LI, MINGYAN, Ph.D. Penalized Weighted Methods for Robust Offline and Online Learning. (2024)
Directed by Dr. Haimeng Zhang. 101 pp.

Data contamination is a prevalent issue in real-life data sets, with approximately 10% of observations being affected, as noted by Hampel et al. in 1986 [32]. The presence of data contamination undermines the assumptions underlying existing machine learning algorithms. In this dissertation, we address this challenge by employing a penalized weighted method to enhance Stochastic Gradient Descent (SGD) and Random Forest (RF) models for regression analysis, particularly when mean-shift data contamination is present in the data set. The penalized weighted method assigns individual weights to observations in the training data set, and a Lasso-like penalty is applied to the individual weight. These individual weights, ranging from 0 to 1, govern the contribution of each training observation to the estimation of model parameters or the prediction of response variables. We present a novel approach, Penalized Weighted Stochastic Gradient Descent (PWSGD), designed for simultaneous outlier detection and accurate parameter estimation in regression problems. Furthermore, we introduce the Penalized Weighted Random Forest (PWRF) method, which adapts the RF model to enhance its robustness against systematic or trend contamination present in the training set. Both methods assess the impact of contamination in the training set based on the squared residual of each training observation, providing flexibility in handling unknown data contamination. Through numerical experiments and real data analysis, our observations indicate that the proposed methods exhibit competent performance, either yielding comparable results or outperforming benchmarking methods.

PENALIZED WEIGHTED METHODS FOR ROBUST OFFLINE AND ONLINE LEARNING

by

Mingyan Li

A Dissertation
Submitted to the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Greensboro
2024

Approved by

Haimeng Zhang
Committee Chair

To my dear parents for their unwavering love and faith in me. To myself for being persistent. To the four-year journey for introducing me to valuable people and challenges.

APPROVAL PAGE

This dissertation written by Mingyan Li has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair _____
Haimeng Zhang

Committee Members _____
David Banks

_____
Xiaoli Gao

_____
Sat Gupta

_____
Jianping Sun

_____
Date of Acceptance by Committee

_____
Date of Final Oral Examination

ACKNOWLEDGMENTS

The fruition of this dissertation represents far more than my own dedication, it is a testament to the unwavering support and encouragement I have received along this journey.

I have always felt fortunate to have had such a well-rounded and supportive committee for my dissertation. Dr. Xiaoli Gao provided early encouragement and guidance that set me on the right path from the beginning of my research journey. Dr. Haimeng Zhang's meticulous attention to detail and organization kept me on track. Dr. David Banks' warm welcome and inclusion in his group project expanded my horizons beyond my research area. The foundational knowledge imparted by Dr. Sat Gupta through his challenging courses has been invaluable to my doctoral learning. Dr. Jianping Sun's infectious smile and encouraging words have always brightened my day and motivated me to push forward. To each of you, I extend my heartfelt thanks. Your support, insightful comments, and scholarly input have greatly enriched this study. This dissertation would not have been possible without your contributions.

My deepest appreciation goes to my family for their unwavering belief in my abilities and their endless encouragement. To my cherished friends, your unconditional support, care, and companionship have been a constant source of strength throughout this academic journey. I am indebted to my colleagues and collaborators, whose encouragement and camaraderie have enhanced my academic experience and enriched my research endeavors.

Finally, I extend my gratitude to all those who have contributed in various capacities. Your belief in me, shared laughter during moments of stress, and shared experiences have made this path more meaningful and memorable.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Regression/Linear Regression Background

Regression analysis serves as a fundamental tool for estimating quantitative responses. The term 'regression' was first coined by Galton in his 1886 paper [25], wherein he investigated the correlation between the average of the height of father and the height of the offspring. The widely utilized method in regression analysis, known as least squared regression, predates the concept of regression itself and was independently explored by scholars around the same time. Adrien-Marie Legendre [41] and Carl Friedrich Gauss [29] are notably credited for their contributions to the least squared method. The late 18th century, Karl Pearson [51] introduced correlation coefficient to measure the direction and strength of the linear association between two quantitative variables. George Udny Yule [75] furthered the understanding of regression and correlation, where Yule also introduced the concept of partial correlation. By far, regression analysis was confined to simple regression involving two quantitative variables, one response variable and one explanatory variable.

The advent of analysis of variance (ANOVA) by Sir Ronald A. Fisher in 1935 [21] marked the beginning of multiple regression analysis commenced. This laid the

groundwork for the contemporary format expressed by the equation:

$$\mathbf{y}|X = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1.1}$$

where $\mathbf{y}$ represents the response variable vector, X is the matrix of explanatory variables (also known as independent variables or covariates), $\boldsymbol{\beta}$ denotes the model parameter vector, and $\boldsymbol{\varepsilon}$ signifies the random error vector. The post-World Wide Web era and widespread computer usage propelled the significance and popularity of regression analysis. Its applications burgeoned across diverse fields such as economics, social sciences, engineering, and data science. Despite the introduction of various algorithms with advancements in computing power and statistical methodologies, regression analysis remains a pivotal tool in numerous applications and forms the backbone of many advanced machine learning tools.

1.2 Big Data and Challenges

With the booming of technologies to collect data, store the data, and analyze the data, the time for big data had came. We have been immersed in this paradigm for over three decades, with the term 'big data' first gaining prominence in the 1990s, thanks to John Mashey's contributions. Since then, the concept has become integral to our technological landscape. The fundamental characteristics of big data, as succinctly outlined by Laney in a 2001 Meta Group research publication, encompass volume, velocity, and variety. Later on, more characteristics are used to describe the fast changing data format and credibility. As the landscape has evolved, additional characteristics have been identified to encapsulate the dynamic nature of data, including

its rapid changes in format and the imperative of ensuring data credibility.

Large data sets have become pervasive in various fields, including clinical, epidemiological, financial, and psychological or sociological studies. The widespread availability of the internet has further expanded the scope of data, incorporating sources such as search prompts to optimize search engine efficiency. One illustrative example encompassing all three fundamental characteristics of big data is derived from social media data sets. Social media data sets are characterized by their vast volume, containing a wealth of information in the form of texts, images, voices, and more. The sheer magnitude of data generated every second, often in the tens of thousands, presents a rich opportunity for studying and optimizing user behavior or social interactions to enhance retention rates. E-commerce, facilitated by the internet, has given rise to transactional data that provides valuable insights into consumer behavior. In clinical research, data sets may involve thousands of genetic markers studied for their associations with specific diseases. Alternatively, data sets in the medical field might include vast numbers of magnetic resonance images with information from hundreds of subjects, providing a comprehensive view for research and diagnosis. The ubiquity of data in diverse domains underscores that big data has become the new norm. Harnessing the potential of these extensive data sets is crucial for making informed decisions, gaining insights, and driving advancements in various fields.

As data continues to grow in both size and complexity, the traditional regression analysis method may prove inefficient not only due to inherent algorithmic limitations but also because of other constraints, the memory to store data as an example. To address these challenges, various algorithms have been proposed to tackle these issues. In cases where memory poses an obstacle or when the data set is not static

but generated dynamically over time, iterative methods have been devised to store only a subset of the data. Examples include stochastic gradient descent, adaptive gradient, and adaptive moment estimation. To accommodate data in diverse formats, the development of algorithms such as random forest, neural networks, and natural language processing has been pursued. However, as data generation accelerates, issues may arise within the data itself. The speed at which data is produced can potentially introduce challenges that need to be carefully addressed in the analytical process.

1.2.1 Data Contamination

Data contamination refers to the phenomenon within a data set where specific observations deteriorates the analysis performance. Data contamination may arise from various sources, including outliers, missing values, or concept drift. The existence of data contamination pose a substantial impact in regression analysis, since regression analysis assumes homogeneity. For example, a contaminated observation may come from the same type of distribution as the majority of the data set but have a higher or lower mean, exhibit a heavier tail, or even follow an entirely different distribution. Neglecting data contamination can introduce bias in the estimation of the mean of the parameters, resulting in less accurate predictions of the mean responses. This is particularly critical because regression analysis for means is highly sensitive to data contamination, making it imperative to address and account for such issues for more robust and reliable predictions.

