbol{y}_*\|_2$, where n is the number of samples. To validate the performance for binary classification, precision, recall, and F1-score are measured by comparing the estimated labels with the actual ones in both the synthetic and real-world datasets.

Comparison Methods

We use the following methods to evaluate the performance of our algorithm in the robust regression tasks: 1) The *Linear Regression on Clean Data* (*LR-CL*) method takes the linear regression on the clean data only without using the noisy data. 2) The *Linear Regression on All Data* (*LR-AL*) method takes the linear regression on all the data without using the prior knowledge of the clean labels. 3) The method *RLHH* [122] applies a recently proposed heuristic hard-thresholding based robust method, in which the entire dataset was directly applied without utilizing the prior knowledge of clean labels. 4) A variant of the weakly supervised learning method [45] (*WSL*) is applied to estimate the uncorrupted samples of the noisy set under the supervision of clean labels by the feedback of the loss function. The method can also be regarded as a non-paced version of *SPRL*. 5) The traditional self-paced learning algorithm (*SPL*) [53] was also compared in our experiment with the parameter $\lambda = 0.1$ and the step size $\mu = 1.1$. 6) *SPRL-W* is a variant of our proposed method *SPRL* without using the clean data set in its objective function. In particular, the first term in Equation (5.4) is omitted. To evaluate the performance in the binary classification

Table 5.3: Performance on Binary Classification (F1, Precision, Recall)

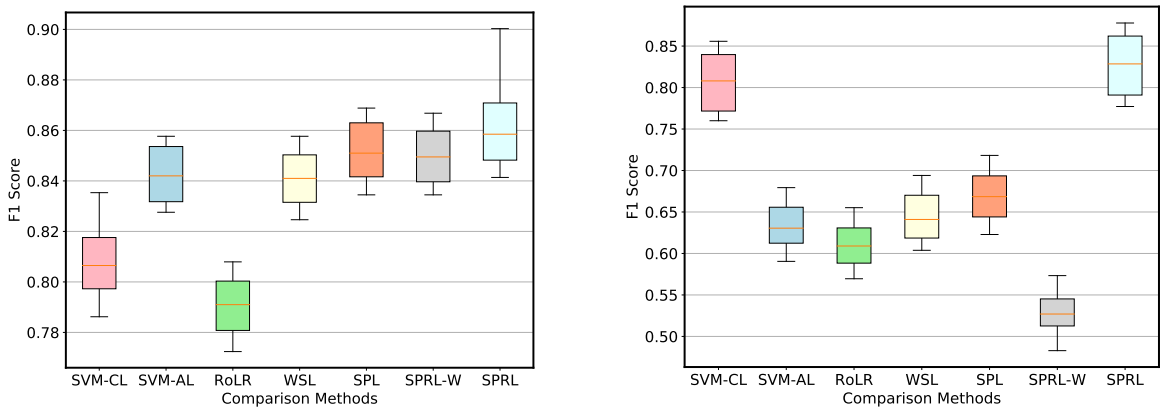
	feature=200, clean set=100, noisy set=5K				**feature=400, clean set=100, noisy set=5K**			
	10%	20%	30%	40%	10%	20%	30%	40%
SVM-CL	0.657,0.656,0.659	0.654,0.650,0.658	0.676,0.667,0.686	0.674,0.688,0.661	0.628,0.629,0.628	0.639,0.640,0.638	0.626,0.621,0.630	0.620,0.627,0.613
SVM-AL	0.928,0.927,0.929	0.900,0.902,0.898	0.835,0.831,0.838	0.750,**0.754**,0.747	0.918,0.916,0.920	0.860,0.861,0.859	**0.786**,**0.796**,0.776	0.665,0.670,0.661
RoLR	0.814,0.817,0.810	0.842,0.840,0.845	0.804,0.795,0.814	0.724,0.730,0.719	0.827,0.834,0.820	0.788,0.790,0.785	0.747,0.758,0.736	0.650,0.659,0.641
WSL	0.886,0.889,0.883	0.791,0.792,0.789	0.745,0.739,0.752	0.706,0.715,0.697	0.870,0.873,0.868	0.786,0.789,0.783	0.690,0.690,0.690	0.644,0.653,0.635
SPL	0.946,0.946,0.946	0.903,0.905,0.902	0.809,0.805,0.813	0.665,0.666,0.665	0.916,0.921,0.912	0.824,0.822,0.826	0.739,0.744,0.735	0.608,0.614,0.602
SPRL-W	0.944,0.942,0.946	0.913,0.916,0.910	0.799,0.796,0.802	0.694,0.699,0.689	0.905,0.903,0.906	0.811,0.815,0.808	0.754,0.760,0.749	0.637,0.647,0.628
SPRL	**0.968**,**0.965**,**0.971**	**0.922**,**0.918**,**0.928**	**0.871**,**0.874**,**0.866**	**0.751**,0.742,**0.754**	**0.935**,**0.936**,**0.932**	**0.864**,**0.863**,**0.865**	0.780,0.785,**0.783**	**0.681**,**0.691**,**0.674**
	feature=200, clean set=200, noisy set=5K				**feature=200, clean set=200, noisy set=10K**			
	10%	20%	30%	40%	10%	20%	30%	40%
SVM-CL	0.758,0.756,0.759	0.722,0.720,0.725	0.734,0.730,0.739	0.734,0.738,0.730	0.715,0.718,0.712	0.730,0.734,0.726	0.732,0.728,0.736	0.701,0.697,0.705
SVM-AL	0.942,0.939,0.944	0.897,0.891,0.904	0.853,0.846,0.861	0.749,0.743,0.756	0.948,0.946,0.950	0.932,0.934,0.930	0.898,0.899,0.897	0.787,0.790,0.784
RoLR	0.833,0.833,0.834	0.834,0.834,0.834	0.808,0.806,0.811	0.699,0.693,0.705	0.879,0.877,0.882	0.886,0.884,0.888	0.665,0.668,0.662	0.771,0.770,0.771
WSL	0.905,0.899,0.911	0.827,0.825,0.829	0.796,0.794,0.798	0.743,0.747,0.740	0.902,0.900,0.904	0.856,0.861,0.851	0.798,0.801,0.795	0.722,0.718,0.727
SPL	0.950,0.951,0.949	0.905,0.899,0.912	0.810,0.810,0.810	0.665,0.665,0.665	0.967,0.965,0.969	0.959,0.963,0.954	0.869,0.875,0.864	0.687,0.689,0.686
SPRL-W	0.949,0.949,0.949	0.896,0.892,0.900	0.822,0.823,0.821	0.745,0.736,0.755	0.966,0.964,0.969	0.950,0.953,0.946	0.902,0.903,0.900	0.721,0.722,0.721
SPRL	**0.963**,**0.967**,**0.960**	**0.926**,**0.925**,**0.927**	**0.876**,**0.878**,**0.875**	**0.768**,**0.780**,**0.763**	**0.981**,**0.977**,**0.985**	**0.959**,**0.954**,**0.963**	**0.920**,**0.928**,**0.912**	**0.787**,**0.782**,**0.793**

task, we applied the following competing methods: 1) The *SVM on Clean Data* (*SVM-CL*) method uses the binary support vector machine on the clean data only, without using the noisy data. 2) The *SVM on All Data* (*SVM-AL*) method takes the support vector machine method on the entire dataset without using the prior knowledge of the clean dataset. 3) A recently proposed robust logistic regression algorithm, called *RoLR* [34], which estimates the parameter through a linear programming procedure was also compared. 4) A similar *WSL* method using the *SVM* loss function was also evaluated in the robust classification task. 5) The self-paced learning algorithm (*SPL*) [53] was also compared in the robust classification task with the initial parameter $\lambda = 0.1$ and the step size $\mu = 1.1$. 6) *SPRL-W* using the classification loss function was also compared with the same setting as the *SPL* method. For our proposed method, *SPRL*, we choose the parameter λ_∞ as 1 and 3.5 for regression and classification tasks, respectively. All the results from comparison methods were averaged over 10 runs.

5.5.2 Performance on Robust Regression

Regression Coefficient Recovery on Synthetic Data

Figure 7.2 shows coefficient recovery performance for different settings on corruption ratios and data sizes. Specifically, Figure 7.2(a) and Figure 7.2(b) show the recovery performance for different amounts of noisy data with the same amounts of features and clean data. Looking at the results, we can conclude: 1) The *SPRL* methed outperforms all the competing methods, including the traditional self-paced learning method with the same parameter settings. 2) *LR-CL* has stable performance because the clean data is not affected by the change of corruption ratio. However, if a model trained in clean data with a small data size only, the performance is almost two times worse than *SPRL*. When the corruption ratio is larger than 80%, it outperforms the other competing methods except for *SPRL* because



(a) Clean set = 2K, Noisy set = 10K, Corruption Ratio = 10%

(b) Clean set = 2K, Noisy set = 10K, Corruption Ratio = 50%

Figure 5.2: Sentiment Classification Performance on Movie Reviews

little uncorrupted data is contained in the noisy set. 3) *LR-AL* always performs worse than the other methods since it does not consider the data corruption and the prior knowledge on the clean set. 4) *RLHH* has a competitive performance when the data corruption is less than 50%, however, when the corruption ratio is over 60%, the recovery error increases dramatically. 5) The recovery errors of *SPL* and *SPRL-W* also dramatically increased when the corruption ratio was over 60% because these two methods do not properly utilize the prior knowledge of the clean set. 6) The *WSL* method performs around 10% worse than *SPRL*, which shows the self-paced training process can improve performance in the robust learning problem. When the size of the clean set increases, Figure 7.2(c) shows similar results as Figure 7.2(a) with a decreased overall recovery error. When the number of features increases in Figure 7.2(d), the overall recovery error increased around 200% on average for all the methods. Figures 7.2(e) and 7.2(f) show the performance in a no-dense-noise setting. Since *LR-CL* has no impact on different corruption ratios, it can always recover the ground truth coefficient exactly. However, *SPRL* can still outperform the other methods in different corruption ratios, achieving a close recovery of the ground truth coefficient.

Blog Feedback Prediction

Table 5.2 shows the results of blog feedback prediction in different corruption settings. Since the *RLHH* method cannot perfectly handle data corruption greater than 40%, its result is not shown in Table 5.2. From the results, we can conclude: 1) *SPRL* consistently outperforms the other competing methods except when the corruption ratio is 10%, in which case most of the methods have extremely close results. 2) The error of the *SPL* and *SPRL-W* methods increase dramatically when the corruption ratio is raised. However, *SPRL* has a consistent

performance with little impact of the corruption ratio. 3) *WSL* has competitive results, but its error is still around 40% larger than our proposed method. 4) *LR-CL* method has a good performance when the data is extremely noisy. However, our proposed method still gets 11.9% improvement when combining both clean and noisy data. 5) *LR-AL* performs the worst since it is simply applied in the entire dataset without considering the data corruption.

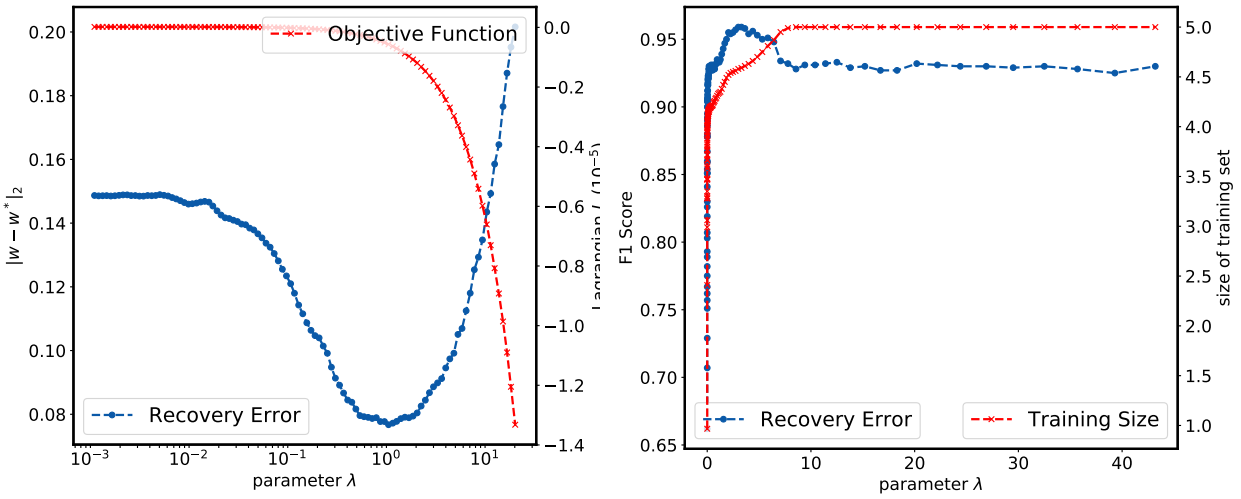
5.5.3 Performance on Robust Classification

Binary Classification on Synthetic Data

Table 5.3 shows the results on the robust binary classification task on different settings of features and data sizes. From the results, we conclude: 1) The *SPRL* method outperforms all the competing methods consistently in different settings, including *RoLR*, whose corruption ratio parameter uses the ground truth value. 2) *SVM-CL* trained on the clean set performs worse than the other methods, which shows that utilizing the large-scale noisy data is important when the size of clean set is small. 3) *SVM-AL* has a competitive performance when the size of the clean set is not large enough for training. But our proposed method, *SPRL*, can still perform better than *SVM-AL* even when 40% of the data labels are corrupted. 4) The *SPRL-W* method usually performs better than *SPL* when the corruption ratio is larger than 20%. This is because *SPRL-W* utilizes the prior knowledge on the clean set, which greatly helps when the corruption ratio increases.

Sentiment Classification of Movie Reviews

The performance on sentiment classification of IMDb movie reviews is shown in Figure 5.2. We use box plots to show the distribution of results based on the summary of five numbers: minimum, first quartile, median, third quartile, and maximum. The results are run on 20 independent datasets sampled from the IMDb dataset with different corruption ratios. From the results, we can conclude: 1) *SPRL* outperforms the other competing methods in different settings of corruption ratio. 2) When the corruption ratio is small, the performance of *SVM-CL* is worse than all the other methods. However, when the corruption ratio is increased, its F1 scores are better than all the other methods except *SPRL* which has a consistent performance. 3) When the corruption ratio increases, the performance of *SPRL-W* degrades dramatically since the clean set is not included in its training set. 4) Even when the corruption ratio is 50%, *SPRL* still performs better than *SVM-CL* because it can utilize the uncorrupted data in the noisy data to improve its performance of the training process.



(a) Linear Regression (b) Binary Classification

Figure 5.3: Impact of Parameter λ on Robust Regression and Classification Tasks

5.5.4 Analysis of Parameter λ

Figure 5.3 shows the impact of parameter λ on both the robust regression and classification tasks. In Figure 5.3(a), the blue line depicts the relationship between parameter λ and the coefficient recovery error. As λ increases, the recovery error continues to decrease until it reaches a critical point, after which it increases. These results indicate that the training process will keep improving the model until parameter λ becomes so large that some corrupted samples are incorporated into the training data. The red line shows the value of the objective function $\mathcal{J}$ in terms of different values of parameter λ, leading us to conclude: 1) The value of objective function $\mathcal{J}$ monotonically decreases as λ increases. 2) The objective function $\mathcal{J}$ decreases much faster once λ reaches a critical point, following the same pattern as the recovery error shown in the blue line. In Figure 5.3(b), the blue line shows the values of the F1 score for the binary classification task. When λ increases, the F1 score increases quickly until it reaches a peak point. After that point, the score decreases because more corrupted data is incorporated into the training set. The red line shows the size of the training set. We can conclude: 1) When parameter λ increases, the size of the training set continuously increases until it reaches its maximum value. 2) When all the data is included into the training set, the F1 score also remains stable.

5.6 Conclusion

In this paper, a self-paced robust learning algorithm is proposed to leverage clean labels in noisy data. To achieve this, the self-paced learning process selects data instances from clean to noisy under the supervision of the well-labeled data, which helps to hedge the risk of

learning from corrupted data samples. Moreover, theoretical analysis shows our algorithm can be converged when the loss function is lower bounded. Extensive experiments on both synthetic data and real-world data on robust regression and classification tasks demonstrated that the proposed algorithm outperforms the other comparable methods over a range of different data settings.

Chapter 6

Distributed Self-Paced Learning in ADMM

The chapter extends the self-paced robust learning method to a new distributed self-paced learning version via alternating direction method of multipliers (ADMM). The introduction is provided in Section 6.1, then Section 6.2 gives a formal problem formulation. The proposed distributed self-paced learning algorithm is presented in Section 6.3 and Section 6.4 presents a theoretical analysis of the convergence of the proposed method. In Section 6.5, the experimental results are analyzed and the paper concludes with a summary of our work in Section 6.6.

6.1 Introduction

Inspired by the learning processes used by humans and animals [4], Self-Paced Learning (*SPL*) [53] considers training data in a meaningful order, from easy to hard, to facilitate the learning process. Unlike standard curriculum learning [4], which learns the data in a predefined curriculum design based on prior knowledge, *SPL* learns the training data in an order that is dynamically determined by feedback from the individual learner, which means it can be more extensively utilized in practice. In self-paced learning, given a set of training samples along with their labels, a parameter λ is used to represents the "age" of the *SPL* in order to control the learning pace. When λ is small, "easy" samples with small losses are considered. As λ grows, "harder" samples with larger losses are gradually added to the training set. This type of learning process is modeled on the way human education and cognition functions. For instance, students will start by learning easier concepts (e.g., Linear Equations) before moving on to more complex ones (e.g., Differential Equations) in the mathematics curriculum. Self-paced learning can also be finely explained in a robust learning manner, where uncorrupted data samples are likely to be used for training earlier

in the process than corrupted data.

In recent years, self-paced learning [53] has received widespread attention for various applications in machine learning, such as image classification [44], event detection [42, 124] and object tracking [104, 118]. A wide assortment of *SPL*-based methods [64, 88] have been developed, including self-paced curriculum learning [44], self-paced learning with diversity [43], multi-view [112] and multi-task [50, 58] self-paced learning. In addition, several researchers have conducted theoretical analyses of self-paced learning. [76] provides a theoretical analysis of the robustness of *SPL*, revealing that the implicit objective function of *SPL* has a similar configuration to a non-convex regularized penalty. Such regularization restricts the contributions of noisy examples to the objective, and thus enhances the learning robustness. [65] proved that the *SPL* algorithm can always converge to critical points of its implicit objective under mild conditions, while [32] studied self-paced implicit regularizers, named self-paced implicit regularizers that are derived from convex conjugacy.

Existing self-paced learning approaches typically focus on modeling the entire dataset at once; however, this may introduce a bottleneck in terms of memory and computation, as today's fast-growing datasets are becoming too large to be handled integrally. For those seeking to address this issue, the main challenges can be summarized as follows: 1) *Computational infeasibility of handling the entire dataset at once.* Traditional self-paced learning approaches gradually grow the training set according to their learning pace. However, this strategy fails when the training set grows too large to be handled due to the limited capacity of the physical machines. Therefore, a scalable algorithm is required to extend the existing self-paced learning algorithm for massive datasets. 2) *Existence of heterogeneously distributed "easy" data.* Due to the unpredictability of data distributions, the "easy" data samples can be arbitrarily distributed across the whole dataset. Considering the entire dataset as a combination of multiple batches, some batches may contain large amount of "hard" samples. Thus, simply applying self-paced learning to each batch and averaging across the trained models is not an ideal approach, as some models will only focus on the "hard" samples and thus degrade the overall performance. 3) *Dependency decoupling across different data batches.* In self-paced learning, the instance weights depend on the model trained on the entire dataset. Also, the trained model depends on the weights assigned to each data instance. If we consider each data batch independently, a model trained in a "hard" batch can mistakenly mark some "hard" samples as "easy" ones. For example, in robust learning, the corrupted data samples are typically considered as "hard" samples, then more corrupted samples will therefore tend to be involved into the training process when we train data batches independently.

In order to simultaneously address all these technical challenges, this paper presents a novel Distributed Self-Paced Learning (*DSPL*) algorithm. The main contributions of this paper can be summarized as follows: 1) We reformulate the self-paced problem into a distributed setting. Specifically, an auxiliary variable is introduced to decouple the dependency of the model parameters for each data batch. 2) A distributed self-paced learning algorithm based on consensus ADMM is proposed to solve the *SPL* problem in a distributed setting. The algorithm optimizes the model parameters for each batch in parallel and consolidates their

values in each iteration. 3) A theoretical analysis is provided for the convergence of our proposed *DSPL* algorithm. The proof shows that our new algorithm will converge under mild assumptions, e.g., the loss function can be non-convex. 4) Extensive experiments have been conducted utilizing both synthetic and real-world data based on a robust regression task. The results demonstrate that the proposed approaches consistently outperform existing methods for multiple data settings. To the best of our knowledge, this is the first work to extend self-paced learning to a distributed setting, making it possible to handle large-scale datasets.

6.2 Problem Formulation

In the context of distributed self-paced learning, we consider the samples to be provided in a sequence of mini batches as $\{(X^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (X^{(m)}, \boldsymbol{y}^{(m)})\}$, where $X^{(i)} \in \mathcal{R}^{p \times n_i}$ represents the sample data for the i^{th} batch, $\boldsymbol{y}^{(i)}$ is the corresponding response vector, and n_i is the instance number of the i^{th} batch.

The goal of self-paced learning problem is to infer the model parameter $\boldsymbol{w} \in \mathcal{R}^p$ for the entire dataset, which can be formally defined as follows:

$$\begin{aligned} &\arg\min_{\boldsymbol{w},\boldsymbol{v}} \sum_{i=1}^{m} f_i(\boldsymbol{w}, \boldsymbol{v}_i) + \|\boldsymbol{w}\|_2^2 \\ s.t. \quad &v_{ij} \in [0,1], \quad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n_i \end{aligned} \tag{6.1}$$

where $\|\boldsymbol{w}\|_2^2$ is the regularization term for model parameters $\boldsymbol{w}$. Variable $\boldsymbol{v}_i$ represents the instance weight vector for the i^{th} batch and v_{ij} is the weight of the j^{th} instance in the i^{th} batch. The objective function $f_i(\boldsymbol{w}, \boldsymbol{v}_i)$ for each mini-batch is defined as follows:

$$f_i(\boldsymbol{w}, \boldsymbol{v}_i) = \sum_{j=1}^{n_i} v_{ij} \mathcal{L}(y_{ij}, g(\boldsymbol{w}, \boldsymbol{x}_{ij})) - \lambda \sum_{j=1}^{n_i} v_{ij} \tag{6.2}$$

We denote $\boldsymbol{x}_{ij} \in \mathcal{R}^p$ and $y_{ij} \in \mathcal{R}$ as the feature vector and its corresponding label for the j^{th} instance in the i^{th} batch. The loss function $\mathcal{L}$ is used to measure the error between label y_{ij} and the estimated value from model g. The term $-\lambda \sum_{j=1}^{n_i} v_{ij}$ is the regularization term for instance weights $\boldsymbol{v}_i$, where parameter λ controls the learning pace. The notations used in this paper are summarized in Table 7.2.

The problem defined above is very challenging in the following three aspects. First, data instances for all m batches can be too large to be handled simultaneously in their entirety, thus requiring the use of a scalable algorithm for large datasets. Second, the instance weight variable $\boldsymbol{v}_i$ of each batch depends on the optimization result for $\boldsymbol{w}$ shared by all the data, which means all the batches are inter-dependent and it is thus not feasible to run them in parallel. Third, the objective function of variables $\boldsymbol{w}_i$ and $\boldsymbol{v}_i$ are jointly non-convex and it is

Notations	Explanations
p	feature number in data matrix $X^{(i)}$
n_i	instance number in the i^{th} data batch
$X^{(i)}$	data matrix of the i^{th} batch
$\boldsymbol{y}^{(i)}$	the response vector of the i^{th} batch
$\boldsymbol{w}$	model parameter of the entire dataset
$\boldsymbol{v}_i$	instance weight vector of the i^{th} batch
v_{ij}	weight of the j^{th} instance in the i^{th} batch
λ	parameter to control the learning pace
$\mathcal{L}$	loss function of estimated model

Table 6.1: Mathematical Notations

an NP-hard problem to retrieve the global optimal solution [35]. In next section, we present a distributed self-paced learning algorithm based on consensus ADMM to address all these challenges.

6.3 The Proposed Methodology

In this section, we propose a distributed self-paced learning algorithm based on the alternating direction method of multipliers (ADMM) to solve the problem defined in Section 6.2.

The problem defined in Equation (6.1) cannot be solved in parallel because the model parameter $\boldsymbol{w}$ is shared in each batch and the result of $\boldsymbol{w}$ will impact on the instance weight variable $\boldsymbol{v}_i$ for each batch. In order to decouple the relationships among all the batches, we introduce different model parameters $\boldsymbol{w}_i$ for each batch and use an auxiliary variable $\boldsymbol{z}$ to ensure the uniformity of all the model parameters. The problem can now be reformulated as follows:

$$\begin{gathered} \arg\min_{\boldsymbol{w}_i, \boldsymbol{v}_i, \boldsymbol{z}} \sum_{i=1}^{m} f_i(\boldsymbol{w}_i, \boldsymbol{v}_i; \lambda) + \|\boldsymbol{z}\|_2^2 \\ s.t. \quad v_{ij} \in [0,1], \quad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n_i \\ \boldsymbol{w}_i - \boldsymbol{z} = 0, \quad \forall i = 1, \ldots, m \end{gathered} \tag{6.3}$$

where the function $f_i(\boldsymbol{w}_i, \boldsymbol{v}_i)$ is defined as follows.

$$f_i(\boldsymbol{w}_i, \boldsymbol{v}_i; \lambda) = \sum_{j=1}^{n_i} v_{ij} \boldsymbol{\mathcal{L}}(y_{ij}, g(\boldsymbol{w}_i, \boldsymbol{x}_{ij})) - \lambda \sum_{j=1}^{n_i} v_{ij} \tag{6.4}$$

Unlike the original problem defined in Equation (6.1), here each batch has its own model parameter $\boldsymbol{w}_i$ and the constraint $\boldsymbol{w}_i - \boldsymbol{z} = 0$ for $\forall i = 1, \ldots, m$ ensures the model parameter

$\boldsymbol{w}_i$ has the same value as the auxiliary variable $\boldsymbol{z}$. The purpose of the problem reformulation is to optimize the model parameters $\boldsymbol{w}_i$ in parallel for each batch. It is important to note that the reformulation is *tight* because our new problem is equivalent to the original problem when the constraint is strictly satisfied.

In the new problem, Equation (6.3) is a bi-convex optimization problem over $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$ for each batch with fixed $\boldsymbol{z}$, which can be efficiently solved using the Alternate Convex Search (ACS) method [35]. With the variable $\boldsymbol{v}$ fixed, the remaining variables $\{\boldsymbol{w}_i\}$, $\boldsymbol{z}$ and $\boldsymbol{\alpha}$ can be solved by consensus ADMM [6]. As the problem is an NP-hard problem, in which the global optimum requires polynomial time complexity, we propose an alternating algorithm *DSPL* based on ADMM to handle the problem efficiently.

The augmented Lagrangian format of optimization in Equation (6.3) can be represented as follows:

$$\begin{aligned} L = \sum_{i=1}^{m} f_i(\boldsymbol{w}_i, \boldsymbol{v}_i; \lambda) + \|\boldsymbol{z}\|_2^2 + \sum_{i=1}^{m} \boldsymbol{\alpha}_i^T(\boldsymbol{w}_i - \boldsymbol{z}) \\ + \frac{\rho}{2} \sum_{i=1}^{m} \|\boldsymbol{w}_i - \boldsymbol{z}\|_2^2 \end{aligned} \tag{6.5}$$

where $\{\alpha_i\}_{i=1}^m$ are the Lagrangian multipliers and ρ is the step size of the dual step.

The process used to update model parameter $\boldsymbol{w}_i$ for the i^{th} batch with the other variables fixed is as follows:

$$\begin{aligned} \boldsymbol{w}_i^{k+1} = \arg\min_{\boldsymbol{w}_i} f_i(\boldsymbol{w}_i, \boldsymbol{v}_i; \lambda) + [\boldsymbol{\alpha}_i^k]^T(\boldsymbol{w}_i - \boldsymbol{z}^k) \\ + \frac{\rho}{2} \|\boldsymbol{w}_i - \boldsymbol{z}^k\|_2^2 \end{aligned} \tag{6.6}$$

Specifically, if we choose the loss function $\mathcal{L}$ to be a squared loss and model $g(\boldsymbol{w}, \boldsymbol{x}_{ij})$ to be a linear regression $g(\boldsymbol{w}, \boldsymbol{x}_{ij}) = \boldsymbol{w}^T \boldsymbol{x}_{ij}$, we have the following analytical solution for $\boldsymbol{w}_i$:

$$\begin{aligned} \boldsymbol{w}_i^{k+1} = &\left(2 \sum_{j=1}^{n_i} \boldsymbol{v}_{ij} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T + \rho \cdot I \right)^{-1} \\ &\cdot \left(2 \sum_{j=1}^{n_i} \boldsymbol{v}_{ij} \boldsymbol{x}_{ij} y_{ij} - \boldsymbol{\alpha}_i^k + \rho \boldsymbol{z}^k \right) \end{aligned} \tag{6.7}$$

The auxiliary variable $\boldsymbol{z}$ and Lagrangian multipliers $\boldsymbol{\alpha}_i$ can be updated as follows:

$$\begin{aligned} \boldsymbol{z}^{k+1} &= \frac{\rho}{2 + \rho m} \sum_{i=1}^{m} (\boldsymbol{w}_i^{k+1} + \frac{1}{\rho} \boldsymbol{\alpha}_i^k) \\ \boldsymbol{\alpha}_i^{k+1} &= \boldsymbol{\alpha}_i^k + \rho(\boldsymbol{w}_i^{k+1} - \boldsymbol{z}^{k+1}) \end{aligned} \tag{6.8}$$

The stop condition of consensus ADMM is determined by the (squared) norm of the primal and dual residuals of the k^{th} iteration, which are calculated as follows:

$$\begin{aligned} \|r_k\|_2^2 &= \sum_{i=1}^{m} \|\boldsymbol{w}_i^k - \boldsymbol{z}^k\|_2^2 \\ \|s_k\|_2^2 &= m\rho^2 \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|_2^2 \end{aligned} \tag{6.9}$$

After the weight parameter $\boldsymbol{w}_i$ for each batch has been updated, the instance weight vector $\boldsymbol{v}_i$ for each batch will be updated based on the fixed $\boldsymbol{w}_i$ by solving the following problem:

$$\boldsymbol{v}_i^{t+1} = \arg\min_{\boldsymbol{v}_i} \sum_{j=1}^{n_i} v_{ij} \mathcal{L}(y_{ij}, g(\boldsymbol{w}_i^{t+1}, \boldsymbol{x}_{ij})) - \lambda \sum_{j=1}^{n_i} v_{ij} \tag{6.10}$$

For the above problem in Equation (6.10), we always obtain the following closed-form solution:

$$\boldsymbol{v}_i^{t+1} = \infty \Big(\mathcal{L}\big(y_{ij}, g(\boldsymbol{w}_i^{t+1}, \boldsymbol{x}_{ij})\big) < \lambda \Big) \tag{6.11}$$

where $\infty(\cdot)$ is the indicator function whose value equals to one when the condition $\mathcal{L}\big(y_{ij}, g(\boldsymbol{w}_i^{t+1}, \boldsymbol{x}_{ij})\big) < \lambda$ is satisfied; otherwise, its value is zero.

The details of algorithm *DSPL* are presented in Algorithm 6. In Lines 1-2, the variables and parameters are initialized. With the variables $\boldsymbol{v}_i$ fixed, the other variables are optimized in Lines 5-13 based on the result of consensus ADMM, in which both the model weights $\boldsymbol{w}_i$ and Lagrangian multipliers $\boldsymbol{\alpha}_i$ can be updated in parallel for each batch. Note that if no closed-form can be found for Equation (6.6), the updating of $\boldsymbol{w}_i$ is the most time-consuming operation in the algorithm. Therefore, updating $\boldsymbol{w}_i$ in parallel can significantly improve the efficiency of the algorithm. The variable $\boldsymbol{v}_i$ for each batch is updated in Line 14, with the variable $\boldsymbol{w}_i$ fixed. In Lines 15-18, the parameter λ is enlarged to include more data instances into the training set. τ_λ is the maximum threshold for λ and μ is the step size. The algorithm will be stopped when the Lagrangian is converged in Line 20. The following two alternative methods can be applied to improve the efficiency of the algorithm: 1) dynamically update the penalty parameter ρ after Line 11. When $r > 10s$, we can update $\rho \leftarrow 2\rho$. When $10r < s$, we have $\rho \leftarrow \rho/2$. 2) Move the update of variable $\boldsymbol{v}_i$ into the consensus ADMM step after Line 9. This ensures that the instance weights are updated every time the model is updated, so that the algorithm quickly converges. However, no theoretical convergence guarantee can be made for the two solutions, although in practice they do always converge.

6.4 Theoretical Analysis

In this section, we will prove the convergence of the proposed algorithm. Before we start to prove the convergence of Algorithm 6, we make the following assumptions regarding our objective function and penalty parameter ρ:

ALGORITHM 6: DSPL ALGORITHM

Input: $X \in \mathcal{R}^{p\times n}$, $\boldsymbol{y} \in \mathcal{R}^n$, $\lambda_0 \in \mathcal{R}$, $\tau_\lambda \in \mathcal{R}$, $\mu \in \mathcal{R}$
Output: solution $\boldsymbol{w}^{(t+1)}$, $\boldsymbol{v}^{(t+1)}$
Initialize $\boldsymbol{w}_i^0 = \mathbf{1}$, $\boldsymbol{v}_i^0 = \mathbf{1}$
Choose $\varepsilon_L > 0$, $\varepsilon_r > 0$, $\varepsilon_s > 0$, $\lambda \leftarrow \lambda_0$, $t \leftarrow 0$
repeat
  $k \leftarrow 0$
  repeat
    $\boldsymbol{z}^{k+1} \leftarrow \frac{1}{m}\sum_{i=1}^m (\boldsymbol{w}_i^{k+1} + \frac{1}{\rho}\boldsymbol{\alpha}_i^k)$
    Update variables $\boldsymbol{w}_i^{k+1}$ in parallel, for $\forall i = 1 \dots m$
      $\boldsymbol{w}_i^{k+1} \leftarrow \arg\min f_i(\boldsymbol{w}_i, \boldsymbol{v}_i) +$
        $[\boldsymbol{\alpha}_i^k]^T(\boldsymbol{w}_i - \boldsymbol{z}^k) + \frac{\rho}{2}\|\boldsymbol{w}_i - \boldsymbol{z}^k\|_2^2$
    Update dual $\boldsymbol{\alpha}_i^{k+1} \leftarrow \boldsymbol{\alpha}_i^k + \rho(\boldsymbol{w}_i^{k+1} - \boldsymbol{z}^{k+1})$ in parallel
    Update primal and dual residuals r^{k+1} and s^{k+1}.
    $k \leftarrow k + 1$
  until $\|r^{k+1}\|_2^2 < \varepsilon_r$ *and* $\|s^{k+1}\|_2^2 < \varepsilon_s$
  $\boldsymbol{v}_i^{t+1} \leftarrow \infty\Big(\mathcal{L}\big(y_{ij}, g(\boldsymbol{w}_i^{t+1}, \boldsymbol{x}_{ij})\big) < \lambda\Big)$, for $\forall i = 1 \dots m$
  if $\lambda < \tau_\lambda$ **then**
    $\lambda \leftarrow \lambda * \mu$
  else
    $\lambda \leftarrow \tau_\lambda$
  $t \leftarrow t + 1$
until $\|L^{t+1} - L^t\|_2 < \varepsilon_L$
return $\boldsymbol{z}^{t+1}$, $\boldsymbol{v}^{t+1}$

Assumption 2 (Gradient Lipchitz Continuity)**.** *There exists a positive constant φ_i for objective function $f_i(\boldsymbol{w}_i)$ of each batch with the following properties:*

$$\begin{gathered}\|\nabla_{\boldsymbol{w}_i} f_i(\boldsymbol{x}_i) - \nabla_{\boldsymbol{w}_i} f_i(\boldsymbol{y}_i)\| \le \varphi_i \|\boldsymbol{x}_i - \boldsymbol{y}_i\|, \\ \forall \boldsymbol{x}_i, \boldsymbol{y}_i, i = 1, \dots, m\end{gathered} \tag{6.12}$$

Assumption 3 (Lower Bound)**.** *The objective function in problem* (6.3) *has a lower bound $\mathcal{B}$ as follows:*

$$\mathcal{B} = \min_{\boldsymbol{w}_i, \boldsymbol{v}_i, \boldsymbol{z}} \left\{ \sum_{i=1}^m f_i(\boldsymbol{w}_i, \boldsymbol{v}_i) + \|\boldsymbol{z}\|_2^2 \right\} > -\infty \tag{6.13}$$

Assumption 4 (Penalty Parameter Constraints)**.** *For $\forall i = 1 \dots m$, the penalty parameter ρ_i for each batch is chosen in accord with the following constraints:*

- *For $\forall i$, the subproblem* (6.6) *of variable $\boldsymbol{w}_i$ is strongly convex with modulus $\gamma_i(\rho_i)$.*
- *For $\forall i$, we have $\rho_i\gamma_i(\rho_i) > 2\varphi_i^2$ and $\rho_i \ge \varphi_i$.*

Note that when ρ_i increases, subproblem (6.6) will be eventually become strongly convex with respect to variable $\boldsymbol{w}_i$. For simplicity, we will choose the same penalty parameter ρ for all the batches with $\rho = \max_i(\rho_i)$. Based on these assumptions, we can draw the following conclusions.

Lemma 8. *Assume the augmented Lagrangian format of optimization problem* (6.3) *satisfies Assumption 1, the augmented Lagrangian L has the following property:*

$$L(\{\boldsymbol{w}_i^{k+1}\}, \boldsymbol{z}^{k+1}, \boldsymbol{\alpha}^{k+1}) \leq L(\{\boldsymbol{w}_i^{k}\}, \boldsymbol{z}^{k}, \boldsymbol{\alpha}^{k}) \tag{6.14}$$

Proof. Since the the objective function $f_i(\boldsymbol{w}_i)$ for each batch is gradient Lipchitz continuous with a positive constant φ_i, the Lagrangian in Equation (6.5) has the following property according to Lemma 2.2 in [40]:

$$\begin{aligned} & L(\{\boldsymbol{w}_i^{k+1}\}, \boldsymbol{z}^{k+1}, \boldsymbol{\alpha}^{k+1}) - L(\{\boldsymbol{w}_i^{k}\}, \boldsymbol{z}^{k}, \boldsymbol{\alpha}^{k}) \\ & \leq \sum_{i=1}^{m} \left(\frac{\varphi_i^2}{\rho} - \frac{\gamma_i(\rho)}{2} \right) \|\boldsymbol{w}_i^{k+1} - \boldsymbol{w}_i^{k}\|_2^2 - \frac{\gamma}{2} \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k}\|_2^2 \\ & \overset{(a)}{\leq} -\frac{\gamma}{2} \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k}\|_2^2 \leq 0 \end{aligned} \tag{6.15}$$

where $\gamma = m\rho > 0$. The inequality (a) follows from Assumption 2, namely that $\rho\gamma_i(\rho) > 2\varphi_i^2$, so we have $\left(\frac{\varphi_i^2}{\rho} - \frac{\gamma_i(\rho)}{2} \right) < 0$. □

Lemma 9. *Assume the augmented Lagrangian of problem* (6.3) *satisfies Assumptions 1-3, the augmented Lagrangian L is lower bounded as follows:*

$$\lim_{k \to \infty} L(\{\boldsymbol{w}_i^{k+1}\}, \boldsymbol{z}^{k+1}, \boldsymbol{\alpha}^{k+1}) \geq \mathcal{B} \tag{6.16}$$

where $\mathcal{B}$ is the lower bound of the objective function in problem (6.3)*.*

The proof details of 9 can be found in the paper's extended version[1].