In this dissertation, we focus on a specific type of data contamination characterized by a mean shift in the response variable, with the term 'outlier' used interchangeably. Outliers can arise from human-induced errors or observations originating from a

different population. Additionally, outliers may result from changes in the variability of the data, warranting careful consideration as they can provide valuable information and reveal potential lurking variables. In the context of regression analysis, an outlier is also referred to as a regression outlier [55], indicating a departure from the same linear pattern followed by the majority of the data set. This scenario occurs when contaminated observations and uncontaminated observations share similar covariates, while the response variable in the contaminated observations deviates from that in the uncontaminated observations. Proper handling of such outliers is crucial, distinguishing between those indicating potential insights and those resulting purely from errors should be down-weighted. A linear regression model with outliers can be expressed as follows:

$$\mathbf{y}|X = X\boldsymbol{\beta} + \boldsymbol{\varepsilon} + \mathbf{o}, \tag{1.2}$$

where $\mathbf{o}$ being a contamination vector that is sparse with most of the values in this vector being zero, while the value of the contaminated observations being a specific number, the rest remains the same as Equation (1.1). Various methods have been developed to identify outliers, such as cook's distance [14], studentized residuals [70], or Difference in Fits (DFFITS) [10]. While these methods are effective in identifying one outlier, they may falter when multiple outliers are present in the data set. Conventionally, one might delete the potential outliers and proceed with the analysis with the remaining data set. However, this approach has its limitations, as outliers may contain valuable information that warrants special treatment. Consequently, a more nuanced strategy is required to appropriately handle outliers throughout the analysis.

1.3 Stochastic Gradient Descent

Stochastic gradient descent (SGD), introduced in the 1950s by Robbins and Monro [53], is a widely employed method for iterative procedures in estimation problems. Its applicability extends to various models, including linear regression model, logistic regression, support vector machine, neural networks, etc. SGD stands out as one of the most utilized algorithms in modern data-driven problem-solving [5]. Furthermore, SGD serves as a proximal operator, engaging in iterative optimization with one observation at a time. Proximal operators project points to the minimum of convex functions, which can be viewed as a generalized projection. These operators find applications in statistical learning and model estimation algorithms, where optimization forms the core of the models.

These problems involve finding the optimal estimation that minimizes the expectation of a loss function. Let $\boldsymbol{\theta}^* \in \mathbb{R}^p$ represent the underlying model parameter, where p is the dimensionality of the covariate matrix, $X \in \mathbb{R}^{N\times p}$ denotes the observed data set, where N is the size of the data set. The statistical estimation can be formulated as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}^p}{\operatorname{argmin}}\, E(\rho(\boldsymbol{\theta}, X)), \tag{1.3}$$

where $E(\cdot)$ is the expectation function, and $\rho(\cdot)$ is a loss function defined for the specific problem. Given its iterative nature, proximal operators like SGD are particularly well-suited for streaming data scenarios, as they do not require the entire data set. Additionally, SGD is a valuable tool for efficiently handling large data sets that may pose computational challenges. These characteristics of SGD makes it numerical convenient and memory efficient. These characteristics make SGD numerically convenient

and memory-efficient. Furthermore, SGD operates with minimal assumptions, offering a cost-effective solution for solving non-smooth objectives.

SGD calculates the gradient of the objective function using one data observation at a time, the same process is repeated each time a new observation is generated to update the estimate of the model parameter. In Equation (1.3), the expectation is taken with respect to the entire covariate matrix. In an SGD procedure, the expectation is taken with respect to a stream of i.i.d. covariate vectors, $\mathbf{x}_n$, where $n = 1, 2, ..., N$, and N is the size of all oberved data set. For each new observation with covariate vector and response pair $(\mathbf{x}_n, y_n)$ from a streaming data set, the SGD update for the model parameters, $\hat{\boldsymbol{\theta}}$, is produced using the following method:

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} - \gamma_n \nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{x}_n, y_n), \tag{1.4}$$

where $\hat{\boldsymbol{\theta}}_n$ is the updated parameters after processing the n^{th} iterate of observations, and γ_n, the learning rate, is a non-increasing sequence of positive real numbers. The learning rate is computed as $\gamma_n = \gamma n^{-\alpha}$, where $\gamma > 0$ and $\alpha \in (1/2, 1]$, this setting of learning rate can be traced back to the introduction of this stochastic approximation by Robins and Monro [53].

SGD has been extensively studied in literature and has evolved from the original work since its inception. Numerous researchers, including but not limited to Sakrison [60], Ruppert [58], and Polyak and Juditsky [52], have explored the methodology, establishing consistency and asymptotic normality. Sakrison [60] applied SGD method to estimate the parameters of the covariance function, or the model parameters, of a gaussian process for a regression task. The covariance function is particularly relevant in tasks involving the modeling of correlations, such as in signal processing, geostatistics,

and environmental science. Nagumo and Noda [48] discussed the application of SGD as a learning method in system identification in the automatic control field, which laid the foundation for the modern control theory and machine learning.

Nagumo and Noda applied SGD for estimation in the normalized least-mean squares filter, and the update is very similar to an implicit SGD, which can be expressed as:

$$\boldsymbol{\theta}_n = \underset{\boldsymbol{\theta}\in\mathbb{R}^{\mathbf{p}}}{\operatorname{argmin}} \left\{ \frac{1}{2\gamma_n} ||\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}||^2 + \rho(\boldsymbol{\theta}, X) \right\}. \tag{1.5}$$

This same update was discussed in incremental proximal method in optimization by Bertsekas [4]. His aim was to enhance the efficiency of optimization algorithms, allowing them to handle large-scale (both massive data sets and high-dimensional data sets) convex optimization problems common in machine learning, signal processing, and other fields. Other researchers, including Toulis [68], Défossez and Bach [16], Toulis and Airoldi [67] have demonstrated that implicit SGD is more stable than SGD while maintaining comparable convergence rates. Related work on implicit SGD includes contributions from Kivinen et al. [38], Kulis and Bartlett [40] in the online learning literature, Cheng et al. [12] on implict SGD with kernels, and Duchi and Siner [15], Parikh and Boyd [50], and Rosasco et al. [54] introducing a stochastic proximal gradient algorithm that involves forward step with SGD and backward step with implicit SGD. Although the stochastic proximal gradient algorithm may bring about instability while reaching for accelerating convergence.

Averaging of the SGD updates has been discussed by several scholars, including Ruppert [58], Bather [2], Polilyak and Juditsky [52], among others. Instead of regard $\hat{\boldsymbol{\theta}}_n$ in Equation (1.4) as the estimate of the model parameters after the n^{th} observation,

an average is taken on the SGD estimates over all previous SGD estimates:

$$\bar{\boldsymbol{\theta}}_n = \frac{1}{n}\sum_{i=1}^{n}\hat{\boldsymbol{\theta}}_n. \tag{1.6}$$

This method does not require all estimates being stored, but only the latest averaged SGD estimate, $\bar{\boldsymbol{\theta}}_{n-1}$, and the total number of previous SGD estimates, n. With these values, the newest averaged SGD estimate can be updated as follows:

$$\bar{\boldsymbol{\theta}}_n = \frac{(n-1)\bar{\boldsymbol{\theta}}_{n-1} + \hat{\boldsymbol{\theta}}_n}{n} \tag{1.7}$$

The discussions around averaged SGD argue that averaging can invoke statistical optimality, leading to better estimation. The asymptotic optimality of averaging the SGD estimations is also demonstrated in these discussions indicating that slow-convergent stochastic approximation with a large learning rate benefits from averaging. Zhang [76], Shamir and Zhang [62], and Bach and Moulines [1] have shown the superiority of averaged SGD. Toulis et al. [69] combined the implicit SGD with averaging methodolody, further enhancing the efficiency and stability of SGD.

Bottou and Bousquet in 2008 [6] discussed the scalability of batch learning in face of big data. They advocated the use of SGD with mini-batches to improve efficiency compared to using the entire data set when the data set is massive. Mini-batch SGD leverages the computational efficiency of SGD, converges rapidly, and requires less memory. It enhances the stability of traditional SGD methods and reduces sensitivity to the choice of learning rate. Additionally, the use of mini-batches decreases the likelihood of SGD getting trapped in local minima since each iteration involves multiple observations rather than just one. The use of mini-batches in SGD

remains a conventional method for statistical learning with SGD after its introduction.

Statistical inference for model parameter applying SGD for estimation has also been studied by Fang et al. [19]. They proposed an online bootstrap method for constructing confidence intervals for the model parameters. The method generates a set of randomly perturbed SGD estimations, $\hat{\boldsymbol{\theta}}_n^\star$, for each new observation added to the data stream with a set of i.i.d. non-negative random variable $\mathcal{W} = \{W_i, i = 1, ..., N\}$,

$$\hat{\boldsymbol{\theta}}_n^\star = \hat{\boldsymbol{\theta}}_{n-1}^\star - \gamma_n W_n \nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{X}_n, \mathbf{y}_n), \tag{1.8}$$

$$\bar{\boldsymbol{\theta}}_n^\star = \frac{1}{n} \sum_{i=1}^{n} \hat{\boldsymbol{\theta}}_n^\star. \tag{1.9}$$

By demonstrating that $\sqrt{n}(\bar{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0)$ and $\sqrt{n}(\bar{\boldsymbol{\theta}}_n^\star - \bar{\boldsymbol{\theta}}_n)$ converge in distribution, Fang et al.were able to construct confidence regions for the model parameters with a large amount of bootstrap samples from the perturbed SGD estimates.

1.3.1 Robust SGD Using Re-weighting Method

SGD can be significantly influenced by outliers in the data set, given the relatively small amount of data used in each iteration. This effect is particularly pronounced in traditional SGD, where only one observation is utilized to update the estimate in each iteration, causing the SGD estimate to deteriorate if the observation is contaminated. The presence of outliers can also lead to overfitting issues, as SGD highly relies on the observed data [33], overfitting is another issue that practitioners and scholars dedicate to avoid. Although addressing overfitting is important, it is not the primary focus of this dissertation. The impact of outliers on SGD estimation becomes even more pronounced when the mean is the focal point of the analysis, as the mean is highly

sensitive to outliers. While applying mini-batch and averaging the previous estimates may provide some help in this case, the estimate of expectations is still likely to be skewed. However, the impact might be less compared to traditional SGD. To mitigate the influence of outliers, scholars have dedicated efforts to developing robust SGD methodologies.

Needell et al. [49] employed importance sampling for SGD to alleviate the high dependency of the estimation variance on the random error, $\boldsymbol{\varepsilon}$. The re-weighting method diverges from the traditional SGD which uniformly selects observations for estimation updates. While uniform sampling provides an unbiased estimator, it often leads to high variance. Needell et al. [49] proposed implementing importance sampling via rejection sampling, where each observation is assigned a weight (w) such that the expectation of all weights equals one ($\mathbb{E}[w(i)] = 1$), where i is the observation index. They determined the weights using a partially biased sampling method corresponding to the Huber contamination model [35] defined as:

$$w^{\lambda}(i) = \lambda + (1 - \lambda)\frac{L_i}{\bar{L}}, \qquad \lambda \in [0, 1],$$

where L_i represents the Lipschitz constant of the i_{th} iteration, $\bar{L}$ denotes the averaged Lipschitz constant, and λ is a parameter that controls the proportion. The acceptance probability is $w(i)/W$, for some $W \geq \sup_i w(i)$, and the accepted samples are used to update SGD estimates with the reciprocal of the weight multiplying the gradient in Equation (1.4). This importance sampling technique involves re-sampling observations using a re-weighting distribution, increasing the likelihood of outlier rejection. However, this method applies to each iteration of the update and faces challenges in online settings where mini-batches lack sufficient data, as the entire data set is unavailable.

Zhao and Zhang [77] worked at the same time on proposing importance sampling specifically on Proximal Stochastic Gradient Descent (or general proximal stochastic mirror descent) and Proximal Stochastic Dual Coordinate Ascent, their goal was also to reduce the variance of stochastic gradient descent. Xiao and Zhang [73] proposed a multi-stage proximal stochastic gradient method to reduce the estimate variance when solving the regularized empirical risk minimization problems. They both have the same issue as Needell et al. [49].

Shah et al. [61] proposed MKL-SGD method, is a weighted sampling method based on residuals. This method assumes that there is a small portion of data may be contaminated, and these observations are typically associated with elevated loss values. The approach involves selecting a set of k samples and identifying the sample with the smallest loss within this set. Subsequently, the gradient of this chosen sample is utilized in Equation (1.4) for updating the model parameter estimation. The fundamental concept of the method revolves around choosing one sample in each batch that provides the most information and leveraging this information to update the model parameters estimation. However, there remains a possibility that the chosen sample may still contain outliers, potentially compromising the performance of the SGD estimation. MKL-SGD requires that a sample undergo the loss calculation step to be selected among the k samples first. The probability of the i_{th} sample being selected, denoted as p_i, is determined by the following method:

$$p_i = \begin{cases} \frac{\binom{n-i}{k-1}}{\binom{n}{k}} & \text{without replacement,} \\ \frac{(n-(i-1))^k-(n-i)^k}{n^k} & \text{with replacement,} \end{cases}$$

where n is the total number of observations in a batch. The selection process excludes

the observations that are not in the chosen sample from the batch, resulting in a loss of more information than gain. This process may resemble outlier detection, where potential outliers are identified and subsequently removed from the analysis.

1.4 Random Forest

Random forest (RF) [7] is an ensemble method that is based on Breiman's earlier work with some other distinguished scholars on the decision trees (classification and regression trees) [8]. It is an extension of his prior work on the bagging method [9]. In a decision tree, observations are categorized into groups using a combination of grouping criteria, with each criterion leading to the partitioning of the covariate space into rectangles or hyper-rectangles. Each criterion serves as a splitting node in the tree, and observations with covariates in the same rectangle or hyper-rectangle fall into the same leaf of the tree. For a classification tree, the estimation of a new observation using a decision tree involves assigning it the dominant category within the leaf to which it belongs. In a regression tree, it entails calculating the average value of the response variable for all observations in the same node. While an individual tree predictions are low bias but high variance, it has been proved that an ensemble or boosting method can effectively reduce the variance but maintain low bias [9], [24].

A RF comprises many decision trees, each of which trains on a bootstrapped sample of the training data set and a subset of the covariates. The estimation or a prediction of the RF for a classification task is the majority category predicted by all trees, while for regression task it is the averaged prediction of all trees. Lin and Jeon [44] argued that a RF is akin to an adaptive neighbor method. In this dissertation emphasizes the regression RF. The utilization of bootstrapped samples of observations

and subsets of covariates renders RF a robust method, addressing randomness from both a data and an information perspective. Averaging across trees reduces estimation and prediction variance through the law of large numbers, enhancing algorithm stability and mitigating the risk of overfitting without introducing additional bias. RF's robustness to noisy data sets, particularly those with heavily-tailed noise, stems from its invariance to such noise. Moreover, as each tree employs a randomly selected subset of covariates, RF takes an alternative approach in high-dimensional data sets, avoiding challenges associated with the curse of dimensionality. Consequently, RF proves powerful when addressing high-dimensional problems.

RF has long been perceived as a somewhat mysterious method due to its lack of interpretability. This limitation arises from its inability to be explicitly formulated like a linear model or traced down like one would do for a decision tree. However, it has been demonstrated that RF models can indeed be expressed in a familiar mathematical format. In 2006, Meinshausen and Ridgeway [46] showed that an RF model can be represented by a mathematical equation, where the prediction is a weighted sum of the training responses. Let θ_t be the random subset of covariates for tree t, with a total of m trees in a RF model. Let (X_i, y_i) be observation i in the training set, where there are n training observations in total, the RF prediction for a new observation with $X = \mathbf{x}$ can be expressed as follows:

$$\widehat{Y}_{RF}(\mathbf{x}) = \sum_{i=1}^{n} w(X_i, \mathbf{x}) y_i, \tag{1.10}$$

$$w(X_i, \mathbf{x}) = \frac{1}{m} \sum_{t=1}^{m} w(X_i, \mathbf{x}, \boldsymbol{\theta}_t), \tag{1.11}$$

where $w(X_i, \mathbf{x}, \boldsymbol{\theta}_t)$ is the weight of training observation i from tree t, and it is easy to

show that $\sum_{i=1}^{n} w(X_i, \mathbf{x}) = 1$. The weight $w(X_i, \mathbf{x})$ varies for each new observation $X = \mathbf{x}$, and it is adaptively assigned to observations in the same leaf based on the similarity among covariates. This weight tends to be larger when the conditional distribution of the response given an observation is similar to the conditional distribution of the response given the new observation [44]. This is analogous to the process of assigning nearest neighbors. About ten years later, Li and Martin [42] demonstrated that RF prediction can be expressed in generalized loss function form:

$$\widehat{Y}_{RF}(\mathbf{x}) = \underset{\mathcal{S}\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} w(X_i, \mathbf{x})\rho(y_i, \mathcal{S}(\mathbf{x}_i)), \tag{1.12}$$

where $\mathcal{F}$ is a family of functions and $\rho(\cdot)$ is a general loss function. When using the least squared method, the loss function becomes the sum of the squared response residuals, i.e. $\rho(\cdot) = ||y - \widehat{y}_{RF}||^2$, where $\widehat{y}_{RF}$ is the RF estimation.