Theorem 6. *The Algorithm 6 converges when Assumptions 1-3 are all satisfied.*

Proof. In Lemmas 1 and 2, we proved that the Lagrangian is monotonically decreasing and has a lower bound during the iterations of ADMM. Now we will prove that the same

[1] `https://xuczhang.github.io/papers/ijcai18_dspl_extend.pdf`

properties hold for the entire algorithm after updating variable $\boldsymbol{v}$ and parameter λ.

$$
\begin{aligned}
&L(\{\boldsymbol{w}^{t+1}\}, \boldsymbol{v}^{t+1}, \boldsymbol{z}^{t+1}, \boldsymbol{\alpha}^{t+1}; \lambda^{t+1}) \\
&\overset{(a)}{\leq} L(\{\boldsymbol{w}^{t}\}, \boldsymbol{v}^{t+1}, \boldsymbol{z}^{t}, \boldsymbol{\alpha}^{t}; \lambda^{t+1}) \overset{(b)}{\leq} L(\{\boldsymbol{w}^{t}\}, \boldsymbol{v}^{t}, \boldsymbol{z}^{t}, \boldsymbol{\alpha}^{t}; \lambda^{t+1}) \\
&= L(\{\boldsymbol{w}^{t}\}, \boldsymbol{v}^{t}, \boldsymbol{z}^{t}, \boldsymbol{\alpha}^{t}; \lambda^{t}) + (\lambda^{t} - \lambda^{t+1}) \sum_{i=1}^{m} \sum_{j=1}^{n_i} v_{ij}^{t} \\
&\overset{(c)}{\leq} L(\{\boldsymbol{w}^{t}\}, \boldsymbol{v}^{t}, \boldsymbol{z}^{t}, \boldsymbol{\alpha}^{t}; \lambda^{t})
\end{aligned}
$$

Inequality (a) follows Lemma 1 and inequality (b) follows the optimization step in Line 14 in Algorithm 6. Inequality (c) follows from the fact that λ increases monotonically so that $\lambda^t \leq \lambda^{t+1}$. As $L(\{\boldsymbol{w}^{t+1}\}, \boldsymbol{z}^{t+1}, \boldsymbol{\alpha}^{t+1})$ for some constant values of $\boldsymbol{v}$ and λ has a lower bound $\mathcal{B}$, we can easily prove that $L(\{\boldsymbol{w}^{t+1}\}, \boldsymbol{v}^{t+1}, \boldsymbol{z}^{t+1}, \boldsymbol{\alpha}^{t+1}; \lambda^{t+1}) \geq \mathcal{B} + C - \tau_\lambda n$, where C is a constant and n is the size of the entire dataset. Therefore, the Lagrangian L is convergent. According to the stop condition for Algorithm 6, the algorithm converges when the Lagrangian L is converged.

□

6.5 Experimental Results

In this section, the performance of the proposed algorithm *DSPL* is evaluated for both synthetic and real-world datasets in robust regression task. After the experimental setup has been introduced in Section 6.5.1, we present the results for the regression coefficient recovery performance with different settings using synthetic data in Section 6.5.2. Due to the space limitation, the real-world data evaluation based on house rental price prediction is presented in the extended version[2]. All the experiments were performed 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 the datasets used in the experiment can be downloaded here[3].

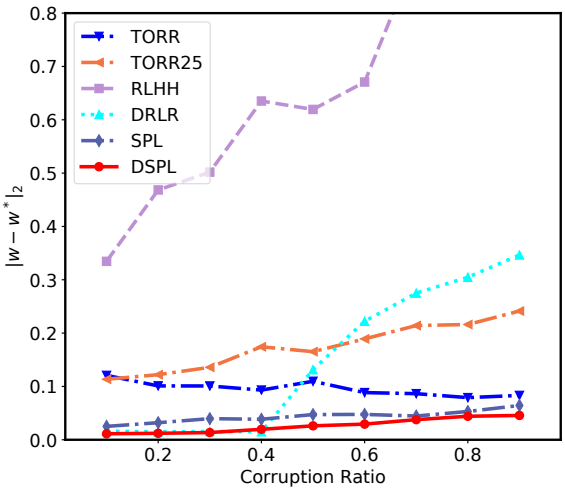
6.5.1 Experimental Setup

Datasets and Labels

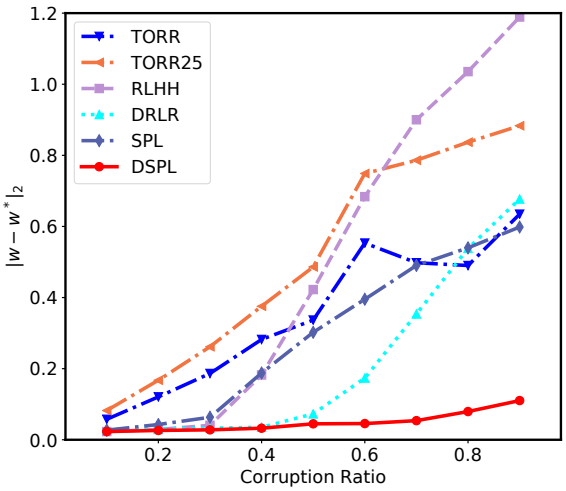
The datasets used for the experimental verification were composed of synthetic and real-world data. The simulation samples were randomly generated according to the model $\boldsymbol{y}^{(i)} =$

[2] `https://xuczhang.github.io/papers/ijcai18_dspl_extend.pdf`

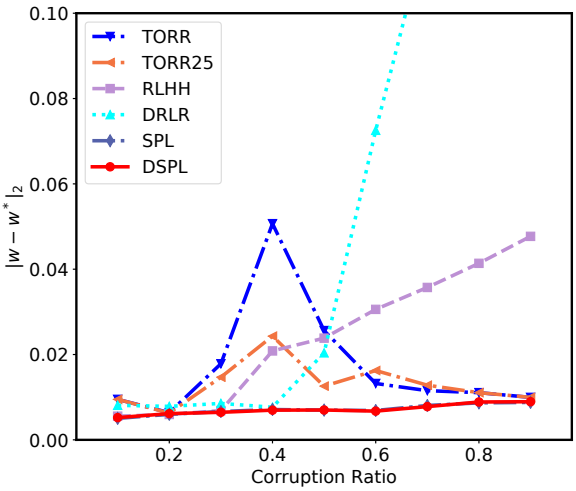
[3] `https://goo.gl/cis7tK`



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



(b) p=400, n=10K, b=10, dense noise



(c) p=100, n=50K, b=10, dense noise

Figure 6.1: Regression coefficient recovery performance for different corruption ratios.

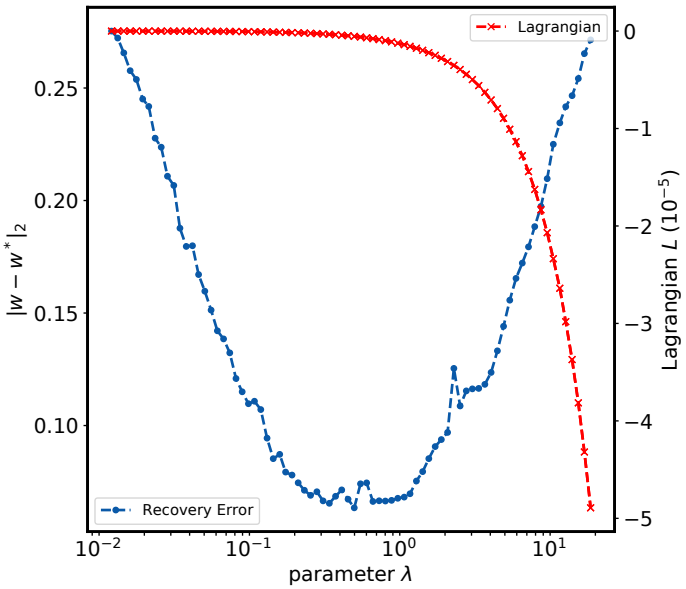


Figure 6.2: Relationship between parameter λ and coefficient recovery error and the corresponding Lagrangian.

$[X^{(i)}]^T \boldsymbol{w}_* + \boldsymbol{u}^{(i)} + \boldsymbol{\varepsilon}^{(i)}$ for each mini-batch, where $\boldsymbol{w}_*$ represents the ground truth coefficients and $\boldsymbol{u}^{(i)}$ the adversarial corruption vector. $\varepsilon^{(i)}$ represents the additive dense noise for the i^{th} batch, where $\varepsilon_j^{(i)} \sim \mathcal{N}(0, \sigma^2)$. We sampled the regression coefficients $\boldsymbol{w}_* \in \mathcal{R}^p$ 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 $\boldsymbol{y}_*^{(i)} = \left[\boldsymbol{X}^{(i)}\right]^T \boldsymbol{w}_* + \boldsymbol{\varepsilon}^{(i)}$. 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 uniform distribution $[-5\|\boldsymbol{y}_*^{(i)}\|_\infty, 5\|\boldsymbol{y}_*^{(i)}\|_\infty]$. The set of uncorrupted points was selected as a uniformly random subset of $[n_i]$ for each batch.

The real-world datasets utilized consisted of house rental transaction data from two cities, *New York City* and *Los Angeles* listed on the Airbnb[4] website from January 2015 to October 2016. These datasets can be publicly downloaded[5]. For the *New York City* dataset, the first 321,530 data samples from January 2015 to December 2015 were used as the training data and the remaining 329,187 samples from January to October 2016 as the test data. For the *Los Angeles* dataset, the first 106,438 samples from May 2015 to May 2016 were used as training data, and the remaining 103,711 samples as test 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.

[4] `https://www.airbnb.com/`

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

	4/10	**5/10**	**6/10**	**7/10**	**8/10**	**9/10**
TORR	0.093	0.109	0.088	0.086	0.079	0.083
TORR25	0.174	0.165	0.189	0.214	0.216	0.241
RLHH	0.635	0.619	0.670	0.907	0.851	0.932
DRLR	**0.014**	0.131	0.222	0.274	0.304	0.346
SPL	0.038	0.047	0.047	0.044	0.053	0.064
DSPL	0.030	**0.034**	**0.039**	**0.036**	**0.041**	**0.045**

Table 6.2: Regression Coefficient Recovery Performance for Different Corrupted Batches

Evaluation Metrics

For the synthetic data, we measured the performance of the regression coefficient recovery using the averaged L_2 error $e = \|\hat{\boldsymbol{w}} - \boldsymbol{w}_*\|_2$, where $\hat{\boldsymbol{w}}$ represents the recovered coefficients for each method and $\boldsymbol{w}_*$ represents the ground truth regression coefficients. For the real-world dataset, we used the mean absolute error (MAE) to evaluate the performance for rental price prediction. Defining $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ as the predicted price and ground truth price, respectively, the mean absolute error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ can be presented as $\text{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right|$, where y_i represents the label of the i^{th} data sample.

Comparison Methods

We used the following methods to compare the performance of the robust regression task: *Torrent* (*Abbr. TORR*) [5], which is a hard-thresholding based method that includes a parameter for the corruption ratio. As this parameter is hard to estimate in practice, we opted to use a variant, *TORR25*, which represents a corruption parameter that is uniformly distributed across a range of $\pm 25\%$ off the true value. We also used *RLHH* [122] for the comparison, which applies a recently proposed robust regression method based on heuristic hard thresholding with no additional parameters. This method computes the regression coefficients for each batch, and averages them all. The *DRLR* [121] algorithm, which is a distributed robust learning method specifically designed to handle large scale data with a distributed robust consolidation. The traditional self-paced learning algorithm (*SPL*) [53] with the parameter $\lambda = 1$ and the step size $\mu = 1.1$ was also compared in our experiment. For *DSPL*, we used the same settings as for *SPL* with the initial $\lambda_0 = 0.1$ and $\tau_\lambda = 1$. All the results from each of these comparison methods were averaged over 10 runs.

6.5.2 Robust Regression in Synthetic Data

Recovery Coefficients Recovery

Figure 7.2 shows the coefficient recovery performance for different corruption ratios in uniform distribution. Specifically, Figures 7.2(a) and 7.2(b) show the results for a different number of features with a fixed data size. Looking at the results, we can conclude: 1) Of the six methods tested, the *DSPL* method outperformed all the competing methods, including *TORR*, whose corruption ratio parameter uses the ground truth value. 2) Although *DRLR* turned in a competitive performance when the data corruption level was low. However, when the corruption ratio rose to over 40%, the recovery error is increased dramatically. 3) The *TORR* method is highly dependent on the corruption ratio parameter. When the parameter is 25% different from the ground truth, the error for *TORR25* was over 50% compared to *TORR*, which uses the ground truth corruption ratio. 4) When the feature number is increased, the average error for the *SPL* algorithm increased by a factor of four. However, the results obtained for the *DSPL* algorithm varied consistently with the corruption ratio and feature number. The results presented in Figures 7.2(a) and 7.2(c) conform that the *DSPL* method consistently outperformed the other methods for larger datasets, while still achieving a close recovery of the ground truth coefficient.

Performance on Different Corrupted Batches

The regression coefficient recovery performance for different numbers of corrupted batches is shown in Table B.1, ranging from four to nine corrupted batches out of the total of 10 batches. Each corrupted batch used in the experiment contains 90% corrupted samples and each uncorrupted batch has 10% corrupted samples. The results are shown for the averaged L_2 error in 10 different synthetic datasets with randomly ordered batches. Looking at the results shown in Table B.1, we can conclude: 1) When the ratio of corrupted batches exceeds 50%, *DSPL* outperforms all the competing methods with a consistent recovery error. 2) *DRLR* performs the best when the mini-batch is 40% corrupted, although its recovery error increases dramatically when the number of corrupted batch increases. 3) *SPL* turns in a competitive performance for different levels of corrupted batches, but its error almost doubles when the number of corrupted batches increases from four to nine.

Analysis of Parameter λ

Figure 6.2 show the relationship between the parameter λ and the coefficient recovery error, along with the corresponding Lagrangian L. This result is based on the robust coefficient recovery task for a 90% data corruption setting. Examining the blue line, as the parameter λ increases, the recovery error continues to decrease until it reaches a critical point, after which it increases. These results indicate that the training process will keep improving the model

until the parameter λ becomes so large that some corrupted samples become incorporated into the training data. In the case shown here, the critical point is around 1.0. The red line shows the value of the Lagrangian L in terms of different values of the parameter λ, leading us to conclude that: 1) the Lagrangian monotonically decreases as λ increases. 2) The Lagrangian decreases much faster once λ reaches a critical point, following the same pattern as the recovery error shown in blue line.

6.6 Conclusion

In this paper, a distributed self-paced learning algorithm (*DSPL*) is proposed to extend the traditional *SPL* algorithm to its distributed version for large scale datasets. To achieve this, we reformulated the original *SPL* problem into a distributed setting and optimized the problem of treating different mini-batches in parallel based on consensus *ADMM*. We also proved that our algorithm can be convergent under mild assumptions. Extensive experiments on both synthetic data and real-world rental price data demonstrated that the proposed algorithms are very effective, outperforming the other comparable methods over a range of different data settings.

Chapter 7

Robust Multi-Factor Personality Prediction with Correlated Noises

The chapter presents a novel robust multi-factor personality prediction model with correlated data corruption. Section 7.1 introduces the work, then Section 7.2 reviews background and related work, and Section 7.3 introduces the problem setup. The proposed RMFP algorithm is presented in Section 7.4. Section 7.5 presents the proof of convergence rate and recovery guarantee. The experiments on both synthetic and real-world datasets are presented in Section 7.6, and the paper concludes with a summary of the research in Section 7.7.

7.1 Introduction

Personality is a combination of attitudinal, emotional, and behavioral response patterns accounting for individual differences in people [106]. The classic Five-Factor model [71] comprises five traits: *Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness*, and *Neuroticism*. Individual personality analysis is important and widely used in many real-world applications with characterized services w [78]. For instance, an *extraversive* user may have a higher frequency of online activities and be more likely to use a recommendation system to make new friends with strangers. To identify an individual's personality, the most commonly used method is self-report inventory [27], which requires individuals to answer questions about their typical behavior. For example, the Berkeley Personality Lab has designed a widely used Big-Five Inventory (BFI) containing 44 questions to form reliable five-factor personality scores. Despite its wide usage, the self-report method still suffers from two major problems: It is particularly difficult to conduct in large scale, and *social desirability* [83] may influence responses in that people might state what they wish, which will reduce the credibility of their responses.

Recently, the growing popularity of social media has provided a new way to manifest per-

Table 7.1: Example of Corrupted Big-Five Personality Score

	Agr.	*Con.*	*Ext.*	*Ope.*	*Neu.*
Corruption-A	4.10 ✓	2.65 ✓	2.25 ✗	3.15 ✓	2.15 ✓
Corruption-B	3.20 ✗	2.50 ✓	4.25 ✓	1.15 ✗	2.05 ✓
Corruption-C	4.20 ✓	4.50 ✗	4.30 ✓	1.45 ✗	4.15 ✗
Corruption-D	1.20 ✗	3.50 ✗	2.25 ✗	1.15 ✗	4.05 ✗
Groud-Truth	**4.20**	**2.50**	**4.25**	**3.15**	**2.05**

sonality through users' online activities, which yields important insights into users' interests, preferences, and sentiments. As online behaviors share a significant amount in common with real-world behaviors [55], social media provides an excellent data source to reproduce social activities in real life from a large and diverse population. Although social media can easily extend the traditional methods to large scale, the personality data collected from social media is more easily corrupted by *careless response* [74] and *social desirability* [83] due to the lack of supervision. For instance, one may distort the response of personality assessment to exaggerate the personality score of *conscientiousness* due to one's social desirability; or randomly respond the questions in personality assessment carelessly. Table 7.1 shows four types of data corruption compared to their ground truth Big-Five personality score. For example, the person in Corruption-C case exaggerates the *conscientiousness* score while the scores of *openness* and *neuroticism* are also impacted. Existing work on personality prediction in social media typically only focuses on building the relation between personality and online behaviors, although careless or malicious user annotations are actually a crucial issue in practice [83]. For those seeking to address this issue, the major challenges can be summarized as follows. 1) **Existence of correlated corruption**. Featured by the characteristics of personality, factors of personality have corruption correlations. For example, when one tends to fake a higher score of *extraversion* in one personality test, the *neuroticism* score will be correspondingly impacted [132]. Thus, simply estimating corruption independently for each factor is not an ideal strategy, as their interactions also need to be considered. 2) **Difficulty in estimating the corruption ratio**. Existing methods typically assume the corruption ratio is a user known parameter; however, the parameter can hardly be estimated in practice even when users obtain the corresponding domain knowledge. Moreover, due to the existence of correlated corruption among the factors, as mentioned earlier, corruption ratio are not independent of each other. It is thus clearly necessary to utilize this correlation pattern to regulate the corruption estimation process. 3) **Scalability to massive datasets**. Different from the traditional inventory-based psychological analysis based on at most hundreds of data samples, millions of samples are being generated by social media users everyday. Therefore, considering the complexity of robust personality prediction problem, an efficient algorithm is required to handle the massive datasets.