RF is widely acclaimed for its adaptability to both categorical covariate and quantitative variables, as well as its ability to perform effectively in both classification and regression tasks. Its popularity is further heightened by its ease of application, requiring only two user inputs: the number of covariates to be randomly selected for each tree and the number of trees to be grown in the model. Scholars and practitioners across various academic disciplines and industries have successfully applied the RF method, demonstrating its robust performance. Liaw and Wiener published their work using R language applying RF method quickly after the publication of Brieman's work [43]. An attractive feature of RF is its capability to provide practitioners with insights into the importance of covariates after model fitting. This attribute, along with its resilience to heavy-tailed distributions and capacity to handle both categorical and quantitative covariates, contributes to RF's widespread adoption in both academic

and industrial settings.

RF's robustness to data sets with response values that exhibit heavy tails, indicating the presence of extreme cases, suggests its resilience to outliers. However, this assumption holds true only when both the training and testing observations originate from the same distribution. RF loses its robustness if the training and testing responses stem from different distributions. This occurs when response values of some observations in the training set are contaminated, causing the contaminated response variable to follow a distinct distribution with a different mean compared to the uncontaminated distribution. Furthermore, the RF weight $w(X_i, \mathbf{x})$ is larger when two observations have stronger similarity. However, if the training obesrvation is contaminated, then the weight can become extremely misleading. Consequently, the prediction from a RF model for an uncontaminated new observation with covariates similar to the contaminated observation may be impacted, leading to a less accurate prediction.

1.4.1 Robust random forest

Several scholars have delved into addressing the sensitivity of RF to outliers, aiming to enhance its robustness in the face of diverse data contamination. An effective strategy to bolster robustness, particularly in the presence of outliers, involves leveraging order statistics, such as using the median instead of the mean to characterize the central tendency of a distribution. Meinshausen and Ridgeway [46] proposed a novel approach to mitigate the impact of outliers by estimating quantile, denoted as $F(y|X = x) = P(Y \leq y|X = x) = E(\mathbb{1}_{\{Y \leq y\}}|X = x)$, instead of the expectation of the response variable, $E(Y|X = x)$. This method, known as Quantile Regression Forest

(QRF), focuses on the weighted distribution of responses rather than the weighted mean. The estimated cumulative conditional distribution, denoted as $\widehat{F}(y|X=\mathbf{x})$, is expressed as the following:

$$\widehat{F}(y|X=\mathbf{x}) = \sum_{i=1}^{n} w(X_i, \mathbf{x}) \mathbb{1}_{\{Y_i \leq y\}}, \tag{1.13}$$

the weight $w(X_i, \mathbf{x})$ shares the same definition as in RF. By estimating the conditional distribution of the response variable, QRF not only provides a robust prediction interval but also detects potential outliers using either conditional median absolute deviation or the conditional interquartile range. Roy and Larocque [56] demonstrated that applying robust aggregation methods to RF, especially through QRF, enhances robustness against outliers. While QRF proves to be a versatile tool, it does not offer a solution when the objective is to estimate the mean response in the presence of outliers.

Li and Martin [42] advocated for enhancing the robustness of RF to outliers by employing robust loss functions. They established a connection between RF and k-potential nearest neighbors proposed by Lin and Jeon [44], expressing the loss function of RF in the form of Equation (1.12). They reformulated the QRF in the format of Equation (1.12) using the τ-th quantile loss function defined by Koenker [39]:

$$\rho_\tau(y - \widehat{y}) = (y - \widehat{y})(\tau - \mathbb{1}_{\{y - \widehat{y} \leq 0\}}).$$

They then proposed that using robust loss function can improve robustness of RF. They introduced a smooth approximation of the Huber loss function [35], known as

the pseudo-Huber loss function by Charbonnier et al. [11],

$$\rho_\delta(y-\widehat{y}) = \delta^2\left(\sqrt{1+\left(\frac{y-\widehat{y}}{\delta}\right)^2}-1\right),$$

where δ is the tuning parameter to control the penalization strength, and Tukey's biweight function, a non-convex redescending loss by Huber [36]:

$$\rho_\delta(y-\widehat{y}) = \frac{d}{d(y-\widehat{y})} = \begin{cases} (y-\widehat{y})\left(1-\frac{(y-\widehat{y})^2}{\delta^2}\right)^2 & \text{for } |y-\widehat{y}| \le \delta, \\ 0 & \text{elsewhere.} \end{cases}$$

The pseudo-Huber loss function adapts the RF weight and shrinks closer to zero for observations with disparate responses. Tukey's biweight function diminishes to zero for large dissimilarities between response values. Through simulations and real-world data, Li and Martin demonstrated the effectiveness of these proposed robust RF methods, particularly in providing robust point estimates for mean responses even in the presence of outliers. However, it is noted that their computational approach requires repetition for new observations, as the loss function relies on the proximity between the predicted responses of new observations and the training set.

In the following year, Sage [59] proposed the method, RF-LOWESS, assigning an individual weight for each training case based on the training residuals using Tukey's bisquare function to enhance the robustness of RF. Tukey's bisquare function is applied to the ratio of the residual of a training observation and the median of all absolute residuals:

$$B\left(\frac{e}{\alpha m}\right) = \begin{cases} \left(1-\left(\frac{e}{\alpha m}\right)\right)^2 & if\left|\frac{e}{\alpha m}\right| < 1, \\ 0 & if\left|\frac{e}{\alpha m}\right| \ge 1, \end{cases}$$

where e is the residual of a training observation, m is the median of all absolute residuals, and α is a tuning parameter. Sage argued that α should be smaller with substantial contamination in training data set, as a smaller value aggressively weakens the contribution of potential outliers aggressively. Therefore, Sage's method can stabilize the impact of outliers while simultaneously detecting outliers from the training observations. Additionally, the weights depend solely on the residuals of the training observations, allowing for one-time calculation when building the RF model and applying it to all new observations. This save the computation cost and differs from the method by Li and Martin, where the weight is calculated based on the discrepancy between the new observation's predicted response and the training observations' responses, which varies for each new observation. Although Sage demonstrated improvement of RF-LOWESS against QRF by Meinshausen and Ridgeway, RF with Huber loss function by Li and Martin, and the original RF, the performance of RF-LOWESS did not show superiority on real data sets.

1.5 Penalized Weighted Method

The application of penalized weighted methods has been instrumental in enhancing the robustness of various algorithms against outliers, described as Equation (1.2). Within the framework of a penalized weighted approach, individual weights are allocated to each training observation, thereby regulating their influence on the estimation or prediction process. These weights are subject to penalization, which dictates their assignment. It is noteworthy that the concept of penalization in this context diverges from its conventional application in high-dimensional data analysis. In the latter, penalization typically involves constraining parameters to achieve variable selection

by shrinking certain parameters towards zero or near-zero values. Within the context of penalized weighted methods, the penalty is applied to the weights assigned to each training observation. This mechanism serves the purpose of regulating the influence of outliers by adjusting the weights allocated to training observations.

Gao and Fang [26] proposed penalized weighted least square (PWLS)as a method to enhance least square regression's robustness against data contamination and identify outliers simultaneously. They applied a lasso-type penalty on the log-transformed weight, and assumed a linear model as the following:

$$y_i = \mathbf{x}_i'\boldsymbol{\theta}^* + \varepsilon_i/w_i^*,$$

where $\mathbf{x}_i' \in \mathbb{R}^p$ is the covariate vector of observation $i = 1, 2, ..., N$, N denotes the size of the training data set, y_i is the corresponding response of training observation i; $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, ..., \theta_p^*)'$ is the coefficient vector; ε_i's are independent random error with zero mean and constant variance if homogeneity is assumed; w_i^* is the underlying weight of this training observation. $w_i^* < 1$ indicates an outlier and $w_i^* = 1$ indicates a non-outlier. The objective function for this linear model under the penalized weighted method is

$$(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{w}}) = \underset{\boldsymbol{\theta}, \mathbf{w}}{\arg\min} \left\{ \sum_{i=1}^{n} w_i^2 (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 + \sum_{i=1}^{n} \lambda |\log(w_i)| \right\}, \tag{1.14}$$

where λ is a tuning parameter that controls outlier detection, they also proved that PWLS is an M-estimator. Gao and Fang extended PWLS to the adaptive Lasso method, incorporating it into PWLS as aPWLS. The objective function of aPWLS is

given by:

$$(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{w}}) = \arg\min_{\boldsymbol{\theta},\mathbf{w}} \left\{ \sum_{i=1}^{n} w_i^2 (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 + \sum_{i=1}^{n} \lambda \varpi_i |\log(w_i)| \right\}, \tag{1.15}$$

where ϖ_i is a pre-defined penalty scale factor for an observation. PWLS is a special case of aPWLS when all ϖ_is being 1. Furthermore, they extended extended PWLS to heterogeneous model (H-PWLS) where the random error $\varepsilon_i = g(\mathbf{x}_i'\vartheta)\epsilon_i$, where $g(\cdot)$ is a known function, ϑ is the estimation parameter, and $E(\epsilon_i) = 0$ and $E(|\epsilon_i|) = 1$. The objective function of H-PWLS is

$$(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{w}}) = \arg\min_{\boldsymbol{\theta},\mathbf{w}} \left\{ \sum_{i=1}^{n} w_i^2 (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 / g(\mathbf{x}_i'\widehat{\boldsymbol{\vartheta}}) + \sum_{i=1}^{n} \lambda \varpi_i |\log(w_i)| \right\}. \tag{1.16}$$

They conducted simulation studies in both homogeneous and heterogeneous settings, and applied the method to three real-life data sets for comparison with the hard-IPOD method proposed by She and Owen [64]. PWLS outperforms HIPOD in all homogeneous setting in terms of the joint outlier detection rate (the fraction of zero masking phenomenon) and overall masking rate. Moreover, H-PWLS significantly outperformed both PWLS and HIPOD in all heterogeneous settings, even when the variance function was misspecified. Additionally, PWLS accurately identified outliers in the real data sets while HIPOD exhibited tendencies to mask outliers or swamp non-outliers.

Around the same timeframe, Gao [27] proposed penalized weighted low-rank approximation method (WPLA) method aimed at robustly recovering recurrent copy-number variations (CNVs), a critical research area in genomics study essential for understanding complex disease. Data contamination in this domain may arise from

non-Gaussian random noise, heteroscedasticity among some probes, or pure errors. WPLA applied a penalized weighted approach akin to the previously mentioned method to an existing robust technique for recurrent CNV detection. In this method, each probe of every sample was assigned a weight, with a weight value of 1 indicating a normal probe, while other values signaled potential data contamination due to individual-specific effects (heteroscedasticity). Gao demonstrated the superiority of WPLA over two existing robust recurrent CNV recovery methods using synthetic data sets associated with both Gaussian and t-distributed random errors (contaminated cases). WPLA successfully recovered the recurrent CNVs and exhibited robustness even under misspecified model assumptions. Furthermore, WPLA outperformed the two existing robust recurrent CNV recovery methods on real data sets by efficiently detecting recurrent CNVs and data contamination arising from individual-specific effects. These findings highlight that WPLA is not constrained by assumptions of Gaussian random errors nor does it necessitate homogeneity in the random error distribution.

Gao and Feng [28] proposed penalized weighted least absolute deviation (PWLAD) method, aiming to enhance the robustness of traditional least absolute deviation (LAD) method in the presence of outliers in the response variable or leverage points (outliers in covariates). PWLAD, like the other penalized weighted methods, a weight to each observation. However, it enforces a lasso-type penalty on $1 - w$, where w is the weight. This weight is utilized to detect outliers or leverage points in a data set. They demonstrated that PWLAD outperforms LAD, weighted absolute deviation method (WLAD), and PWLS. It was shown to correctly detect outliers through numerical experiments with homogeneous random errors and maintained robustness even in the presence of heterogeneous and heavy-tailed random errors. The real data analysis

presented in the paper further demonstrated that PWLAD exhibited robustness in the presence of outliers compared to LAD and WLAD, achieving high accuracy in outlier detection.

Luo et al. [45] have recently employed the penalized weighted method to the Cox proportional hazards (PH) regression to improve robustness for censored survival outcomes. This method addresses the challenge of exceptional responders in precision medicine, where these responders are considered outliers. It raises a concern in precision medicine as the PH method is very sensitive to these exceptional responders, where these exceptional responders are outliers in this field. The proposed method, termed Penalized Adaptive Weighted Proportional Hazards (PAWPH), introduces a weight to the hazard function. It applies a lasso-type penalty on the log-transformed weight to accurately identify influential outliers. Additionally, it enforces a lasso-type penalty on the regression parameters for variable selection, resulting in more robust estimation. Unlike other penalized weighted methods, in PAWPH, the weight w, signifies a long survivor when $w < 1$, $w > 1$ indicates an short survival, and $w = 1$ refers to regular survival time. The authors demonstrated the robustness and accuracy of PAWPH by applying it to both high-dimensional and low-dimensional settings and comparing the results with those obtained using the Cox method and robust Cox method. The findings indicated that PAWPH successfully detected influential observations in both high and low-dimensional scenarios, achieving high accuracy in variable selection. Furthermore, when applied to two different real-life data sets, PAWPH demonstrated the ability to detect most outliers with a low false rate.

1.6 Main Contributions

1.6.1 Robust stochastic gradient descent for online linear regression learning

In Chapter 2 we explore improvement of stochastic gradient descent (SGD) with potential data contamination in the response variable. We propose penalized weighted stochastic gradient descent (PWSGD) method for online linear regression learning. This method entails assigning individual weights, denoted as $w > 0$, to each training observation while imposing a Lasso-type penalty on $1 - w$. The individual weight $w$serves to gauge the influence of each observation on the estimation of the linear regression model parameter. If an observation is suspected to be an outlier, a small w is assigned to that observation, otherwise, $w = 1$. PWSGD simultaneously estimates the individual weight alongside the model parameter based on the residual of each observation, thereby rendering the weight determination entirely data-dependent. The penalty shrinks the individual term to 1, thus mitigating the risk of over-identification of outliers. PWSGD is based on stochastic gradient descent (SGD) where we follow the mini-batch framework and utilize the averaged SGD as the SGD estimator. The proposed method is able to simultaneously perform outlier detection and linear regression model parameter estimation, accommodating both fixed large data set and streaming data in an online setting.

We compare PWSGD to SGD using synthetic data with different synthesized levels of data contamination. The results show that PWSGD improves SGD estimation performance by accurately identifying outliers. Additionally, when applied to a real-world data set, the results provide compelling evidence that PWSGD exhibits

significantly higher robustness compared to SGD.

1.6.2 Robust Random Forest

In Chapter 3 we delve into RF method in the presence of outlying responses among training observations. We propose penalized weighted random forest (PWRF) method, designed to detect outliers in the training data set and improve prediction accuracy compared to the original RF. In this method, an individual weight, $0 < w \leq 1$, is assigned to each training observations. A shrinkage rule is applied to the log-transformed individual weight vector to govern the amount of outliers. PWRF simultaneously estimates both the individual weight w and the response variable for all training observations. The individual weight estimation is based on the residuals of the training observations; a large individual weight is assigned to observations with small residuals, and vice versa. Consequently, an observation is flagged as a suspected outlier if the estimated individual weight is small, with the ideal scenario being $w = 1$ for non-outliers. Importantly, the weight depends solely on the training residuals and needs to be calculated only once for all new observations. This stands in contrast to the approach by Li and Martin [42], where weights depend on the dispersion between the response variable of training observations and the predicted response variable for each new observation, necessitating recalculation for each new observation. This efficiency contributes to computational cost savings in real-life applications.

PWRF demonstrates superior robustness compared to the original RF in robustness across extensive simulation study examples with synthetic data contamination at various proportions. Furthermore, it exhibits performance on par with two widely used robust RF methods. Additionally, in a real-life data application, PWRF, alongside

the other two robust RF methods and the original RF model, is deployed. Notably, PWRF outperforms all three methods in prediction accuracy and performance in this real-life scenario.

Chapter 2: Penalized Weighted SGD for Robust Online Linear Regression Learning

2.1 Overview

Many machine learning algorithms involve optimizing an objective function across the entire data setwhich can be computationally exhaustive and memory inefficient, particularly with large data sets, often leading to computational failures. Stochastic gradient descent (SGD) is well recognized by its efficiency in estimating model parameters in large-scale data due to its reduced memory requirements and computational efficiency. SGD was initially proposed by Robin and Monro in 1951 [53] as a stochastic approximation method, also referred to as Robbins–Monro method. Since then, it has been extensively studied in academia and integrated into various machine learning approaches. SGD is considered a special case of online learning algorithms [34], since it iteratively processes the data one observation at a time. Wang et al. [71] reviewed some achievements on application of SGD to big data and streaming data, particularly

its utilization in advanced machine learning algorithms such as neural networks. The versatility of SGD extends to various domains, including but not limited to regression analysis, classification problems, image recognition, time series analysis, and functional analysis.