To simultaneously address all these technical challenges, this paper presents a novel model

based on multi-factor learning, **Robust Multi-Factor Personality Prediction (RMFP)**, which jointly optimizes the regression coefficients with correlated corruption. In our RMFP model, the uncorrupted sample set for the prediction of each factor will be consolidated into a unified set. This improves the estimation of not only the regression coefficients but also the corruption patterns. Moreover, by letting the factors learn the estimation of corruption from each other, our algorithm achieves faster convergence than optimizing them individually. In addition, each factor can be optimized in parallel, which is extremely beneficial to efficiency when the number of factors becomes large. The main contributions of this paper are as follows: 1) **Formulating a model for robust multi-factor personality prediction.** The proposed model considers correlation of multiple personality factors by utilizing the correlated corruption property. Based on this property, each personality factor learns the estimated corruption pattern from the others to improve overall performance. 2) **Proposing a distributed robust algorithm for the multi-factor regression problem.** The optimization of the proposed multi-factor model is a non-convex discrete optimization problem, which is technically challenging. By optimizing individual factor in parallel, the uncorrupted set is combined from each factor following two strategies: global consensus and majority voting. 3) **Providing a strong recovery guarantee under the multi-factor problem setting.** We prove that our RMFP algorithm with global consensus strategy converges at a geometric rate and recovers coefficients of each factor exactly under the assumption of Subset Strong Convexity and Subset Strong Smoothness properties. Specifically, we prove that our algorithm ensures a rigorous error bound of the regression coefficient compared to ground truth. 4) **Conducting extensive experiments for performance evaluations.** The proposed method was evaluated on both synthetic data and real-world datasets with various corruption settings. The experimental results show that our proposed RMFP algorithm runs efficiently and can consistently outperform the other existing methods along multiple metrics.

The rest of this paper is organized as follows. Section 7.2 reviews background and related work, and Section 7.3 introduces the problem setup. The proposed RMFP algorithm is presented in Section 7.4. Section 7.5 presents the proof of convergence rate and recovery guarantee. The experiments on both synthetic and real-world datasets are presented in Section 7.6, and the paper concludes with a summary of the research in Section 7.7.

7.2 Related Work

The work related to this paper falls into three categories and is summarized below.

Personality prediction in social media: Many approaches [84] [54] [119] that combine personality and social media have been proposed. Some research focuses on the association relation between personality and behavior rather than quantitative analysis of personality. For instance, Orr et al. [84] discovered the relation between shyness and the number of "friends", while Correa et al. [20] found extraversion and openness have a positive relation

to the user experience in social media. A large body of work explores linguistic feature selection in personality prediction. The studies focus on a broad set of demographic and psychological features in user postings such as social status [54] [77], occupation [89], mental illness [38] [90], political orientation [85], and gender [12], as well as other online behavior features such as Facebook likes [52] and profile pictures [16] [60]. However, none of these works considers the noise or corruption in the social media dataset. Recently, Zhang et al. [119] proposed the only work that considers the robustness of the personality prediction model. However, the model treats the corruption of different factors independently and does not consider the correlation between the corruption of multiple factors.

Robust regression model: A large body of literature on the robust regression problem has been built up over the last few decades. Most of them lack the theoretical guarantee of regression coefficients recovery [100] [41] [48]. To pursue the exact recovery results for the robust regression problem, some work focused on L_1 penalty-based convex formulations [109] [82]. However, these methods imposed severe restrictions on the data distribution such as row-sampling from an incoherent orthogonal matrix [82]. Several studies require the corruption ratio parameter, which is difficult to determine manually. For instance, She and Owen [99] 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. [19] require the upper bound of outliers number, which is also difficult to estimate. Recently, Bhatia et al. [5] proposed a hard-thresholding algorithm. Although the method gives a strong guarantee of coefficient recovery under a mild assumption on input data, their results are highly dependent on the corruption ratio parameter inputed by users. Zhang et al. [119] consider the robust multivariate regression problem in personality prediction with gross corruption; however, their method also requires regularization parameters to control the ratio and sparsity of corruption. None of these approaches focuses specifically on the robust multi-factor regression problem with a correlated corruption ratio under a strong recovery guarantee.

Multi-task learning: Multiple related tasks are simultaneously learned in multi-task learning (MTL) to prove the generalization performance [126]. Many multi-task approaches [125] [127] [31] have been proposed in recent years to model the relationship between tasks. The relatedness of tasks can be characterized by making multiple tasks to share a common feature space, such as a common subspace [125] [31], a common set of features [1], or using a structured model [51]. For instance, Evgeniou et al. [31] proposed a regularized multi-task learning method that make the models of all the tasks as close as each other. Multi-task learning methods have been utilized in many applications, including biomedical informatics and image processing. To the best of our knowledge, our work is the first one that applies multi-task learning in the domain of robust personality prediction.

7.3 Problem Setting

In this section, the problem addressed by this research is formulated.

Denote $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$ as a collection of social media features for N users, where each column $\boldsymbol{x}_i \in \mathbb{R}^{P\times 1}$ represents the feature set for the i^{th} user, and $\boldsymbol{y} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(M)}\}$ represents the personality labels for M factors, where $\boldsymbol{y}^{(m)} \in \mathbb{R}^{N\times 1}$ is the personality label for all the N users in the m^{th} factor. We assume the response vector $\boldsymbol{y}^{(m)}$ of the m^{th} factor is generated using the model:

$$\boldsymbol{y}^{(m)} = \boldsymbol{X}^T\boldsymbol{\beta}_*^{(m)} + \boldsymbol{u}^{(m)} + \boldsymbol{\varepsilon}^{(m)} \tag{7.1}$$

where $\boldsymbol{\beta}_*^{(m)}$ represents the ground truth coefficients of the regression model and $\boldsymbol{u}^{(m)} \in \mathbb{R}^{N\times 1}$ is the unbounded corruption vector introduced adversarily or unintentionally. When the i^{th} sample is uncorrupted, we have $\boldsymbol{u}_i^{(m)} = 0$; otherwise, $\boldsymbol{u}_i^{(m)}$ represents the corresponding corruption value. Denoting S_* as the set of uncorrupted samples in m^{th} factor, then we have $S_* = \overline{supp(\boldsymbol{u}^{(m)})}$, where $supp(\cdot)$ is the subset whose elements are not zero. $\boldsymbol{\varepsilon}^{(m)}$ is denoted as the additive dense noise for the m^{th} factor, where $\varepsilon_i^{(m)} \sim \mathcal{N}(0, \sigma^2)$. The notations used in this paper are summarized in Table 7.2.

Before formally stating the problem, we first introduce two definitions related to corruption in personality assessment.

Definition 1. ***Corruption in personality assessment.*** *Corruption or faking in personality assessment represents the response distortion aimed at providing a self-description that helps to achieve personal goals (e.g., social desirability [83], careless response [74]).*

Existing psychological research [132] [131] shows that corruption indeed increases correlations between the Big Five factors in personality. The empirical evidence of these works concludes that faking on personality assessment can be considered a source of systematic variance that is added to (or even replaces) the true score variance. For example, a faked score on one particular test of *extraversion* has higher correlations with other factors of personality such as *conscientiousness* or *neuroticism* than scores on other *extraversion* tests. Based on this observation, we formally define the correlated corruption property as follows.

Definition 2. ***Correlated Corruption Property.*** *Denoting $\boldsymbol{u}^{(m)}$ as the corruption vector of the m^{th} factor, the M factors satisfy the correlated corruption property if the following holds:*

$$supp(\boldsymbol{u}^{(i)}) = supp(\boldsymbol{u}^{(j)}) \quad \forall i, j \in \{1...M\} \tag{7.2}$$

where $supp(\boldsymbol{u}^{(i)})$ denotes the set of corrupted points in the i^{th} factor.

The correlated corruption property assumes all the personality factors have correlated data corruption. Specifically, the corruption vector $\boldsymbol{u}^{(i)}$ for the i^{th} factor has the same non-zero

Table 7.2: Math Notations

Notations	Explanations
$X \in \mathbb{R}^{F\times N}$	collection of social media feature set
$\boldsymbol{y}^{(m)} \in \mathbb{R}^{N\times 1}$	response vector of the m^{th} factor
$\boldsymbol{\beta}^{(m)} \in \mathbb{R}^{F\times 1}$	estimated regression coefficient of the m^{th} factor
$\boldsymbol{u}^{(m)} \in \mathbb{R}^{N\times 1}$	corruption vector of the m^{th} factor
$\boldsymbol{r}^{(m)} \in \mathbb{R}^{N\times 1}$	residual vector of the m^{th} factor
$S \subseteq [n]$	estimated uncorrupted set
$S_* \subseteq [n]$	ground truth uncorrupted set, where $S_* = \overline{supp(\boldsymbol{u})}$

elements as any other factor. The property utilizes the observation of correlated corruption in psychological research [131] and establishes the relation between multiple factors in Big-Five personality domain.

The goal of our study is to learn a new multi-factor robust regression problem with correlated corruption, which jointly recovers the multi-factor regression coefficients $\hat{\boldsymbol{\beta}} = \{\hat{\boldsymbol{\beta}}^{(1)}...\hat{\boldsymbol{\beta}}^{(M)}\}$ and determines the correlated uncorrupted set $\hat{S}$ simultaneously. The problem is formally defined as follows:

$$\begin{aligned} \hat{\boldsymbol{\beta}}, \hat{S} &= \arg\min_{\boldsymbol{\beta}, S} \sum_{m=1}^{M} \|\boldsymbol{y}_S^{(m)} - X_S^T \boldsymbol{\beta}^{(m)}\|_2^2 \\ s.t. \quad S &\in \left\{\Gamma\big(Z\big) \;\middle|\; \forall m \in \{1, \dots, M\} : |S^{(m)}| \geq h(\boldsymbol{r}^{(m)})\right\} \end{aligned} \tag{7.3}$$

Denoting $S^{(m)}$ as the estimated uncorrupted set for the m^{th} factor and $Z = \{S^{(1)}, \dots, S^{(M)}\}$ as the collection of uncorrupted set for all the factors, the function $\Gamma(\cdot)$ consolidates the estimation of all the factors into one correlated uncorrupted set $S \subset [n]$. $\boldsymbol{y}_S$ 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 $\boldsymbol{y}_S^{(m)} \in \mathbb{R}^{|S|\times 1}$ and $X_S \in \mathbb{R}^{P\times|S|}$. The size of $S^{(m)}$ is lower-bounded by a heuristic function $h(\cdot)$ according to the m^{th} factor's residual vector $\boldsymbol{r}^{(m)} = \boldsymbol{y}^{(m)} - X^T\boldsymbol{\beta}^{(m)}$. Also, we use $\boldsymbol{r}_S^{(m)}$ to represent the $|S|$-dimensional residual vector containing the components in S for the m^{th} factor. The detail of heuristic function $h(\cdot)$ and consolidation function $\Gamma(\cdot)$ will be explained in Section 7.4.1 and 7.4.2.

7.4 RMFP Model

In this section, we propose a new model, Robust Multi-Factor Personality Prediction (RMFP), which is based on robust multi-factor regression. In Section 7.4.1, the corruption estimation for single factor is characterized mathematically by a heuristic hard thresholding method,

and then these estimations are consolidated for multiple factors in Section 7.4.2. In Section 7.4.3, a novel distributed robust regression algorithm is proposed that ensures a strong guarantee of coefficient recovery.

7.4.1 Single-Factor Corruption Estimation

To estimate the data corruption for single factor, we propose a new method that heuristically estimates the size of uncorrupted set based on the factor's residual vector. Specifically, the size of uncorrupted set $S^{(m)}$ for m^{th} factor in Equation (7.3) is lower-bounded by heuristic function $h(\cdot)$, which is defined as follows.

$$h(\boldsymbol{r}^{(m)}) := \underset{\tau \in \mathbb{Z}^+, \tau \leq n}{\arg\max}\, \tau \quad s.t. \;\; r^{(m)}_{\delta(\tau)} \leq \frac{2\tau r^{(m)}_{\delta(\tau_o)}}{\tau_o} \tag{7.4}$$

The variable τ_o in the constraint is defined as follows:

$$\tau_o = \underset{1 \leq \tau \leq n}{\arg\min} \left| \left(r^{(m)}_{\delta(\tau)}\right)^2 - \frac{\|\boldsymbol{r}^{(m)}_{S_{\tau'}}\|_2^2}{\tau'} \right| \tag{7.5}$$

where $\tau' = \tau - \lceil n/2 \rceil$ and $S_{\tau'}$ is the position set containing the smallest τ' elements in residual $\boldsymbol{r}^{(m)}$, and $r^{(m)}_{\delta(k)}$ represents the k^{th} elements of $\boldsymbol{r}^{(m)}$ in ascending order of magnitude.

Basically, the design follows a natural intuition that data points with unbounded corruption always have a higher residual $r_i^{(m)} = y_i^{(m)} - X_i\boldsymbol{\beta}^{(m)}$ in magnitude compared to uncorrupted data. Moreover, the constraint in Equation (7.4) ensures the residual of the largest element τ in our estimation cannot be too much larger than the residual of a smaller element τ_o. This is because if the element τ_o is too small, some uncorrupted elements will be excluded from our estimation; otherwise, if it is too large, some corrupted elements will be included. The formal definition of τ_o is shown in Equation (7.5), in which τ_o is defined as a value whose squared residual is closest to $\|\boldsymbol{r}^{(m)}_{S_{\tau'}}\|_2^2/\tau'$, where τ' is less than the ground truth threshold τ_* as the corruption ratio is typically assumed not larger than half [19] [5]. This design ensures that $|S_*^{(m)} \cap S_t^{(m)}| \geq \tau - n/2$, which means at least $\tau - n/2$ elements are correctly estimated in $S_t^{(m)}$. The property will be used in the proof in Lemma 2 in Section 7.5. 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 (7.4) chooses the maximum uncorrupted set size.

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

Definition 3. ***Heuristic Hard Thresholding***. *Denoting $\delta_{\boldsymbol{r}}^{-1}(i)$ as the position of the i^{th} element in residual vector $\boldsymbol{r}$'s ascending order of magnitude, the heuristic hard thresholding of $\boldsymbol{r}$ is defined as*

$$\mathcal{H}_\tau(\boldsymbol{r}) = \{i \in [n] : \delta_{\boldsymbol{r}}^{-1}(i) \leq h(\boldsymbol{r})\} \tag{7.6}$$

ALGORITHM 7: RMFP ALGORITHM

Input: Multi-factor training dataset X, $\boldsymbol{y}$, tolerance ϵ
Output: solution $\hat{\boldsymbol{\beta}}$, $\hat{S}$
$S_0 \leftarrow [n]$, $\Psi \leftarrow [M]$, $t \leftarrow 0$
repeat
  for $m \in \Psi$ **do**
    $\boldsymbol{\beta}_{t+1}^{(m)} \leftarrow (X_{St} X_{St}^T)^{-1} X_{St} \boldsymbol{y}_{St}^{(m)}$
    $\boldsymbol{r}_{t+1}^{(m)} \leftarrow |\boldsymbol{y}^{(m)} - X^T \boldsymbol{\beta}^{(m)}|$
    $S_{t+1}^{(m)} \leftarrow \mathcal{H}_\tau(\boldsymbol{r}_{t+1}^{(m)})$ *// **Heuristic hard threshold on residual***
    if $\|\boldsymbol{r}_{t+1}^{(m)} - \boldsymbol{r}_t^{(m)}\|_2 / n < \epsilon$ **then**
      $\Psi \leftarrow \Psi \setminus \{m\}$ *// **Remove factor from active set***
    end
  end
  $S_{t+1} = \Gamma(S_{t+1}^{(1)}, ..., S_{t+1}^{(M)})$ *// **Corruption Consolidation: GC or MV***
  $t \leftarrow t + 1$
until $\Psi = \emptyset$
return $\boldsymbol{\beta}_t^{(1)} ... \boldsymbol{\beta}_t^{(M)}, S_t$

The optimization of $S^{(m)}$ is formulated as solving Equation (7.6), where the set returned by $\mathcal{H}_\tau(\boldsymbol{r}^{(m)})$ will be used as the estimated uncorrupted set of the m^{th} factor.

7.4.2 Corruption Consolidation

To consolidate the uncorrupted sets of all the factors, we propose two strategies for the consolidation function $\Gamma(\cdot)$ in Equation (7.3): *global consensus* (GC) and *majority voting* (MV). The definition of global consensus strategy is presented as follows.

Definition 4. ***Global Consensus.*** *Denoting $S^{(m)}$ as the uncorrupted set of the m^{th} factor, the global consensus uncorrupted set S is defined as $S = \bigcap_{m=1}^{M} S^{(m)}$.*

Global consensus strategy selects the estimated uncorrupted set as the intersection set from all the factors. The strategy strictly follows the *correlated corruption property* (Definition 2) and only keeps the uncorrupted elements contained in estimators of all the factors. It increases the precision of the estimated uncorrupted set, but also excludes some correct samples from the estimation. As dense noise will increase the difference between the estimations of multiple factors, the strategy is more suitable to cases in which the amount of dense noise is small or there is no dense noise.

To avoid the problem of global consensus strategy, we propose the second strategy, majority voting, as follows.

Definition 5. ***Majority Voting.*** *Denoting $S^{(m)}$ as the uncorrupted set of the m^{th} factor and M as the total number of factors, the uncorrupted set S of majority voting satisfies:*

$$\sum_{m=1}^{M} \mathbb{1}\Big(S_i \in S^{(m)}\Big) \geq \left\lceil \frac{M}{2} \right\rceil \qquad \forall i \leq |S| \tag{7.7}$$

Different from global consensus, which requires the consolidated elements to be contained in estimators of all factors, the majority voting strategy requires them to be contained only in the majority of factors. It can handle the case of an uncorrupted element contained in estimators of all the factors except one factor that has a large amount of dense noise in the element. Therefore, the majority voting strategy is more suitable to cases with dense noise. However, with little dense noise, it will introduce some false positive elements into the estimation.

7.4.3 Algorithm of RMFP

In order to efficiently solve the problem in Equation (7.3), we propose a novel distributed robust regression algorithm, RMFP, in Algorithm 7.

In the algorithm, the active set Ψ is initialized as all the factors from 1 to M, and the initial uncorrupted set S_0 is set as the entire data samples in Line 1. Then the algorithm follows an intuitive strategy of updating $\boldsymbol{\beta}^{(m)}$ for each factor to provide a better fit for the current estimated set S in line 4, and updating the residual vector for each factor in Line 5. It then estimates the uncorrupted set $S^{(m)}$ of each factor via heuristic hard thresholding in Line 6 based on the residual vector $\boldsymbol{r}^{(m)} = \boldsymbol{y}^{(m)} - X^T\boldsymbol{\beta}^{(m)}$ in the current iteration. In Lines 7 and 8, the factor whose residual difference compared to the previous iteration is smaller than a predefined threshold will be removed from the active set Ψ. After the uncorrupted set of all the factors is computed, the estimation of all the factors will be consolidated in Line 9 with either the global consensus or majority voting strategy. The algorithm continues until the active set Ψ is empty. To improve the efficiency of the algorithm, the for loop from Lines 3 to 8 can be run in parallel for each factor.

Figure 7.1 shows the residual of the uncorrupted set of one factor in the 1^{st} and 7^{th} iterations for both global consensus and majority voting strategies, respectively. It intuitively explains the convergence progress of our algorithm: The optimization steps of $\boldsymbol{\beta}^{(m)}$ based on the consolidated uncorrupted set S_t make the overall $\boldsymbol{r}^{(m)}_{S_t}$ smaller than its previous iteration; then these items in S_t have a much higher possibility to be kept in S_{t+1} than items in the set $[n] \setminus S_t$. This progress continues until the consolidated uncorrupted set is fixed. Figures 7.1(b) and 7.1(d) show that the correlated uncorrupted set by global consensus strategy has a smaller size than majority voting strategy.

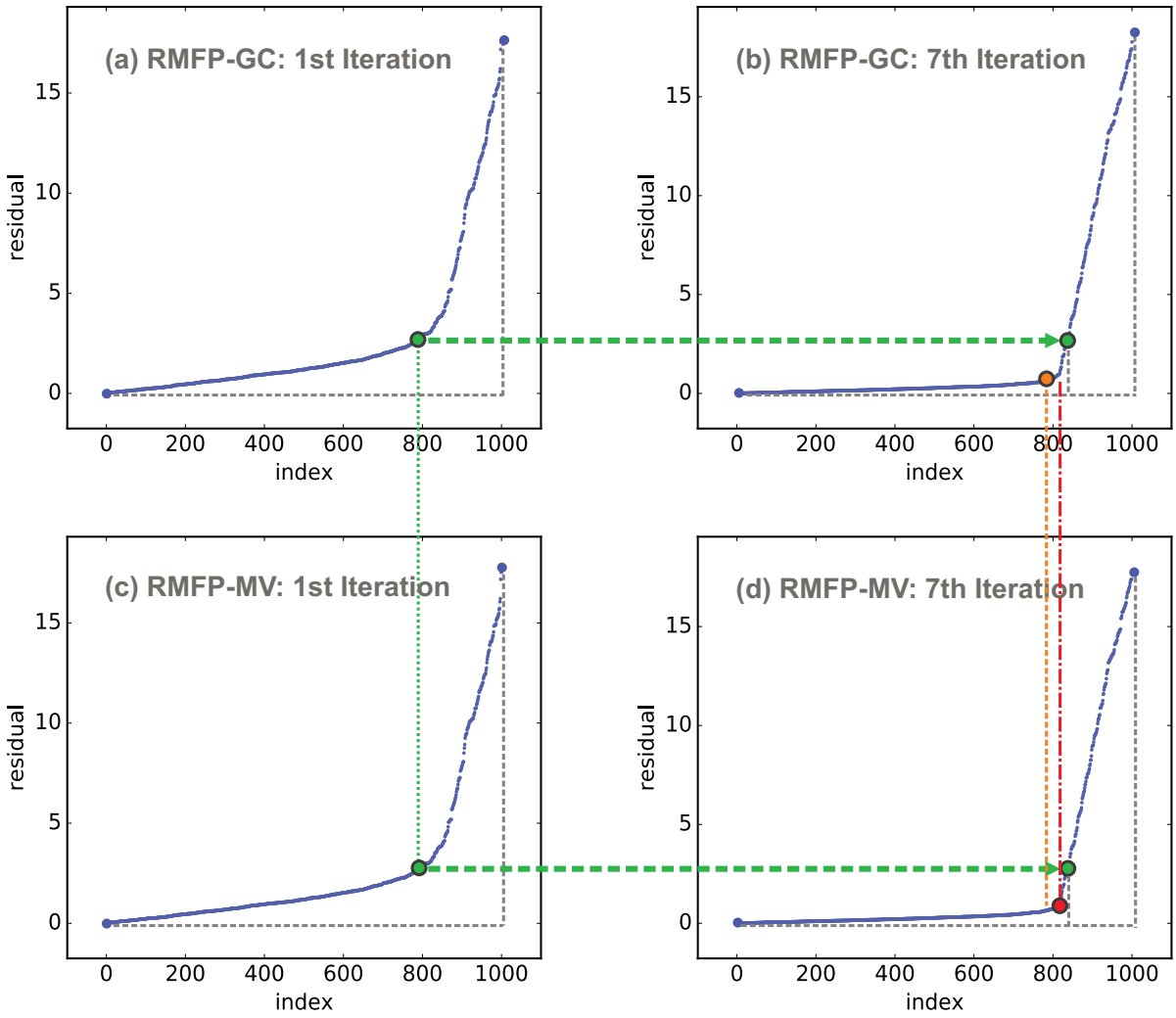


Figure 7.1: Residual $\boldsymbol{r}$ of one factor in ascending order for the 1^{st} (left) and 7^{th} (right) iterations in global consensus and majority voting strategies.

7.5 Recovery Analysis

In this section, the convergence analyses for the case with a global consensus strategy will be presented.

Lemma 2. *Let τ^t be the estimated uncorrupted threshold at the t-th iteration. If $\tau_* = \gamma n$, then each factor's residual satisfies $\|\boldsymbol{r}^t_{S_t}\|_2^2 \leq \left[1 + \frac{128(1-\gamma)}{2\gamma - 1}\right] \|\boldsymbol{r}^t_{S_*}\|_2^2$.*

The proof of Lemma 2 can be found in supplementary document [1]. Lemma 2 gives an upper bound of the residual value of the estimated uncorrupted set compared to the ground truth uncorrupted set. When γ is very close to 1, $\|\boldsymbol{r}_{S_t}\|_2^2$ reaches its upper bound $\|\boldsymbol{r}_{S_*}\|_2^2$. To prove the theoretical recovery of regression coefficients, we require that the least squares function satisfies the *Subset Strong Convexity (SSC)* and *Subset Strong Smoothness (SSS)*, which are defined as follows:

Definition 6. ***SSC and SSS Properties***. *The least squares function $f(\boldsymbol{\beta}) = \|\boldsymbol{y}_S - X_S^T\boldsymbol{\beta}\|_2^2$ satisfies the $2\zeta_\gamma$-Subset Strong Convexity property and the $2\kappa_\gamma$-Subset Strong Smoothness property if the following holds:*

$$\zeta_\gamma I \preceq \frac{1}{2}\nabla^2 f_S(\boldsymbol{\beta}) \preceq \kappa_\gamma I \quad \text{for } \forall S \in S_\gamma \tag{7.8}$$

Note that Equation (7.8) is equivalent to:

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

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

Theorem 2. *Let $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^p$ be the given data matrix and the corrupted output of each factor $\boldsymbol{y}^{(m)} = X^T\boldsymbol{\beta}_*^{(m)} + \boldsymbol{u}^{(m)} + \boldsymbol{\varepsilon}^{(m)}$ with $\|\boldsymbol{u}^{(m)}\|_0 = \gamma n$. Let Σ_0 be an invertible matrix such that $\tilde{X} = \Sigma_0^{-1/2}X$, $f(\boldsymbol{\beta}^{(m)}) = \|\boldsymbol{y}_S^{(m)} - \tilde{X}_S\boldsymbol{\beta}^{(m)}\|_2^2$ 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 $t = \mathcal{O}\left(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}\|_2}{\sqrt{n}\epsilon}\right)$ iterations, Algorithm 7 yields an ϵ-accurate solution $\boldsymbol{\beta}_t^{(m)}$ with $\|\boldsymbol{\beta}_*^{(m)} - \boldsymbol{\beta}_t^{(m)}\|_2 \leq \epsilon + \frac{C\|\boldsymbol{\varepsilon}\|_2}{\sqrt{n}}$ for some $C > 0$.*

Proof. To simplify the notation, we will omit all the superscripts (m) that denotes the m^{th} factor in the explanation below. We use S_t to represent the uncorrupted set after consolidation in the t^{th} iteration. We observe that the optimality of the regression coefficient $\boldsymbol{\beta}^{(m)}$ on the estimated corrupted set S_t ensures:

[1] `https://goo.gl/WhZJ6T`

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

Using the triangle inequality of L_2 norm, we have

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

Inequality *(a)* follows $|S_t| \leq \alpha n$, where $\alpha = \max_t\{\frac{\tau_t}{n}\}$. Inequality *(b)* follows the triangle inequality $\|\boldsymbol{\varepsilon}_{S_t} + \boldsymbol{u}_{S_t}\|_2 \leq \|\boldsymbol{\varepsilon}_{S_t}\|_2 + \|\boldsymbol{u}_{S_t}\|_2$. According to the hard thresholding step in Definition 3, we have:

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

Inequality *(c)* follows the Lemma 2, 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

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

Let $\varphi_{\alpha,\gamma} = \sqrt{\kappa_\alpha} + (\lambda + \sqrt{\lambda^2-1})\sqrt{\kappa_{1-\gamma}}$, inequality *(e)* follows $|\mathrm{FP}_{t+1}| \leq \alpha n$, $|\mathrm{TP}_{t+1}| \leq (1-\gamma)n$, $|\mathrm{FN}_{t+1}| \leq (1-\gamma)n$ and $\|\boldsymbol{\varepsilon}_{S_{t+1}}\|_2 \leq \|\boldsymbol{\varepsilon}\|_2$. Inequality *(f)* follows 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

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

For a large enough n, we have $\sqrt{\zeta_\alpha} \geq \mathcal{O}(\sqrt{n})$. Let $C = \frac{2(3+2\lambda-\eta)}{1-\eta}$, then after $t = \mathcal{O}\left(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}\|_2}{\sqrt{n}\epsilon}\right)$ iterations, Algorithm 7 yields an ϵ-accurate solution $\boldsymbol{\beta}^t$ with $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_t\|_2 \leq \epsilon + \frac{C\|\boldsymbol{\varepsilon}\|_2}{\sqrt{n}}$. □

7.6 Experiment

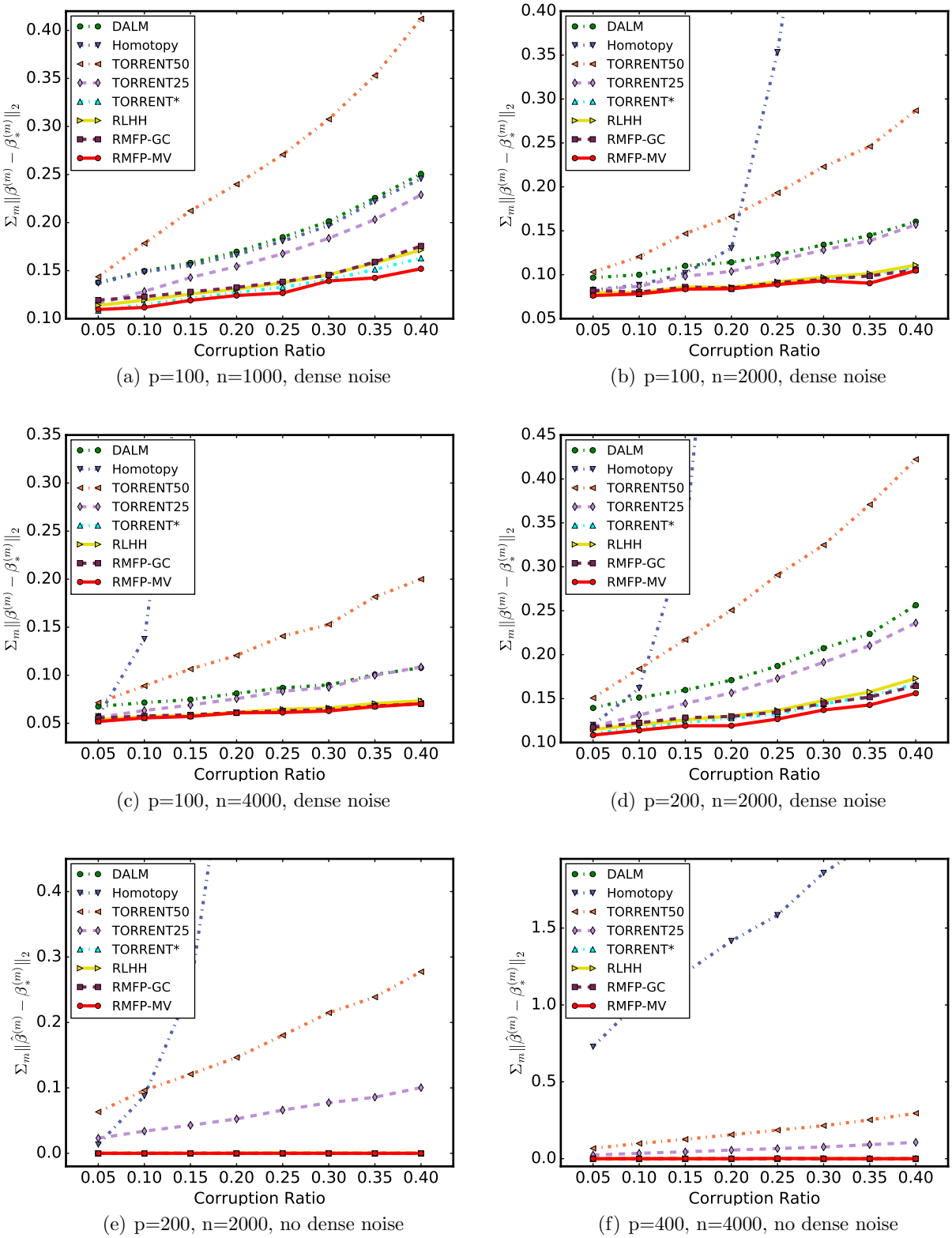
In this section, the proposed RMFP model is evaluated on both synthetic and real-world datasets. After the experiment setup has been introduced in Section 7.6.1, the effectiveness of the methods is evaluated against several existing methods on both the synthetic and real-world datasets, along with an analysis of efficiency for all the comparison methods, in Section 7.6.2. 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 experiment can be downloaded here.[2]

7.6.1 Experiment Setup

Datasets and Labels

Our dataset is composed of synthetic and real-world data. The simulation samples were randomly generated according to the model in Equation (7.1) for each factor, sampling the regression coefficients $\boldsymbol{\beta}_*^{(m)} \in \mathbb{R}^p$ as a random unit norm vector. The covariance data X was independently drawn and identically distributed from $\boldsymbol{x}_i \sim \mathcal{N}(0, I_p)$, and the uncorrupted response variables were generated as $\boldsymbol{y}_*^{(m)} = \boldsymbol{X}^T\boldsymbol{\beta}_*^{(m)}$. The set of uncorrupted points S_* was selected as a uniformly random (n-τ_*)-sized subset of $[n]$, where τ_* is the size of the uncorrupted set. The corrupted response vector for each factor was generated as

[2] `https://goo.gl/JEoo5j`



(a) p=100, n=1000, dense noise

(b) p=100, n=2000, dense noise

(c) p=100, n=4000, dense noise

(d) p=200, n=2000, dense noise

(e) p=200, n=2000, no dense noise

(f) p=400, n=4000, no dense noise

Figure 7.2: Performance on regression coefficients recovery for different corruption ratios.

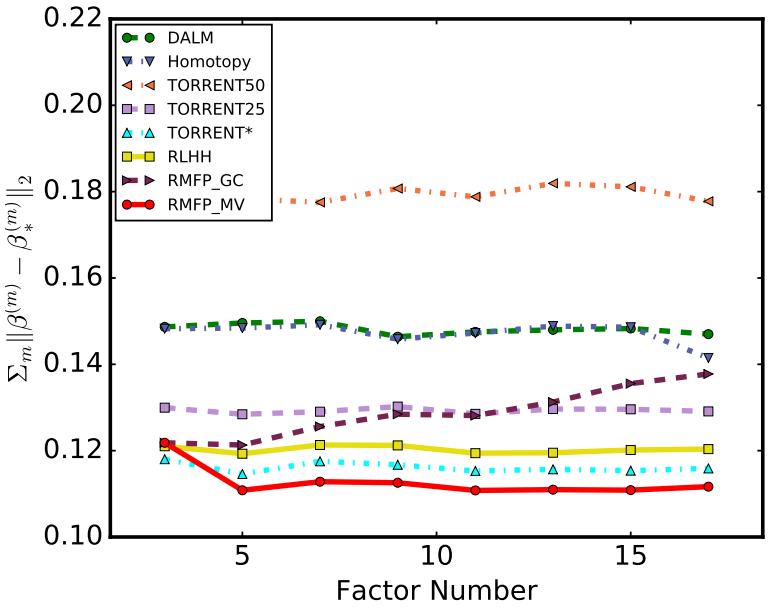


Figure 7.3: Regression coefficients recovery based on factor number with p=100, n=1000, and dense noise.

$\boldsymbol{y}^{(m)} = \boldsymbol{y}_*^{(m)} + \boldsymbol{u}^{(m)} + \boldsymbol{\varepsilon}^{(m)}$, where the corruption vector $\boldsymbol{u}^{(m)}$ was sampled from the uniform distribution $[-5\|\boldsymbol{y}_*^{(m)}\|_\infty, 5\|\boldsymbol{y}_*^{(m)}\|_\infty]$ and the additive dense noise was $\varepsilon_i^{(m)} \sim \mathcal{N}(0, \sigma^2)$. According to our correlated corruption assumption, the corruption vector for each factor used the unified corruption set S_*, where we have $S_* = \overline{supp(\boldsymbol{u}^{(m)})}$. Note that although each factor shares the same corruption set, the volume of corruption is factor dependent.

The real-world dataset we use is published in [61], which is built from the Sina microblog (the Chinese counterpart of Twitter). In that dataset, 1,721 volunteers, who have enough Sina microblog data, are recruited to participate in the data collection process. All of them are required to finish the Big Five questionnaire [14] online as their personality labels, and their microblogs and public personal information such as age, gender, and personal description are authorized to be obtained. We retrieve the Linguistic Inquiry and Word Count (LIWC) features from the user microblogs, which contain 63 features in total. We use the first 1,000 samples as training data and the remaining samples as testing data. Also, invalid and noisy data, such as too short answering time, is kept to evaluate our robust model.

Evaluation Metrics

For the synthetic data, we measured the performance of the regression coefficients recovery using the averaged L_2 error:

$$e = \frac{1}{M}\sum_{m=1}^{M}\|\hat{\boldsymbol{\beta}}^{(m)} - \boldsymbol{\beta}_*^{(m)}\|_2$$

where $\hat{\boldsymbol{\beta}}^{(m)}$ represents the recovered coefficients for each method and $\boldsymbol{\beta}_*^{(m)}$ is the true regression coefficients. To validate the performance for corrupted set discovery, precision, recall, and F1-score are measured by comparing the discovered corrupted sets with the actual ones. 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 Pearson correlation coefficient (PCC) to evaluate the linear correlation between the predicted personality score $\hat{\boldsymbol{y}}^{(m)}$ and labeled personality $\boldsymbol{y}_*^{(m)}$ as follows:

$$\rho_{X,Y} = \frac{\boldsymbol{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}$$

where μ_X is the mean of X and σ_X is the standard deviation of X. The Pearson correlation coefficient has a value between -1 and +1, where +1 is total positive linear correlation, 0 represents no linear correlation, and -1 stands for total negative linear correlation.