Suppose we aim to estimate the model parameter $\boldsymbol{\theta}$ by minimizing the loss function $\rho(\boldsymbol{\theta}; X, \mathbf{y})$. Given an initial estimate $\hat{\boldsymbol{\theta}}_0$, SGD updates the estimate iterativelly with the following method:

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} - \gamma_n \nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{x}_n, y_n), \tag{2.1}$$

where $n = 1, 2, ..., N$, N is the total number of observations in the data set, γ_n is the learning rate, and $\nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{x}_n, y_n)$ is the gradient of the loss function. Conventionally, the learning rate is calculated with the following formula:

$$\gamma_n = \gamma n^{-\alpha}, \tag{2.2}$$

where $\gamma > 0$ and $\alpha \in (0.5, 1)$.

The asymptotic properties of SGD estimators have been well established in the literature. Kiefer and Wolfowitz [37] were among the first to rapidly apply the SGD method to regression function and proved its asymptotic properties. Sahrison [60] proposed a modified SGD method and demonstrated its stability by deriving the limit of mean squared error, while also elucidating the asymptotic properties of SGD in their study. Ruppert [58] extensively discussed the convergence rate of the SGD process in relation to the learning rate. Additionally, Polyak and Juditsky [52] formalized the average SGD estimate for linear regression, providing rigorous proof of the asymptotic

normality of the averaged SGD estimates, emphasizing almost sure convergence.

The aforementioned desirable properties of SGD estimates are established under the assumption that consecutive observations are identically and independently distributed with homogeneous noise. Here, homogeneous noise is assumed to follow an identical and independent normal distribution.

As data comes in sequentially, the possibility of data contamination increases. Similar to many other online learning algorithms, SGD does not account for the data "veracity" issue and assumes the perfect quality of all incoming data [34]. In the presence of contaminated data, SGD incorporates this information into the model parameter update as if it were uncontaminated. Consequently, this can negatively impact estimation accuracy and result in poor predictions for future instances. As discussed by Goodfellow et al. [30], algorithms with higher complexity are more susceptible to the impact of outliers. Hence, it becomes crucial to safeguard complex models against deterioration caused by outliers.

In this chapter, we propose a novel method called penalized weighted stochastic gradient descent (PWSGD) method, designed for simultaneously detecting data contamination and accurately estimating model parameters $\boldsymbol{\theta}$ using SGD method in the least square regression setting. Our proposed method controls the impact of the identified outliers by assigning a very small weight, while the non-outliers are assigned weights equal to one. These weights are regulated by a tuning parameter λ, and are updated together with the model parameters $\boldsymbol{\theta}$. The optimal tuning parameter $\hat{\lambda}$ is determined with a random weighting procedure. The inspiration for our approach is drawn from Fang et al. [19]. Their work proposed a random perturbation online bootstrap method for SGD estimator statistical inference. This method generates a large bootstrap sample of perturbations, $\mathcal{W} = \{W+i, i = 1, ..., N\}$, from a distribution

with an expectation and variance both equal to 1. The SGD update incorporates these perturbation, modifying Equation (2.1) as follows:

$$\hat{\boldsymbol{\theta}}_n^\star = \hat{\boldsymbol{\theta}}_{n-1}^\star - \gamma_n W_n \nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{X}_n, \mathbf{y}_n), \tag{2.3}$$

$$\bar{\boldsymbol{\theta}}_n^\star = \frac{1}{n} \sum_{i=1}^{n} \hat{\boldsymbol{\theta}}_n^\star. \tag{2.4}$$

Their proof demonstrates that the Kolmogorov-Smirnov distance between $\sqrt{n}(\bar{\boldsymbol{\theta}}_n - \boldsymbol{\theta}^*)$ and $\sqrt{n}(\bar{\boldsymbol{\theta}}_n^\star - \bar{\boldsymbol{\theta}}_n)$ converges to zero in probability. This allows the construction of confidence regions for the model parameters $\boldsymbol{\theta}$ using the bootstrap-perturbed SGD estimates. While similar resampling techniques were initially introduced by Rubin [57] and Shao abd Tu [63], they were applied in a traditional offline setting, not an online one. While the primary focus of this chapter is not statistical inference for model parameters using SGD estimates, we adapt the logic of perturbation, applying it to introduce small weights to each observation as a label for outliers. The perturbation concept is also utilized in the tuning parameter selection step of our proposed method to create two sets of perturbed observations.

The remaining of this chapter is organized as follows. We provide a detailed analytical exposition of PWSGD method analytically in Section 2.2. This includes elucidating our approach to select the optimal tuning parameter $\hat{\lambda}$ and assess the stability of our proposed method. In Section 2.3, we present the simulation results with various data settings using PWSGD and compare its estimation performance to that of the SGD method. In addition, we showcase the performance of PWSGD on real data in Section 2.4. In Section 2.5, we summarize the findings and outline potential avenues for future research.

2.2 PWSGD for linear regression

SGD is a powerful method that facilitates the development of complex models for finding optimal solutions in a scalable manner. However, its performance can be significantly affected by outliers or shift data contamination. In order to mitigate the influence of outliers, we introduce individual weights, $w \in [0, 1]$, to each training observations. Ideally, these weights would be 1 for non-outliers and 0 for outliers. However, in real-world examples, due to the presence of noise in the data sets, the estimated true weights typically fall within the range of 0 and 1, inclusively. These individual training weights serve as mediators, determining the contribution of each observation to the overall model during training. A smaller weight implies a potential outlier, thereby constraining the observation contribution to the model parameter estimation. Conversely, a larger weight indicates a lower likelihood that an observation is an outlier, thereby imposing less restriction on the contribution of the observation to the overall model estimation.

Assume a regression linear model:

$$\mathbf{y} = \boldsymbol{\theta}\mathbf{X}' + \boldsymbol{\varepsilon}, \tag{2.5}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the response variable vector, n is the size of the data set, $\mathbf{X} \in \mathbb{R}^{n \times p}$ is the covariate matrix, or the independent variable matrix, p is the dimension of the model or the number of independent variables, $\boldsymbol{\theta} \in \mathbb{R}^p$ is the model parameter vector, and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is the random error, or noise in the data set.

The traditonal SGD processes individual observations one at a time, where each observation has an equal probability of being selected to update the parameter vector

estimation using Equation (2.1). However, a more prevalent approach involves dividing the training set into mini-batches, with each mini-batch having an equal likelihood of being chosen for updating the parameter vector estimation using Equation (2.1). The goal of SGD for a linear regression model is to minimize the objective function:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{\mathbf{P}}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 \right\}. \tag{2.6}$$

PWSGD adopts mini-batch method to update estimations of the model parameters, and the final estimate is obtained by taking the average of all previous updated estimations. With an individual weight, $w \in [0, 1]$, assigned to each training observation, the goal of PWSGD is to estimate the model parameters along with the individual weight. A Lasso-like penalty is added to the objective function on the individual weight with a tuning parameter λ to discourage assigning too many observations with small individual weights. The objective function can be reformulated as follows:

$$(\widehat{\mathbf{w}}, \widehat{\boldsymbol{\theta}}) = \underset{\boldsymbol{\theta}\in\mathbb{R}^{\mathbf{P}},\mathbf{w}\in\mathbb{R}^{n}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} w_i^2 (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 + \lambda \sum_{i=1}^{n} |1 - w_i| \right\}. \tag{2.7}$$

The update for the model parameters using the n^{th} mini-batch of the training data changes accordingly to the change in the objective function:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_n &= \hat{\boldsymbol{\theta}}_{n-1} - \gamma_n \nabla \rho(\hat{\boldsymbol{\theta}}_{n-1}; \mathbf{X}_n, \mathbf{y}_n) \\ &= \hat{\boldsymbol{\theta}}_{n-1} - 2\gamma_n \mathbf{X}_n' \mathbf{W}_n' \mathbf{W}_n (\mathbf{y}_n - \mathbf{X}_n \hat{\boldsymbol{\theta}}_{n-1}), \end{aligned} \tag{2.8}$$

where $\hat{\boldsymbol{\theta}}_n$ is the update for the model parameters using the n^{th} mini-batch of the training data. $\hat{\boldsymbol{\theta}}_{n-1}$ is the update for the model parameters using the $(n-1)^{th}$ mini-batch of the training data. γ_n is the learning rate for the n^{th} update. ($\mathbf{X}_n$ is

the covariate matrix of the n^{th} mini-batch, $\mathbf{y}_n$) is the response variable vector of the n^{th} mini-batch. $\mathbf{W}_n$ is a diagonal matrix with $\mathbf{w}_n$ on the diagonal, and $\mathbf{w}_n$ is the individual weight vector for the n^{th} mini-batch. For individual weight estimation, the assignment is shown in Equation (2.9):

$$\widehat{w}_i = \begin{cases} 1 & \text{if } (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 \leq 0.5\lambda, \\ \frac{\lambda}{(y_i - \mathbf{x}_i'\boldsymbol{\theta})^2} & \text{if } (y_i - \mathbf{x}_i'\boldsymbol{\theta})^2 > 0.5\lambda, \end{cases} \tag{2.9}$$

where i, $i = 1, 2, ..., n_b$, is the index of the training observations in a mini-batch, n_b is the total amount of training observations in a mini-batch.