Comparison Methods

The following methods are included in the performance comparison presented here: *Ordinary least squares* (*OLS*). The *OLS* method ignores the corruption of data and trains the model based on the whole dataset. We also compared our method to the regularized L_1 algorithm for robust regression [109] [82]. For extensive L_1 minimization solvers, [113] showed that the *Homotopy* and *DALM* solvers outperform other proposed methods both in terms of recovery properties and running time. Both of the L_1 solver methods are parameter free. A hard thresholding method, *TORRENT (abbreviated "Torr")*, developed for robust regression [5] 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 three versions with different parameter settings: *TORR**, *TORR25*, and *TORR50*. *TORR** uses the true corruption ratio as its parameter, and the others apply parameters that are uniformly distributed across the range of $\pm 25\%$, and $\pm 50\%$ off the true value, respectively. Another recently proposed heuristic hard thresholding method, *RLHH* [122], is also compared in our experiment. The method is a parameter-free approach, as it estimates the corruption set by a heuristic hard thresholding method . As all these methods are not designed for the multi-factor robust regression problem with correlated corruption, we run them individually for each factor under

Table 7.3: Pearson Correlation of Personality Prediction

	Agr.	*Con.*	*Ext.*	*Ope.*	*Neu.*	**Avg**
OLS	0.2461/35.46%	0.2437/33.07%	**0.1921/24.95%**	0.2733/37.03%	0.0910/11.88%	0.2092/28.48%
TORR	0.2075/29.90%	0.2157/29.27%	0.1766/22.94%	0.2405/32.59%	0.0971/12.68%	0.1875/25.47%
RLHH	0.2111/30.42%	0.2332/31.64%	0.1867/24.25%	0.2739/37.11%	0.1064/13.89%	0.2023/27.46%
RMFP-GC	0.2146/30.92%	0.2296/31.15%	0.1887/24.51%	0.2552/34.58%	**0.1098/14.33%**	0.1996/27.10%
RMFP-MV	**0.2472/35.62%**	**0.2442/33.13%**	0.1919/24.92%	**0.2743/37.17%**	0.0967/12.62%	**0.2109/28.69%**

the correlation corruption assumption. For our proposed methods, we use *RMFP-GC* and *RMFP-MV* to represent the *RMFP* algorithm with a global consensus or majority voting strategy, respectively. All the results will be averaged over 10 runs.

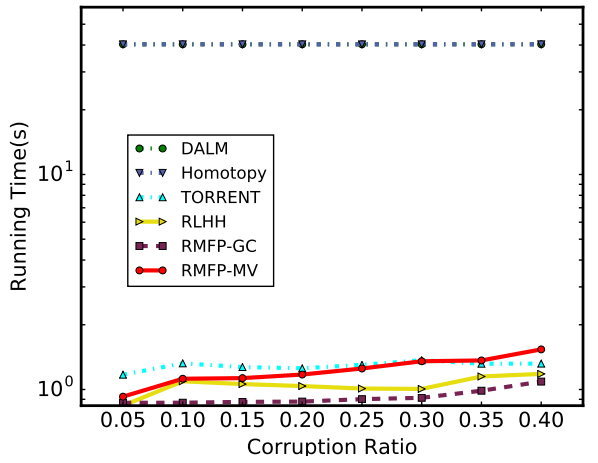
7.6.2 Performance

This section presents the recovery performance of the regression coefficients and the uncorrupted sets.

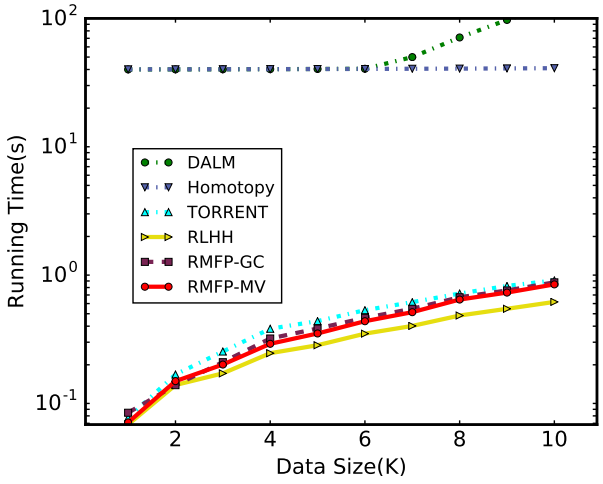
Recovery of regression coefficients

We selected seven competing methods with which to evaluate the average recovery performance of all the factors: *OLS*, *DALM*, *Homotopy*, *TORR**, *TORR25*, *TORR50*, and *RLHH*. As the recovery error for the *OLS* method is almost 10 times larger than those of the other methods, its result is not shown in Figure 7.2 in order to present the other results properly. Figures 7.2(a), 7.2(b), and 7.2(c) show the recovery performance for different data sizes when the feature number is fixed. Looking at the results, we can conclude: 1) The *RMFP-MV* method outperforms all the competing methods, including *TORR**, whose corruption ratio parameter uses the ground truth value. Also, the *RMFP-GC* method has a very competitive result compared to *TORR** and *RLHH*. 2) The results of the *TORR* 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, both *RMFP-GC* and *RMFP-MV* 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 7.2(d) shows its similar performance when the feature number increases. Figures 7.2(e) and 7.2(f) show that both the *RMFP-GC* and *RMFP-MV* performs equally as well as *TORR** and *RLHH* without dense noise, with both achieving an exact recovery of regression coefficients $\boldsymbol{\beta}$.

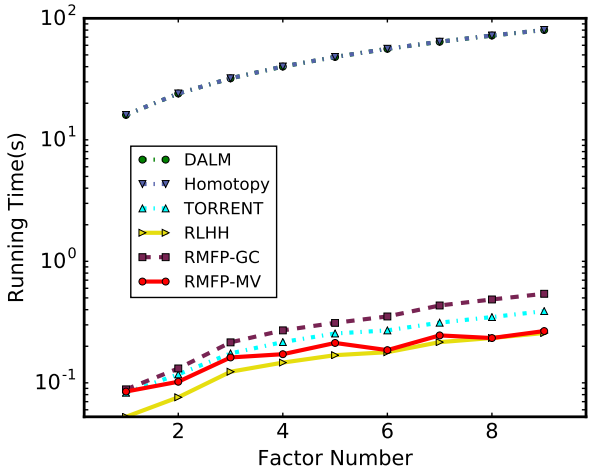
Figure 7.3 shows the result of regression coefficient recovery based on different factor numbers, from which we conclude: 1) The *RMFP-MV* algorithm outperforms the competing methods in all the settings of factors except when the factor is three. It is because the



(a) p=400, n=4000, no dense noise



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



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

Figure 7.4: Running time for different corruption ratios and data sizes

majority number of three factors is so small that it is easier to include the corrupted samples and exclude uncorrupted ones than a larger factor number. 2) The recovery error of the *RMFP-GC* algorithm increases when the number of factors rises. It is because the correlation corruption property used in *RMFP-GC* restricts the uncorrupted samples are accepted by all the factors, which leads to more uncorrupted samples being excluded from the estimation. 3) Similar to the result in Figure 7.2, hard thresholding based algorithms generally outperform L_1-Solver methods in different settings of factor numbers. However, the *TORR50* algorithm performs worse than other methods because a badly estimated corruption ratio is used as its parameter. Due to the space limitation, the performance of uncorrupted set recovery can be found in supplementary document [3].

Result of Personality Prediction

To evaluate the performance of personality prediction, we compared our proposed methods of *RMFP-GC* and *RMFP-MV* to competing methods, including *OLS*, *RLHH*, and *TORR* algorithm. For the setting of the *TORR* algorithm, a fixed 10% corruption ratio is used as its parameter, while the majority number in the *RMFP-MV* algorithm is set as 4 out of 5. The *Pearson correlation coefficient* is used as a metric to evaluate the correlation between the estimated personality scores and labeled scores. The result is shown in Table 7.3, where the columns represent the scores of the five personality factors: *agreeableness*(*Agr.*), *conscientiousness*(*Con.*), *extraversion*(*Ext.*), *openness*(*Ope.*), *neuroticism*(*Neu.*), and their average scores. As one person can get varied personality scores in different personality questionnaires, we use the correlation score from a test-retest reliability experiment [36] as the optimal value in this domain. The value is estimated by correlating scores obtained in the first test with scores from a second test approximately six week later. The optimal correlation value for each factor in the Chinese language [15] is listed as follows: *Agr.* (0.694), *Con.* (0.737), *Ext.* (0.770), *Ope.* (0.738) and *Neu.* (0.766). In Table 7.3, we show the percentage of the evaluated method compared to the optimal value in the second column of each factor.

From the result in Table 7.3, we conclude: 1) *RMFP-MV* outperforms all the competing methods in an average of the Pearson correlation. However, the result of *RMFP-GC* is 5.6% worse than *RMFP-MV* since it uses a more strict assumption of correlated corruption. 2) The *RLHH* method, which considers the robustness for each factor independently, only competes with the *OLS* method in two factors: *Ope.* and *Neu.* The facts show that the correlated corruption property applied in *RMFP-MV* can improve overall performance for multiple factors. 3) The *TORR* algorithm using a fixed corruption ratio performs worse than other methods because some corrupted samples can be included if the estimated corruption is less than the actual corruption. 4) Compared to other factors, the performance of *Neu.* is 55.8% worse on average for all the methods, which means the *Neu.* factor cannot be properly handled by evaluated methods in the collected data. Also, we found that average correlation

[3] `https://goo.gl/X7FmRQ`

can be improved by 12.56% if the *Neu.* factor is removed.

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, and factor numbers. In general, as Figure 7.4 shows, we conclude: 1) The *RMFP* method has a very competitive performance even though it performs the additional consolidation step in either global consensus or majority voting in each optimization iteration. The efficiency difference compared to *RLHH* and *TORR* is trivial, which indicates that the consolidation step in *RMFP* always performs efficiently in different data settings. 2) The running time for *RMFP* methods increases slowly as either the data size or factor number increases, just as in the *TORR* and *RLHH* methods. When the corruption ratio increases, the running time of *RMFP* increases within 10%, which means that data corruption has little impact on the efficiency of our method. 3) The hard thresholding based methods significantly outperform the L_1-Solver based methods.

7.7 Conclusion

In this paper, a novel robust model is proposed to handle multi-factor personality prediction in the presence of correlated corruption. To achieve this, we designed a heuristic hard thresholding method to estimate the corruption set along with global consensus or majority voting strategies, that is alternately updated with the optimized regression coefficients. We demonstrate that our algorithm can handle a general multi-factor robust regression problem in the property of correlated corruption with a strong recovery guarantee on regression coefficients in a geometric convergence rate. Extensive experiments on both synthetic data and real-world data demonstrated that the proposed algorithm outperforms other comparable methods in both effectiveness and efficiency.

Chapter 8

Conclusions and Future Work

The proposed research aims on the development of scalable robust models under adversarial data corruption. We are focused on six major types of approaches, including robust regression via heuristic hard-thresholding, online and distributed robust regression under adversarial data corruption, robust regression via online feature selection with adversarial data corruption, self-paced robust learning for leveraging clean labels in noisy data, distributed self-paced learning in ADMM and robust multi-factor personality prediction with correlated data corruption. For the problem of robust regression via heuristic hard-thresholding, we proposed a new model, Robust Least squares regression algorithm via Heuristic Hard thresholding. The main contributions of our study are summarized as follows: 1) The design of an efficient algorithm to address the RLSR problem without parameterizing its corruption. The algorithm RLHH is proposed to recover the regression coefficients and uncorrupted set efficiently. Unlike with a fixed corruption ratio, our method alternately estimates the optimal corruption ratio based on residual errors using optimized regression coefficients in each iteration. 2) An exact recovery guarantee under a mild assumption regarding input variables. We prove that our RLHH algorithm converges at a geometric rate and recovers $\boldsymbol{\beta}^*$ exactly under the assumption that the least squares function satisfies both the Subset Strong Convexity and Subset Strong Smoothness properties. Extensive empirical studies were conducted with 6 competing methods in synthetic data. The results demonstrate that our approach consistently outperforms existing methods in both regression coefficients and uncorrupted set recovery, delivering a competitive running time. Future work will further apply this method in 1) robust classification tasks; and 2) various heuristic methods in determining the corrupted data instances.

For the problem of online and distributed robust regression under adversarial data corruption, This paper 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. Furthermore, an online ver-

sion of the distributed algorithm is proposed to incrementally update the existing estimates with new incoming data. 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. Further work will extend to the utilization of different methods to determine the size of restored models.

For the problem of robust regression via online feature selection, this work This paper proposes a novel robust regression algorithm via online feature selection (RoOFS) that concurrently addresses all the above challenges. Specifically, the algorithm iteratively updates the regression coefficients and the uncorrupted set via a robust online feature substitution method. We also prove that our algorithm has a restricted error bound compared to the optimal solution. Extensive empirical experiments demonstrated that the effectiveness of our new method is superior to that of existing methods in the recovery of both feature selection and regression coefficients, with very competitive efficiency. Future work will be extended to new scalable online feature selection methods.

For the problem of self-paced robust learning of leveraging the information contained in the clean data, we propose a novel self-paced robust learning algorithm (SPRL) that trains the model in a process from more reliable (clean) data instances to less reliable (noisy) ones under the supervision of well-labeled data. The self-paced learning process hedges the risk of selecting corrupted data into the training set. Moreover, theoretical analysis on the convergence of the proposed algorithm is provided under mild assumptions. Extensive experiments on synthetic and real datasets demonstrate that our approach can outperform the other existing methods in effectiveness and robustness. For the future work, the plan is to extend the self-paced learning algorithm by assuming different data distributions.

For the problem of distributed self-paced learning in ADMM. In this paper, a distributed self-paced learning algorithm (DSPL) is proposed to extend the traditional SPL algorithm to its distributed version for large scale datasets. To achieve this, we reformulated the original SPL problem into a distributed setting and optimized the problem of treating different mini-batches in parallel based on consensus ADMM. We also proved that our algorithm can be convergent under mild assumptions. Extensive experiments on both synthetic data and real-world rental price data demonstrated that the proposed algorithms are very effective, outperforming the other comparable methods over a range of different data settings.

For the problem of robust multi-factor personality prediction in correlated data corruption, this work propose a novel robust multi-factor personality prediction model that concurrently addresses all the above challenges by developing a distributed robust regression algorithm. Specifically, the algorithm optimizes regression coefficients of each factor in parallel with a heuristically estimated corruption ratio and then consolidates the uncorrupted set from multiple factors in two strategies: global consensus and majority voting. We also prove that our algorithm benefits from strong guarantees in terms of convergence rates and coefficient

recovery, which can be utilized as a generic framework for the multi-factor robust regression problem with correlated corruption property. Extensive experiment on synthetic and real dataset demonstrates that our algorithm is superior to those of existing methods in both effectiveness and efficiency. For the future work, the plan is to extend the correlated corruption framework by automatically determining the size of corruption correlation for each data instances.

8.1 Contributions

The major research tasks are described as follows. The current status of these tasks is listed in Table 8.1.

8.1.1 Robust Regression via Heuristic Hard-Thresholding

- **Proposing efficient algorithms to address the *RLSR* problem (A1)** 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 (A2)** 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 (A3)** Provide a proof that our algorithms benefit from strong guarantees analogous to those state-of-the-art methods in terms of convergence rates and recovery guarantees. Specifically, our algorithm can provide a small error bound when the error function satisfy the subset restricted strong convexity and subset restricted strong smoothness.

Table 8.1: Research tasks and status

Task	**Description**	**Status**
Research Area A	Robust Regression via Heuristic Hard-Thresholding	Completed
Research Area B	Online and Distributed Robust Regression	Completed
Research Area C	Robust Regression via Online Feature Selection	
C1	Propose RoOFS algorithm for robust regression in online feature	Completed
C2	Prepare one new real-world dataset	Completed
C3	Validate performance of RoOFS algorithm on the new dataset	Completed
Research Area D	Self-Paced Robust Learning	
D1	Formulate a framework to leverage the clean labels in noisy data	Completed
D2	Propose a self-paced robust learning algorithm to train models under the supervision of clean labels	Completed
D3	Provide a theoretical analysis for the convergence of the proposed algorithm	Completed
D4	Conducting extensive experiments for performance evaluations	Completed
Research Area E	Distributed Self-Paced Robust Learning	
E1	Formulate the distributed self-paced problem	Completed
E2	Propose a distributed self-paced learning algorithm based on consensus ADMM	Completed
E3	PProvide the theoretical analysis for the convergence of the proposed algorithm	Completed
E4	Conduct extensive experiments to evaluate the proposed algorithm	Completed
Research Area F	Robust Learning in Correlated Data Corruption	
F1	Propose a Distributed Robust Algorithm for the Multi-Factor Regression Problem	Completed
F2	Provide a strong recovery guarantee under the multi-factor problem setting	Completed
F3	Conduct extensive experiments for performance evaluations	Completed
Thesis Revision F	Thesis Revision	Completed

8.1.2 Online and Distributed Robust Regression

- **Propose novel distributed and online algorithms for robust regression (B1)** 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.

- **Design an adaptive algorithm to determine dominating set size (B2)** We propose a novel robust consolidation method to determine the dominating set of both the distributed and online robust methods. Specifically, the dominating set will be made of the batches which are close to the pivot batch (a uncorrupted batch estimation).

- **Providing a rigorous robustness guarantee for regression coefficient recovery. (B3)** 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 enough.

- **Extension of the experiments in scalability (B4)** The proposed method was 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.

8.1.3 Robust Regression via Online Feature Selection

- **Propose RoOFS algorithm for robust regression in online feature (C1)** The algorithm *RoOFS* is proposed to recover the regression coefficients and uncorrupted set efficiently. Unlike using entire features, our approach alternately estimates the data corruption and selects the feature set via a robust online feature substitution method.

- **Provide theoretical analysis of the algorithm (C2)** We prove that our method yields a solution with a restricted error bound compared to ground truth coefficients under the Subset Restricted Strong Convexity (*SRSC*) property. The error bound demonstrate that our algorithm has good theoretical property in a mild condition.

- **Validate performance of RoOFS algorithm on sufficient datasets (C3)** Our proposed algorithm was evaluated with 6 competing methods in both robust regression and online feature selection literatures. The results showed that our approach consistently outperforms existing methods in coefficients recovery and uncorrupted set estimation, delivering a competitive running time.

8.1.4 Self-Paced Robust Learning

- **Formulate a framework to leverage the clean labels in noisy data. (D1)** In this research, we focus on utilizing clean labels in a large-scale noisy dataset. Specifically, the clean labels are assumed to contain a limited size of data while the noisy dataset may contain an extremely large amount of data corruption. The approaches in solving robust regression and classification tasks are presented, which demonstrates the proposed framework can be generally used in various tasks.
- **Propose a self-paced robust learning algorithm to train models under the supervision of clean labels. (D2)** The proposed self-paced algorithm learns the data samples from clean to noisy under the supervision of clean labels, which helps to hedge the risk of involving corrupted data into the training set. Furthermore, the algorithm learns the training data in an order that are dynamically determined by the feedback of the learner itself without additional prior knowledge, which makes it more extensively utilized in practice.
- **Provide a theoretical analysis for the convergence of the proposed algorithm. (D3)** We prove that our self-paced robust learning algorithm converges under the assumption that the loss function selected for the estimated model has a finite lower bound. Specifically, the objective function of our algorithm can be monotonically decreased in accord with the increasing learning pace parameter until it reaches the lower bound.
- **Conduct extensive experiments for performance evaluations. (D4)** The proposed method was evaluated on both synthetic data and real-world datasets in robust regression and classification tasks with different corruption and data-size settings. The results demonstrate that the proposed approaches consistently outperform existing methods along multiple metrics.

8.1.5 Distributed Self-Paced Learning

- **Formulate the distributed self-paced problem (E1)** We reformulate the self-paced problem into a distributed setting. Specifically, an auxiliary variable is introduced to decouple the dependency of the model parameters for each data batch
- **Propose a distributed self-paced learning algorithm based on consensus ADMM (E2)** A distributed self-paced learning algorithm based on consensus ADMM is proposed to solve the *SPL* problem in a distributed setting. The algorithm optimizes the model parameters for each batch in parallel and consolidates their values in each iteration.
- **Provide the theoretical analysis for the convergence of the proposed algorithm (E3)** A theoretical analysis is provided for the convergence of our proposed

DSPL algorithm. The proof shows that our new algorithm will converge under mild assumptions, e.g., the loss function can be non-convex.