2.2.1 PWSGD Algorithm

Algorithm 1 provides an overview of the proposed method for processing one mini-batch of new iterations. We obtain model parameter initial value, $\boldsymbol{\theta}_0$, from applying robust linear regression to a given training set. Subsequently, the algorithm computes the current residuals using the initial model parameter value ($\boldsymbol{\theta}_0$) and a mini-batch of new observations, and proceeds to update the individual weights, $\mathbf{w}$. This update is performed using Equation (2.9), starting with the initial weights, $\mathbf{w}_0$, which are assumed to be uniformly 1. Following this, the algorithm employs the updated individual weights to update the model parameter, $\boldsymbol{\theta}$. The update for the individual weights and model parameter is iterated until convergence is achieved for both. The PWSGD estimate of the model parameter after processing this mini-batch of data is computed as the average of all previously converged model parameters, and the individual weights is the PWSGD estimate for each new observation.

Input: Initial model parameter for this iteration $\hat{\boldsymbol{\theta}}^{(0)}$, λ, γ, α, n
Data: $(\mathbf{X}_n, \mathbf{y}_n)$ - the n^{th} iteration of the entire data set
Result: $w_n = w_n^{(j)}$ and $\hat{\boldsymbol{\theta}}_{PWSGD} = \hat{\boldsymbol{\theta}}^{(j)}$
$\mathbf{w}_0 \leftarrow \mathbf{1}$ and $\gamma \leftarrow \gamma n^{-\alpha}$
Compute initial individual weight by
if $(y_i - \mathbf{x}_i'\hat{\boldsymbol{\theta}}^{(0)})^2 > 0.5\lambda$ **then**
  $w_i^{(0)} \leftarrow \frac{0.5\lambda}{(y_i - \mathbf{x}_i'\hat{\boldsymbol{\theta}}^{(0)})^2}$;
else
  $w_i^{(0)} \leftarrow 1$;
end
let $j = 1$;
while *not converge* **do**
  [Update $\hat{\boldsymbol{\theta}}$]
  $\hat{\boldsymbol{\theta}}^{(j)} = \hat{\boldsymbol{\theta}}^{(j-1)} - 2\gamma(w_i^{(j-1)})^2(y_i - \mathbf{x}_i'\hat{\boldsymbol{\theta}}^{(j-1)})\mathbf{x}_i$;
  [Update w] **if** $(y_i - \mathbf{x}_i'\hat{\boldsymbol{\theta}}^{(j)})^2 > 0.5\lambda$ **then**
    $w_i^{(j)} \leftarrow \frac{0.5\lambda}{(y_i - \mathbf{x}_i'\hat{\boldsymbol{\theta}}^{(j)})^2}$;
  else
    $w_i^{(j)} \leftarrow 1$;
  end
  $converge \leftarrow max(||\hat{\boldsymbol{\theta}}^{(j)} - \hat{\boldsymbol{\theta}}^{(j-1)}||_\infty, ||\mathbf{w}^{(j)} - \mathbf{w}^{(j-1)}||_\infty) < \epsilon$;
  $j \leftarrow j + 1$;
end
$\widehat{Y}_{\text{PWSGD}}(\mathbf{x}_n) \leftarrow \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{\theta}_n^{\text{converged}}$

Algorithm 1: The PWSGD for the n^{th} batch

2.2.2 Tuning parameter selection

The tuning parameter plays a pivotal role in controlling the amount of outlier identifications and the penalty in the objective function, thereby significantly influencing the performance of PWSGD. As depicted in Equation (2.9), the value of the individual weight w is governed by λ. Specifically, as λ decreases, an observation is more likely to be classified as an outlier with a smaller individual weight w. On the other hand, Equation (2.7) illustrates that λ regulates the strength of the penalty imposed on the individual weight w. Consequently, a larger λ results in a more substantial penalty on the individual weight w. To balance the individual weight w and the penalty strength, it is imperative to determine the optimal value for λ.

One of the outputs from PWSGD is a list of potential outliers from the data set. By employing stability selection, we aim to find the optimal tuning parameter that produces the most stable outlier detection. Several popular stability selection methods have been widely discussed, including data splitting, bootstrap, and random weighting. While all of them have proven to be useful for stable variable selection, random weighting [63] [65] is the most appropriate for performing outlier detection, as we need to obtain weights for all observations in each mini-batch, while the other two methods leave some observations out during the process. The random weighting method is also used in the paper by Gao and Fang [26] to perform tuning parameter selection. Given a λ value, the subset $\mathcal{O}(\lambda; \mathcal{Z})$ denotes the outlier selection from the input data set $\mathcal{Z}$. Suppose that the data set $\mathcal{Z}$ is perturbed into two data sets, $\mathcal{Z}^{*1}$ and $\mathcal{Z}^{*2}$, then ideally the output outlier selection of these two perturbed data sets, $\mathcal{O}(\lambda; \mathcal{Z}^{*1})$ and $\mathcal{O}(\lambda; \mathcal{Z}^{*2})$, should be similar. If $\mathcal{O}(\lambda; \mathcal{Z}^{*1})$ and $\mathcal{O}(\lambda; \mathcal{Z}^{*2})$ turn out to be very different, neither of the output is reliable. Therefore, we seek the λ value that

produces the most agreement between the two outlier selections from two perturbed data sets.

We employ the perturbation method in Fang et al. [19], which is akin to the resampling random weighting method introduced by Rubin [57] referred to as the Bayesian bootstrap in the offline setting —- to perturb the input data set $\mathcal{Z}$. Let $\omega_1, \omega_2, ..., \omega_n$ be random weights from i.i.d. distributions with the first moment and variance being ones (i.e. $E(\omega_i) = 1$ and $Var(\omega_i) = 1$). Notice that this is a common condition on random weights [20]. Additionally, as discussed in Fang et al. [19], we can use a Poisson distribution $Poisson(1)$ as $n \to \infty$ to generate the desired perturbed samples. Denoting $\boldsymbol{\omega}$ as a set of all random weights, we derive the objective function to estimate the model parameters and individual weight with perturbation as follows:

$$(\widehat{\mathbf{w}}(\lambda, \boldsymbol{\omega}), \widehat{\boldsymbol{\theta}}(\lambda, \boldsymbol{\omega})) = \underset{\boldsymbol{\theta} \in \mathbb{R}^{\mathbf{p}}, \mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \omega_i w_i^2 (y_i - \mathbf{x}_i' \boldsymbol{\theta})^2 + \lambda \sum_{i=1}^{n} |1 - w_i| \right\}. \tag{2.10}$$

For any two sets of random weights, $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_1$, we obtain the corresponding estimate for the individual weight for all observations, $\widehat{\mathbf{w}}(\lambda, \boldsymbol{\omega}_1)$ and $\widehat{\mathbf{w}}(\lambda, \boldsymbol{\omega}_2)$, which yield the respective outlier sets, $\mathcal{O}(\lambda; \boldsymbol{\omega}_1)$ and $\mathcal{O}(\lambda; \boldsymbol{\omega}_2)$. To measure the agreement between the two sets of outliers, we adopt the idea of finding the degree of agreement between two judges' nominal judgement on multiple events proposed by Jacob Cohen [13], the Cohen's kappa coefficient:

$$\kappa((\lambda; \boldsymbol{\omega}_1), \mathcal{O}(\lambda; \boldsymbol{\omega}_2)) = \frac{p(\text{agreement}) - p(\text{chance})}{1 - p(\text{chance})}. \tag{2.11}$$