- **Conduct extensive experiments to evaluate the proposed algorithm (E4)** Extensive experiments have been conducted utilizing both synthetic and real-world data based on a robust regression task. The results demonstrate that the proposed approaches consistently outperform existing methods for multiple data settings.

8.1.6 Robust Learning in Correlated Data Corruption

- **Propose a distributed robust algorithm for the multi-factor regression problem. (F1)** The optimization of the proposed multi-factor model is a non-convex discrete optimization problem, which is technically challenging. By optimizing individual factor in parallel, the uncorrupted set is combined from each factor following two strategies: global consensus and majority voting.

- **Provide a strong recovery guarantee under the multi-factor problem setting (F2)** We prove that our RMFP algorithm with global consensus strategy converges at a geometric rate and recovers coefficients of each factor exactly under the assumption of Subset Strong Convexity and Subset Strong Smoothness properties. Specifically, we prove that our algorithm ensures a rigorous error bound of the regression coefficient compared to ground truth.

- **Conduct extensive experiments for performance evaluations (F3)** The proposed method was evaluated on both synthetic data and real-world datasets with various corruption settings. The results demonstrate that the proposed approach runs efficiently and consistently outperforms the best of the existing methods along multiple metrics.

8.2 Publications

8.2.1 Paper Published or Accepted

1. **Xuchao Zhang**, Fanglan Chen, Chang-Tien Lu, Naren Ramakrishnan, "**Mitigating Uncertainty in Document Classification**", Annual Conference of the North American Chapter of the Association for Computational Linguistics (**NAACL'19**), Minneapolis, Minnesota, June 2-7, 2019.

2. **Xuchao Zhang**, Shuo Lei, Liang Zhao, Arnold Boedihardjo, Chang-Tien Lu, "Robust Regression via Heuristic Corruption Thresholding and Its Adaptive Estimation Variation", ACM Transactions on Knowledge Discovery from Data (**TKDD**), 2019.

3. Lei Zhang, Liang Zhao, **Xuchao Zhang**, Wenmo Kong, Zitong Sheng, Chang-Tien Lu, "**Situation-Based Interaction Learning for Personality Prediction on Facebook**", Proceedings of the IEEE International Conference on Big Data, Seattle, WA, Dec. 10-13, 2018.

4. **Xuchao Zhang**, Shuo Lei, Liang Zhao, Arnold Boedihardjo, Chang-Tien Lu, "**Robust Regression via Online Feature Selection under Adversarial Data Corruption**", Proceedings of the IEEE International Conference on Data Mining (**ICDM'18**), Singapore, Nov. 17-20, 2018. [acceptance rate: 19.94%]

5. Zhiqian Chen, Feng Chen, Rongjie Lai, **Xuchao Zhang**, and Chang-Tien Lu, "**Rational Neural Networks for Approximating Jump Discontinuities of Graph Convolution Operator**", Proceedings of the IEEE International Conference on Data Mining (**ICDM'18**), Singapore, Nov. 17-20, 2018. [acceptance rate: 8.86%]

6. **Xuchao Zhang**, Liang Zhao, Zhiqian Chen, Chang-Tien Lu, "**Distributed Self-Paced Learning in Alternating Direction Method of Multipliers**", Proceeding of the 27th International Joint Conference on Artificial Intelligence (**IJCAI'18**), Stockholm, Sweden, July 13-19, 2018. [acceptance rate: 20.6%]

7. **Xuchao Zhang**, Liang Zhao, Arnold P. Boedihardjo, Chang-Tien Lu, "Online and Distributed Robust Regressions under Adversarial Data Corruption", Proceedings of the IEEE International Conference on Data Mining (ICDM'17), pages 625-634, New Orleans, Louisiana, Nov. 18-21, 2017. [acceptance rate: 9.25%].

8. **Xuchao Zhang**, Liang Zhao, Arnold P. Boedihardjo, Chang-Tien Lu, Naren Ramakrishnan, "**Spatiotemporal Event Forecasting from Incomplete Hyper-local Price Data**", Proceedings of the 26th ACM International Conference on Information and Knowledge Management (**CIKM'17**), pages 507-516, Singapore, Nov. 6-10, 2017. [acceptance rate: 21%].

9. **Xuchao Zhang**, Liang Zhao, Zhiqian Chen, Arnold Boedihardjo, Dai Jing, Chang-Tien Lu, **Trendi: Tracking Stories in News and Microblogs via Emerging, Evolving and Fading Topics**, Proceedings of the IEEE International Conference on Big Data (**BigData'17**), Boston, MA, Dec. 11-14, 2017.

10. **Xuchao Zhang**, Zhiqian Chen, Liang Zhao, Arnold Boedihardjo, Chang-Tien Lu, **TRACES: Generating Twitter Stories via Shared Subspace and Temporal Smoothness**, Proceedings of the IEEE International Conference on Big Data (**BigData'17**), Boston, MA, Dec. 11-14, 2017.

11. **Xuchao Zhang**, Liang Zhao, Arnold P. Boedihardjo, Chang-Tien Lu, **Robust Regression via Heuristic Hard Thresholding**, Proceeding of the 26th International Joint Conference on Artificial Intelligence (**IJCAI'17**), pages 34343440, Melbourne, Australia, August 19-25, 2017. [acceptance rate: 26%].

12. Zhiqian Chen, **Xuchao Zhang**, Arnold P. Boedihardjo, Jing Dai, Chang-Tien Lu, **Multimodal Storytelling via Generative Adversarial Imitation Learning**, Proceeding of the 26th International Joint Conference on Artificial Intelligence (**IJCAI'17**), pages 3967-3973, Melbourne, Australia, August 19-25, 2017. [acceptance rate: 26%].

13. **Xuchao Zhang**, Zhiqian Chen, Weisheng Zhong, Arnold P. Boedihardjo, Chang-Tien Lu, **Storytelling in Heterogeneous Twitter Entity Network based on Hierarchical Cluster Routing**, Proceedings of the IEEE International Conference on Big Data (**BigData'16**), pp. 1522-1531 Washington, DC, Dec. 5-8, 2016.

14. Ting Hua, **Xuchao Zhang**, Wei Wang, Chang-Tien Lu, Naren Ramakrishnan, **Automatical Storyline Generation with Help fromTwitter**, Proceedings of the 25th ACM International Conference on Information and Knowledge Management(**CIKM'16**), Indianapolis, IN, Oct. 24-28, 2016. [acceptance rate: 28.8%].

8.2.2 Paper in Submission or Preprint

1. Xuchao Zhang*, Yifeng Gao*, Jessica Lin, Chang-Tien Lu, "**TapNet: Multivariate Time Series Classification with Attentional Prototype Network**", 2019.

2. **Xuchao Zhang**, Dheeraj Rajagopal, Sujay Jauhar, Michael Gamon and Chang-Tien Lu, "**Modeling the Relationship between User Comments and Edits in Document Revision**", 2019.

3. Dheeraj Rajagopal, **Xuchao Zhang**, Sujay Kumar Jauhar, Michael Gamon and Eduard Hovy, "**One Document, Many Revisions: From Classification to Summarization of Edits**", 2019.

4. **Xuchao Zhang**, Liang Zhao, Zhiqian Chen, Chang-Tien Lu, "**Self-Paced Robust Learning for Leveraging Clean Labels in Noisy Data**", 2018.

5. **Xuchao Zhang**, Lei Zhang, Liang Zhao, Arnold P. Boedihardjo, Chang-Tien Lu, "**Robust Multi-Factor Personality Prediction with Correlated Data Corruption in Social Media**", 2018.

6. Taoran Ji, Kaiqun Fu, Liang Zhao, **Xuchao Zhang**, Chang-Tien Lu, Naren Ramakrishnan, "Multi-task Feature Learning for Cybersecurity Event Detection in Social Media", 2018.

7. Bingsheng Wang*, **Xuchao Zhang***, Chang-Tien Lu, Feng Chen, "**Water Disaggregation via Shape Features based Bayesian Discriminative Sparse Coding**", *These two authors contributed equally, 2018.

8.3 Future Research Directions

8.3.1 Robust regression via heuristic corruption thresholding

1) Consideration of structural data corruption. The current setting of adversarial data corruption is based on the individual data instance. We will try to consider if the data corruption has some correlated impact on the data instances.

2) Extensions to the classification problem. We will extend the heuristic corruption thresholding me to the robust classification problem. The measurement of the heuristic function will be updated by calculating the distance between the between the distance between

3) Utilize on more real-world adversarially corrupted data set. So far the adversarial data corruption we used are generated synthetically, which always assume the data corruption follows some certain distribution. We will try to utilize our model in some real world data attacks in the application such as the network security or adversarial data attack in image data.

8.3.2 Robust Multi-variate Time Series Classification

1) Robust learning in multi-variate time series. Time series data usually contains the data corruption which will harm the performance of existing classifiers. We will propose a robust model to handle the data corruption in the following two types: data noise in the time series features and the noise in instance labels.

2) Prototypical Network for small size of training data. Most of time series dataset contain a small amount of training data. The deep learning based method usually fail in such small size of data set. We will propose a distance-based attentional prototypical network to handle the limited training samples.

8.3.3 Data-Driven Curriculum Robust Learning

1) Developing a data-driven curriculum model. We will try to train a data-driven curriculum model for the clean set to guide the training of the large but noisy set. Specifically, the curriculum model will be learned by not only the instance weight transition but the instance feature. The trained curriculum model will give the score of robustness of data instances in the noisy dataset.

2) Developing a distributed model. Jointly train the curriculum model and the target model is a time consuming iterative process. Thus, we plan to propose new scalable approaches based on our existing distributed framework.

Appendices

Appendix A

Theoretical Analysis on Robust Regression via Heuristic Corruption Thresholding

A.1 Error Bound Analysis on RHCT

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 7. *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 $t = \mathcal{O}\left(\log_{\frac{1}{\eta}} \frac{\|\boldsymbol{u}\|_2}{\sqrt{n}\epsilon}\right)$ iterations, Algorithm 2 yields an ϵ-accurate solution $\boldsymbol{\beta}_t$ with $\|\boldsymbol{\beta}_* - \boldsymbol{\beta}_t\|_2 \leq \epsilon + \frac{C\|\boldsymbol{\varepsilon}\|_2}{\sqrt{n}}$ for some $C > 0$.*

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

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

Using the triangle inequality of the L_2 norm, we have

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

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

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

Inequality *(c)* follows from the Lemma 2, 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

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

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

A.2 Proof of Convergence of RHCT

Proof. Before we prove the convergence of Algorithm 5, we will first show that the value of objective function $\mathcal{J}$ is monotonically decreased. Objective function $\mathcal{J}$ has the following property:

$$\begin{aligned}&\mathcal{J}(\boldsymbol{w}^{t+1}, \boldsymbol{v}^{t+1}; \lambda^{t+1}) \\ &\quad \overset{(a)}{\leq} \sum_{i=1}^{k}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) + \sum_{i=k+1}^{n} v_i^{t+1}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) \\ &\qquad + \|\boldsymbol{w}^{t+1}\|_2^2 + \theta\|\boldsymbol{w}^{t+1} - \tilde{\boldsymbol{w}}\|_2^2 - \lambda^t\sum_{i=k+1}^{n} v_i^{t+1}\end{aligned}$$

The inequality follows from the fact that λ increases monotonically so that $\lambda^{t+1} \geq \lambda^t$ and $v_i^t \geq 0$. The optimization step in Line 7 in Algorithm 5 guarantees the following property:

$$\begin{aligned}\sum_{i=k+1}^{n} v_i^{t+1}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) - \lambda^t \sum_{i=k+1}^{n} v_i^{t+1} \\ \leq \sum_{i=k+1}^{n} v_i^{t}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) - \lambda^t \sum_{i=k+1}^{n} v_i^{t}\end{aligned}$$

Therefore, we have the following property:

$$\begin{aligned}&\mathcal{J}(\boldsymbol{w}^{t+1}, \boldsymbol{v}^{t+1}; \lambda^{t+1}) \\ &\quad\leq \sum_{i=1}^{k} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) + \sum_{i=k+1}^{n} v_i^{t}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t+1})\big) \\ &\quad\quad + \|\boldsymbol{w}^{t+1}\|_2^2 + \theta\|\boldsymbol{w}^{t+1} - \tilde{\boldsymbol{w}}\|_2^2 - \lambda^t \sum_{i=k+1}^{n} v_i^{t}.\end{aligned}$$

Similarly, the following inequality is satisfied since the optimizations step in Line 5 in Algorithm 5.

$$\begin{aligned}&\mathcal{J}(\boldsymbol{w}^{t+1}, \boldsymbol{v}^{t+1}; \lambda^{t+1}) \\ &\quad\leq \sum_{i=1}^{k} \mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t})\big) + \sum_{i=k+1}^{n} v_i^{t}\mathcal{L}\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w}^{t})\big) \\ &\quad\quad + \|\boldsymbol{w}^{t}\|_2^2 + \theta\|\boldsymbol{w}^{t} - \tilde{\boldsymbol{w}}\|_2^2 - \lambda^t \sum_{i=k+1}^{n} v_i^{t} \\ &\quad= \mathcal{J}(\boldsymbol{w}^{t}, \boldsymbol{v}^{t}; \lambda^{t})\end{aligned}$$

Since the objective function is monotonically decreased and it has a lower bound according to Lemma 1, we have $\|\mathcal{J}^{t+1} - \mathcal{J}^t\|_2 < \varepsilon$ for $\forall \varepsilon > 0$. □

Appendix B

Online and Distributed Robust Regression with Adversarial Noises

This this section, we extend the online and distributed robust regression algorithm to handle the another type of adversarial data corruption: the unbalanced order of currupted data batches in Section B.1. Then we present the experimental results in Section B.2.

B.1 Adversarial Online Robust Regression

The ORLR algorithm updates the robust estimate based on new incoming data by removing the oldest one from Ψ^*, which is the set of estimates that are not included in dominating. In general, it can deal with the common adversarial corruption, which in one mini-batch is greater than 50% and produced by different kinds of attack. However, when the data corruption is produced by malicious code, it is infeasible to estimate by using ORLR. Table B.2 shows the performance of regression coefficient recovery in different settings of malicious corrupted mini-batches. From Table B.2 we can find that the sequence of corrupted mini-batches is very important to ORLR. When uncorrupted mini-batches first arrive, it can establish better deterministic set. However, when malicious corrupted mini-batches comes earlier, it will establish wrong deterministic set which will effect the performance of left data. To solve this problem, a novel adversarial online robust regression method is proposed to estimate the malicious corrupted data set.

Before introducing the details of the new proposed method, we will point out the key difference between corrupted data and malicious corrupted data. That is the distribution of corruption and its maker. Specifically, common online data corruption usually happened during data transmission and there will be small and random noise in the data. Instead, malicious corrupted data were produced by attackers, that means the noise added in the data will be large and similar in order to disguise as real data. Suppose over half of the

previous m mini-batches data were corrupted by malicious attackers. The previous pivot batch p and deterministic set Π will be established by malicious data. Under this extreme circumstance, the regression coefficients will not be updated due to new incoming mini-batch estimate $\boldsymbol{\beta}^+ \notin \{\Psi^* \cup \Psi^+\}$, which is case 3 shown in Figure 3.2.

In order to efficiently solve the problems, we proposed new method to update the set of estimates Π. In other words, it's to select the removed estimate $\boldsymbol{\beta}^{(s)}$ from Π while the new estimate $\boldsymbol{\beta}^+$ is appended to the tail of Π. As for adversarial online algorithm, the sequence of incoming data is very important because the previous invocation can be used as the imput of the next one. Specifically, the distribution of corrupted data can be concentrated or separated. As for concentrated circumstance, when corrupted data coming has an influence on robust estimates. Moreover, if corrupted data gathered in the front part of mini-batches, deterministic set will be wrongly established due to its similar regression coefficients. Therefore, a novel selective method is proposed to remove the one from previous mini-batches $\boldsymbol{\Pi}$ considering both deterministic set and time index, which is formally defined as

$$\boldsymbol{v}(\boldsymbol{\beta}^{(i)}) = \boldsymbol{\mu}(\frac{\|\hat{\boldsymbol{\beta}}' - \boldsymbol{\beta}^{(i)}\|_2}{\sum_{i\in\Pi'}\|\hat{\boldsymbol{\beta}}' - \boldsymbol{\beta}^{(i)}\|_2}) + \boldsymbol{\lambda}(\frac{i}{\sum_{i=1}^{m} i}) \tag{B.1}$$

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 *ARLR* are shown in Algorithm 8. In Line 1, the regression coefficients $\boldsymbol{\beta}^+$ of the new data is optimized by *HRR* algorithm. In Line 3, get the robust consolidation $\hat{\boldsymbol{\beta}}'$ from previous mini-batch set Π. Lines 4 and 5 calculate 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. Since new estimates are appended to the tail of Π, the usage of selective method ensures that the malicious 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 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 *ARLR* 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.

ALGORITHM 8: ARLR 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 Ψ. Pivot related hyperparameters μ and time related hyperpatameters λ

Output: solution $\hat{\boldsymbol{\beta}}$, Π, Ψ

$\boldsymbol{\beta}^+ \leftarrow \mathrm{HRR}(X^+, \boldsymbol{y}^+)$

$\Pi' = \Pi \cup \{\boldsymbol{\beta}^+\}$

$\hat{\boldsymbol{\beta}}' = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{m} \sum_{i\in\Psi} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2 \right\}$ *// Robust consolidation $\hat{\boldsymbol{\beta}}'$ in previous mini-batch*

for *i = 1..n* **do**

  $\boldsymbol{V}(\boldsymbol{\beta}^{(i)}) = \boldsymbol{\mu}(\frac{\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)}\|_2}{\sum_{i\in\Pi'} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(i)}\|_2}) + \boldsymbol{\lambda}(\frac{i}{\sum_{i=1}^{m} i})$ *// Update the residual value for each data sample.*

end

$s \leftarrow \min(V)$ *// Select removed estimate s*

$\Pi^+ = \Pi \setminus \{\boldsymbol{\beta}^{(s)}\} \cup \{\boldsymbol{\beta}^+\}$

$p^+ = \arg\min_i \sigma_{\tilde{m}}(\boldsymbol{d}^{(i)})$ *// Optimize new pivot batch p^+*

$\Psi^+ = \left\{ \delta_k(\boldsymbol{d}^{(p^+)}) | 1 \le k \le \tilde{m} \right\}$ *// Find new deterministic set Ψ^+*

$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{\tilde{m}} \sum_{i\in\Psi^+} \|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}\|_2 \right\}$ *// Robust consolidation*

return $\hat{\boldsymbol{\beta}}, \Pi^+, \Psi^+$

B.2 Performance of ARLR algorithm

B.2.1 Performance on different sequence of malicious corrupted mini-batches

Table B.2 shows the performance of malicious corrupted coefficient recovery in different sequence of data mini-batches, and the corrupted ratio was from 0 to 40% in total. As for the sequence settings, it was mainly divided into three parts: uncorrupted data arriving first,

Table B.1: 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
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**

Table B.2: Performance on Regression Coefficients Recovery in Different Sequence of Malicious Corrupted Mini-batches.

	Uncorrupted Batches First, p=100, n=10K, b=30				
	0%	10%	20%	30%	40%
OLS-AVG	0.089	0.109	0.108	0.108	0.107
RLHH-AVG	0.055	0.078	0.078	0.078	0.077
OPAA	5.181	5.145	5.166	5.158	5.161
ORL-H	0.280	0.295	0.294	0.291	0.292
ORL*	0.108	0.142	0.186	0.190	0.229
ORLR	0.042	0.043	0.143	0.167	0.203
DRLR	**0.035**	**0.035**	**0.035**	**0.035**	**0.035**
ARLR	**0.044**	**0.042**	**0.042**	**0.042**	**0.042**
	Random Order, p=100, n=10K, b=30				
	0%	10%	20%	30%	40%
OLS-AVG	0.071	0.127	0.186	0.241	0.296
RLHH-AVG	0.034	0.100	0.168	0.234	0.300
OPAA	5.153	5.221	5.141	5.193	5.203
ORL-H	0.278	0.298	0.316	0.357	0.397
ORL*	0.083	0.113	0.154	0.224	0.341
ORLR	0.038	0.038	0.038	0.237	0.168
DRLR	**0.034**	**0.034**	**0.034**	**0.034**	**0.034**
ARLR	**0.037**	**0.037**	**0.038**	**0.037**	**0.038**
	Corrupted Batches First, p=100, n=10K, b=30				
	0%	10%	20%	30%	40%
OLS-AVG	0.071	0.125	0.178	0.232	0.297
RLHH-AVG	0.034	0.100	0.163	0.226	0.301
OPAA	5.140	5.147	5.247	5.134	5.165
ORL-H	0.273	0.284	0.298	0.312	0.343
ORL*	0.083	0.094	0.107	0.149	0.248
ORLR	0.042	0.043	0.705	0.704	0.703
DRLR	**0.034**	**0.035**	**0.035**	**0.035**	**0.035**
ARLR	**0.038**	**0.038**	**0.038**	**0.038**	**0.038**

arriving randomly and corrupted data arriving fist. Each corrupted mini-batch used in the experiment contains 90% malicious corrupted samples and each uncorrupted mini-batch has 10% malicious 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 B.2, we conclude: 1) When some mini-batches are corrupted, the *DRLR* method outperforms all the competing methods, and *ARLR* achieves the best performance compared to other online methods. 2) Although *ORLR* has competitive performance in the uncorrupted arriving first settings, its recovery error increases dramatically when the corrupted data arriving first. It is reasonable that *ORLR* has time-vary characteristic. But the error of the *ARLR* 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 malicious corrupted mini-batches, but their 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.

B.2.2 Performance on different malicious corrupted mini-batches

Table B.3 shows the performance of regression coefficient recovery in different settings of malicious corrupted mini-batches. 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. From the result in Table B.3, we conclude: 1) When some mini-batches are malicious corrupted, the *DRLR* method outperforms all the competing methods, and *ARLR* achieves the best performance compared to other online methods. 2) When the number of corrupted mini-batches is smaller, the performance of *ORLR* is competitive. But when the number of malicious corrupted mini-batch increases, its recovery error is dramatically increased. Instead, *ARLR* performs consistently when the number of corrupted mini-batches increases. 3) *ORL** and *RLHH-AVG* have competitive performance in different settings of corrupted mini-batches. However, their recovery error still increases two times when the number of corrupted mini-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 *ARLR* were still best and the recovery error increased slowly.

Table B.3: Performance on Regression Coefficients Recovery in Different Malicious Corrupted Mini-batches.

	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
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**
ARLR	**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
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**
ARLR	**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	0.127	0.185	0.241	0.298	0.090	0.132	0.188	0.243	0.298
RLHH-AVG	0.068	0.101	0.168	0.235	0.302	0.082	0.110	0.175	0.240	0.306
OPAA	5.656	5.555	5.504	5.571	5.525	6.782	6.795	6.776	6.817	6.806
ORL-H	0.403	0.404	0.423	0.441	0.451	0.779	0.786	0.788	0.795	0.796
ORL*	0.104	0.119	0.149	0.218	0.335	0.234	0.298	0.438	0.576	0.701
ORLR	0.042	0.043	0.705	0.704	0.703	0.100	0.100	0.706	0.704	0.704
DRLR	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**	**0.067**	**0.067**	**0.067**	**0.068**	**0.068**
ARLR	**0.041**	**0.043**	**0.044**	**0.044**	**0.043**	**0.100**	**0.100**	**0.101**	**0.100**	**0.101**

Bibliography

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

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

[3] A. Ben-Tal, D. Den Hertog, A. De Waegenaere, B. Melenberg, and G. Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[5] K. Bhatia, P. Jain, and P. Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, pages 721–729, 2015.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[7] D. C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90, 2008.

[8] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. *Knowledge and information systems*, 34(1):23–54, 2013.

[9] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive training of deformable part models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1832–1839. IEEE, 2011.

[10] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Optics-of: Identifying local outliers. *Principles of data mining and knowledge discovery*, pages 262–270, 1999.

[11] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[12] J. D. Burger, J. Henderson, G. Kim, and G. Zarrella. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309. Association for Computational Linguistics, 2011.

[13] K. Buza. Feedback prediction for blogs. In M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning, editors, *Data Analysis, Machine Learning and Knowledge Discovery*, pages 145–152, Cham, 2014. Springer International Publishing.

[14] G. V. Caprara, C. Barbaranelli, L. Borgogni, and M. Perugini. The big five questionnaire: A new questionnaire to assess the five factor model. *Personality and Individual Differences*, 15(3):281 – 288, 1993.

[15] R. Carciofo, J. Yang, N. Song, F. Du, and K. Zhang. Psychometric evaluation of chinese-language 44-item and 10-item big five personality inventories, including correlations with chronotype, mindfulness and mind wandering. *PLOS ONE*, 11(2):1–26, 02 2016.

[16] F. Celli, E. Bruni, and B. Lepri. Automatic personality and interaction style recognition from facebook profile pictures. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1101–1104. ACM, 2014.

[17] Y. Chen and C. Caramanis. Noisy and missing data regression: Distribution-oblivious support recovery. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 383–391. JMLR Workshop and Conference Proceedings, 2013.

[18] Y. Chen, C. Caramanis, and S. Mannor. Robust sparse regression under adversarial corruption. In *ICML (3)*, pages 774–782, 2013.

[19] Y. Chen, C. Caramanis, and S. Mannor. Robust sparse regression under adversarial corruption. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 774–782. JMLR Workshop and Conference Proceedings, May 2013.

[20] T. Correa, A. W. Hinsley, and H. G. De Zuniga. Who interacts on the web?: The intersection of users personality and social media use. *Computers in Human Behavior*, 26(2):247–253, 2010.

[21] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[22] C. Cromvik and M. Patriksson. 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):461–478, 2010.

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

[24] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[25] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.

[26] X. V. Doan, S. Kruk, and H. Wolkowicz. A robust algorithm for semidefinite programming. *Optimization Methods and Software*, 27(4-5):667–693, 2012.

[27] G. Domino and M. L. Domino. *Psychological testing: An introduction*. Cambridge University Press, 2006.

[28] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[29] J. Dupačová and M. Kopa. Robustness in stochastic programs with risk constraints. *Annals of Operations Research*, 200(1):55–74, 2012.

[30] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 52(8):2275–2285, 2004.

[31] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM.

[32] Y. Fan, R. He, J. Liang, and B.-G. Hu. Self-paced learning: an implicit regularization perspective. In *AAAI*, 2017.

[33] J. Feng, H. Xu, and S. Mannor. Outlier robust online learning. *CoRR*, abs/1701.00251, 2017.

[34] J. Feng, H. Xu, S. Mannor, and S. Yan. Robust logistic regression and classification. In *Advances in neural information processing systems*, pages 253–261, 2014.

[35] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

[36] S. D. Gosling, P. J. Rentfrow, and W. B. Swann. A very brief measure of the big-five personality domains. *Journal of Research in personality*, 37(6):504–528, 2003.

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

[38] G. C. M. D. C. Harman. Quantifying mental health signals in twitter. *ACL 2014*, 51, 2014.

[39] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[40] M. Hong, Z.-Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[41] P. J. Huber and E. M. Ronchetti. *The Basic Types of Estimates*, pages 45–70. John Wiley & Sons, Inc., 2009.

[42] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 547–556. ACM, 2014.

[43] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

[44] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. Hauptmann. Self-paced curriculum learning, 2015.

[45] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[46] W. Jiang, G. Er, Q. Dai, and J. Gu. Similarity-based online feature selection in content-based image retrieval. *IEEE Transactions on Image Processing*, 15(3):702–712, 2006.

[47] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 293–298, New York, NY, USA, 2001. ACM.

[48] Y. Jung, S. P. Lee, and J. Hu. Robust regression for highly corrupted response by shifting outliers. *Statistical Modelling*, 16(1):1–23, 2016.

[49] S.-C. Kang, T. S. Brisimi, and I. C. Paschalidis. Distribution-dependent robust linear optimization with applications to inventory control. *Annals of operations research*, 231(1):229–263, 2015.

[50] J. C. Keerthiram Murugesan. Self-paced multitask learning with shared knowledge. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2522–2528, 2017.

[51] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In J. Frnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 543–550. Omnipress, 2010.

[52] M. Kosinski, D. Stillwell, and T. Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[53] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[54] V. Lampos, N. Aletras, J. K. Geyti, B. Zou, and I. J. Cox. Inferring the socioeconomic status of social media users based on behaviour and language. In *European Conference on Information Retrieval*, pages 689–695. Springer, 2016.

[55] R. N. Landers and J. W. Lounsbury. An investigation of big five and narrow personality traits in relation to internet usage. *Comput. Hum. Behav.*, 22(2):283–293, Mar. 2006.

[56] L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

[57] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[58] C. Li, J. Yan, F. Wei, W. Dong, Q. Liu, and H. Zha. Self-paced multi-task learning. In *AAAI*, pages 2175–2181, 2017.

[59] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and J. Li. Learning from noisy labels with distillation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1928–1936, 2017.

[60] L. Liu, D. Preotiuc-Pietro, Z. R. Samani, M. E. Moghaddam, and L. H. Ungar. Analyzing personality through social media profile picture choice. In *ICWSM*, pages 211–220, 2016.

[61] X. Liu and T. Zhu. Deep learning for constructing microblog behavior representation to identify social media users personality. *PeerJ Computer Science*, 2:e81, Sept. 2016.

[62] P.-L. Loh and M. J. Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. In *Advances in Neural Information Processing Systems*, pages 2726–2734, 2011.

[63] V. Lourenco, A. M. Pires, and M. Kirst. Robust linear regression methods in association studies. *Bioinformatics*, 27(6):815–821, 2011.

[64] F. Ma, D. Meng, Q. Xie, Z. Li, and X. Dong. Self-paced co-training. In *International Conference on Machine Learning*, pages 2275–2284, 2017.

[65] Z. Ma, S. Liu, and D. Meng. On convergence property of implicit self-paced objective. *arXiv preprint arXiv:1703.09923*, 2017.

[66] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.

[67] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[68] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[69] R. Maronna, R. D. Martin, and V. Yohai. *Robust statistics*. John Wiley & Sons, Chichester. ISBN, 2006.

[70] G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, Oct 2010.

[71] R. R. McCrae and O. P. John. An introduction to the five-factor model and its applications. *Journal of Personality*, 60(2):175–215, 1992.

[72] B. McWilliams, G. Krummenacher, M. Lucic, and J. M. Buhmann. Fast and robust least squares estimation in corrupted linear models. In *Advances in Neural Information Processing Systems*, pages 415–423, 2014.

[73] B. Mcwilliams, G. Krummenacher, M. Lucic, and J. M. Buhmann. Fast and robust least squares estimation in corrupted linear models. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 415–423. Curran Associates, Inc., 2014.

[74] A. W. Meade and S. B. Craig. Identifying careless responses in survey data. *Psychological methods*, 17(3):437, 2012.

[75] B. Medlock and T. Briscoe. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 992–999, 2007.

[76] D. Meng, Q. Zhao, and L. Jiang. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015.

[77] G. J. Meyer, S. E. Finn, L. D. Eyde, G. G. Kay, K. L. Moreland, R. R. Dies, E. J. Eisman, T. W. Kubiszyn, and G. M. Reed. Psychological testing and psychological assessment: A review of evidence and issues. *American psychologist*, 56(2):128, 2001.

[78] K. Moore and J. C. McElroy. The influence of personality on facebook usage, wall postings, and regret. *Computers in Human Behavior*, 28(1):267 – 274, 2012.

[79] I. Naseem, R. Togneri, and M. Bennamoun. Robust regression for face recognition. *Pattern Recognition*, 45(1):104–118, 2012.

[80] R. W. Naylor, C. P. Lamberton, and P. M. West. Beyond the like button: The impact of mere virtual presence on brand evaluations and purchase intentions in social media settings. *Journal of Marketing*, 76(6):105–120, 2012.

[81] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014.

[82] N. H. Nguyen and T. D. Tran. Exact recoverability from dense corrupted observations via l1-minimization. *IEEE transactions on information theory*, 59(4):2017–2035, 2013.

[83] D. S. Ones and C. Viswesvaran. The effects of social desirability and faking on personality and integrity assessment for personnel selection. *Human performance*, 11(2-3):245–269, 1998.

[84] E. S. Orr, M. Sisic, C. Ross, M. G. Simmering, J. M. Arseneault, and R. R. Orr. The influence of shyness on the use of facebook in an undergraduate sample. *CyberPsychology & Behavior*, 12(3):337–340, 2009.

[85] M. Pennacchiotti and A.-M. Popescu. A machine learning approach to twitter user classification. *Icwsm*, 11(1):281–288, 2011.

[86] S. Perkins and J. Theiler. Online feature selection using grafting. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 592–599, 2003.

[87] S. Perkins and J. Theiler. Online feature selection using grafting. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 592–599. AAAI Press, 2003.

[88] T. Pi, X. Li, Z. Zhang, D. Meng, F. Wu, J. Xiao, and Y. Zhuang. Self-paced boost learning for classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1932–1938. AAAI Press, 2016.

[89] D. Preoţiuc-Pietro, V. Lampos, and N. Aletras. An analysis of the user occupational class through Twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL '15, pages 1754–1764, 2015.

[90] D. Preotiuc-Pietro, M. Sap, H. A. Schwartz, and L. Ungar. Mental illness detection at the world well-being project for the clpsych 2015 shared task. *NAACL HLT 2015*, page 40, 2015.

[91] M. Rosenbaum, A. B. Tsybakov, et al. Sparse recovery under matrix uncertainty. *The Annals of Statistics*, 38(5):2620–2651, 2010.

[92] V. Roth. Kernel fisher discriminants for outlier detection. *Neural computation*, 18(4):942–960, 2006.

[93] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

[94] P. J. ROUSSEEUW and K. VAN DRIESSEN. Computing lts regression for large data sets. *Data Mining and Knowledge Discovery*, 12(1):29–45, Jan 2006.

[95] S. Ryali and V. Menon. Feature selection and classification of fmri data using logistic regression with l1 norm regularization. *NeuroImage*, 47:S57, 2009.

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

[97] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[98] S. Sharma, S. Khare, and B. Huang. Robust online algorithm for adaptive linear regression parameter estimation and prediction. *Journal of Chemometrics*, 30(6):308–323, 2016. cem.2792.

[99] Y. She and A. B. Owen. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association*, 106(494):626–639, 2011.

[100] A. Smolic and J. R. Ohm. Robust global motion estimation using a simplified m-estimator approach. In *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*, volume 1, pages 868–871 vol.1, 2000.

[101] H. E. Solberg and A. Lahti. Detection of outliers in reference distributions: performance of horns algorithm. *Clinical chemistry*, 51(12):2326–2332, 2005.

[102] C. Studer, P. Kuppinger, G. Pope, and H. Bolcskei. Recovery of sparsely corrupted signals. *IEEE Transactions on Information Theory*, 58(5):3115–3130, 2012.

[103] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006.

[104] J. S. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2379–2386, 2013.

[105] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–847, 2017.

[106] A. Vinciarelli and G. Mohammadi. A survey of personality computing. *IEEE Transaction on Affective Computing*, 5(3):273–291, 2014.

[107] J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, and X. Wu. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):3029–3041, 2015.

[108] J. Wang, P. Zhao, S. C. Hoi, and R. Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, 2014.

[109] J. Wright and Y. Ma. Dense error correction via l1-minimization. *IEEE Trans. Inf. Theor.*, 56(7):3540–3560, July 2010.

[110] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2009.

[111] X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1159–1166, 2010.

[112] C. Xu, D. Tao, and C. Xu. Multi-view self-paced learning for clustering. In *IJCAI*, pages 3974–3980, 2015.

[113] A. Yang, A. Ganesh, S. Sastry, and Y. Ma. 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, Feb 2010.

[114] H. Yang, R. Fujimaki, Y. Kusumura, and J. Liu. Online feature selection: A limited-memory substitution algorithm and its asynchronous parallel variation. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1945–1954, New York, NY, USA, 2016. ACM.

[115] K. Yu, X. Wu, W. Ding, and J. Pei. Towards scalable and accurate online feature selection for big data. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 660–669. IEEE, 2014.

[116] K. Yu, X. Wu, W. Ding, and J. Pei. Scalable and accurate online feature selection for big data. *ACM Trans. Knowl. Discov. Data*, 11(2):16:1–16:39, Dec. 2016.

[117] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[118] D. Zhang, D. Meng, L. Zhao, and J. Han. Bridging saliency detection to weakly supervised object detection based on self-paced curriculum learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3538–3544. AAAI Press, 2016.

[119] X. Zhang, L. Cheng, and T. Zhu. Robust multivariate regression with grossly corrupted observations and its application to personality prediction. In *Proceedings of The 7th Asian Conference on Machine Learning*, pages 112–126, 2015.

[120] X. Zhang, L. Zhao, A. P. Boedihardjo, and C. Lu. Online and distributed robust regressions under adversarial data corruption. In *2017 IEEE International Conference on Data Mining (ICDM)*, volume 00, pages 625–634, Nov. 2017.

[121] X. Zhang, L. Zhao, A. P. Boedihardjo, and C.-T. Lu. Online and distributed robust regressions under adversarial data corruption. *2017 IEEE International Conference on Data Mining (ICDM)*, pages 625–634, 2017.

[122] X. Zhang, L. Zhao, A. P. Boedihardjo, and C.-T. Lu. Robust regression via heuristic hard thresholding. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, IJCAI'17. AAAI Press, 2017.

[123] X. Zhang, L. Zhao, A. P. Boedihardjo, and C.-T. Lu. Robust regression via heuristic hard thresholding. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3434–3440, 2017.

[124] X. Zhang, L. Zhao, A. P. Boedihardjo, C.-T. Lu, and N. Ramakrishnan. Spatiotemporal event forecasting from incomplete hyper-local price data. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 507–516, New York, NY, USA, 2017. ACM.

[125] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 733–742, Arlington, Virginia, United States, 2010. AUAI Press.

[126] L. Zhao, Q. Sun, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan. Multi-task learning for spatio-temporal event forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1503–1512, New York, NY, USA, 2015. ACM.

[127] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011.

[128] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streamwise feature selection. *Journal of Machine Learning Research*, 7(Sep):1861–1885, 2006.

[129] H. Zhu, H. Leung, and Z. He. A variational bayesian approach to robust sensor fusion based on student-t distribution. *Information Sciences*, 221(Supplement C):201 – 214, 2013.

[130] J. Zhu, N. Lao, and E. P. Xing. Grafting-light: fast, incremental feature selection and structure learning of markov random fields. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 303–312. ACM, 2010.

[131] M. Ziegler and M. Buehner. Modeling socially desirable responding and its effects. *Educational and Psychological Measurement*, 69(4):548–565, 2009.

[132] M. Ziegler, C. MacCann, and R. Roberts. *New perspectives on faking in personality assessment*. Oxford University Press, 2011.

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

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