In other words, we use Cohen's kappa coefficient to measure the proportion of agreement between two sets with the agreement by chance being factored out. When the agreement

between the two sets equals to the agreement by chance, $\kappa((\lambda;\boldsymbol{\omega}_1),\mathcal{O}(\lambda;\boldsymbol{\omega}_2)) = 0$. The larger the coefficient, the more agreement there is between the two sets of outliers. The largest coefficient can be 1, indicating that the two sets of outliers are completely identical. When the two sets are in large disagreement, κ could be less than zero, which is of less interest, as discussed by Cohen. Repeating the same process B times to find the agreement between two outlier sets being perturbed by $\boldsymbol{\omega}_{b1}$ and $\boldsymbol{\omega}_{b2}$, $b = 1, 2, ..., B$, we acquire the estimated stability of the outlier detection by averaging Cohen's kappa coefficients for a given λ:

$$\widehat{S}(\lambda) = \frac{1}{B}\sum_{b=1}^{B} \kappa((\lambda;\boldsymbol{\omega}_{b1}),\mathcal{O}(\lambda;\boldsymbol{\omega}_{b2})). \tag{2.12}$$

The repetition process guarantees stable selection of outliers. We secure the optimal λ value that maximizes the stability, $\widehat{S}(\lambda)$.

2.3 Simulation Studies

We demonstrate the proposed method through simulation studies and compare its performance with SGD.

Example 1

We design our data set to have a dimensionality of 10, i.e., $p = 10$. The covaraites vector of each observation i, $\mathbf{x}_i \in \mathbb{R}^p$, where $i = 1, 2, ..., n$, and n is the total number of observations in a batch or in the entire data set, is generated from a ten-dimensional independent normal distribution, i.e. $\mathbf{x}_i \sim \mathcal{N}_{10}(0, I_{10})$, where I_{10} is a ten-by-ten identity matrix. We set the underlying model parameter following the example from

Fang [18]:

$$\boldsymbol{\theta}^* = (\overbrace{\mu, ..., \mu}^{q \text{ components}}, \overbrace{-\mu, ..., -\mu}^{q \text{ components}}, 0, ..., 0)', \tag{2.13}$$

where q determines the amount of informative covariates, and is randomly selected between 1 and half of the total dimensionality, i.e. 5 in our case. μ is the effective size, which determines the strength of information provided by the informative covariates. A smaller μ implies less impact of covariates on predicting future observations. In this simulation study, we set $\mu = 3$. The underlying response variable for each observation i, y_i, where $i = 1, 2, ..., n_0$, is calculated using the underlying model parameter with a linear model system:

$$y_i = X_i' \boldsymbol{\theta}^* + \varepsilon_i, \tag{2.14}$$

where ε_i, the random error of observation i, is generated from a standard normal distribution, i.e. $\varepsilon_i \sim \mathcal{N}(0, 1)$. The size of the initial batch, n_0, is set to be 100. A 20% contamination proportion is introduced into the initial batch to mimic the real-life situation with the following method:

$$Y_{\text{contaminated}} = Y + Y_{\max}, \tag{2.15}$$

and the contamination observations are randomly selected. Subsequently, we obtain the initial estimation of the model parameter vector, $\widehat{\boldsymbol{\theta}}_0$, by applying linear regression method with the R function `lm()` for SGD, and by applying robust linear regression model with the R function `rlm()` for PWSGD. This differentiation arises from our assumption that potential outliers exist in the data set, a scenario not considered by SGD.

We perform SGD and PWSGD following mini-batch mechanism. After the first

batch, mini-batches with a fixed size continue to be fed to the two algorithms to simulate the SGD process. Each mini-batch has the same dimensionality as the initial batch, $p = 10$, with a size of $n = 20$, and there are $B = 200$ batches in total. For each new batch, we use the same contamination method in Equation (2.15) to contaminate at a certain level, we denote this level, or the contamination proportion, as p_{cont}. In Example 1, we explore five different contamination levels to demonstrate the algorithm performance, including $0\%, 10\%, 20\%, 30\%$, and 40%, or $p_{\text{cont}} = 0, 0.1, 0.2, 0.3$, and 0.4. Additionally, we include a scenario where the contamination level is random in each mini-batch. In this case, we randomly select one contamination level from all the aforementioned proportions and use the randomly selected percentage for the current mini-batch. With each batch, we update the model parameter using both PWSGD and SGD, and evaluate the method performance using mean squared error (MSE) of the model parameters defined as follow:

$$MSE_i = \frac{\sum_{j=1}^{p}(\widehat{\theta}_{ij} - \theta^*_{ij})^2}{p}, \qquad i = 1, 2, ..., B. \tag{2.16}$$

The same process is repeated 50 times, and the average is taken over the repetitions on the MSEs for each iteration.

A byproduct from our method is creating outlier labels. The proposed method produces the corresponding estimated individual weight in each mini-batch using Equation (2.9). We then provide a cutoff value, w_c, to classify outliers. If the individual weight is less than the threshold w_c, the corresponding observation is classified as a contaminated case, otherwise, an uncontaminated case. A grid of cutoff values are explored in this simulation study, and a graph of sensitivity (or true positive rate) against the cutoff value is produced. Sensitivity measure the proportion of the

outliers correctly identified as contaminated cases. This metric is also known as the recall rate elsewhere, providing a measure of the accuracy of the classification method conditioned on the positive cases. Sensitivity can be calculated as the follows:

$$\text{Sensitivity} = \frac{|\text{Outliers being correctly identified}|}{|\text{True outliers}|}, \tag{2.17}$$

where $|\cdot|$ is the cardinality of the enclosed set. The higher the sensitivity, the more accurate the classifier. The final sensitivity for each cutoff value is the average over 50 repetitions. We also examine the receiver operating characteristic (ROC) curve, a graph of sensitivity against the false positive rate. The false positive rate measures the proportion of non-outliers that are misclassified as outliers and can be expressed as $1-$ specificity. The specificity (or the true negative rate) measures the proportion of non-outliers that are correctly classified as non-outliers and is calculated with the following formula:

$$\text{Specificity} = \frac{|\text{Non-outliers being correctly identified}|}{|\text{Non-outliers}|} \tag{2.18}$$

The ROC curve illustrates the performance of the classifier at all cutoff values, and the area enclosed by the ROC and the horizontal axis is called the area under curve (AUC). A concave-down ROC curve with a larger AUC indicates high accuracy in identifying outliers without misclassifying non-outliers, signifying good classifier performance. Conversely, if accuracy in classifying both outliers and non-outliers is low, the ROC curve appears concave up with a smaller AUC, suggesting worse performance than a random guess. A diagonal line with a slope of 1 corresponds to a random guess on the graph. If an ROC curve is above the line, then we say the classifier works better

than a random guess, otherwise, it performs worse. A larger AUC signifies better identification of true outliers and true non-outliers. Specificity values, using all cutoff values, are also averaged over 50 repetitions, similar to the sensitivity values.

As important is the tuning parameter λ being discussed in Section 2.2.2, we investigate how different timing for tuning the optimal λ could affect the efficiency and accuracy of the proposed method. We consider four different timings:

1. Tune only the first batch after the initial batch;
2. Tune every batch after the initial batch;
3. Tune every k^{th} batch after the initial batch; or
4. Tune only the first k^{th} batch after the initial batch.

When tuning the λ value, we assume a contamination proportion. We believe that 20% is a reasonable assumption for the contamination level, despite the actual contamination proportion. The learning rate for SGD and PWSGD updates in Equation (2.1) and Equation (2.8), respectively, is of common interest among early scholars as it plays a vital role in the SGD procedure for determining the update step size. Both Benveniste et al. [3] and Moulines and Bach [47] discussed that a misspecified learning rate result in divergence in the mean squared error (MSE) with respect to convexity and Lipschitz parameter of the function. Toulis et al. [68] argued that the information loss depends on the learning rate. Toulis et al. [69] demonstrated and Fang et al. [19] discussed that $\alpha = 2/3$ yields optimal results in convergence rate in both cases where the objective function is strongly convex and non-strongly convex. This extends the results from Ruppert [58], Xu [74], and Moulines and Bach [47]. For a fixed batch size, if the the fixed learning rate is small, the learning rate sequence will be pair-wisely small, leading

to a small update step size. This may prolong computation time to reach the optimal value and sometimes result in a local optimal value rather than a global optimal value. Conversely, if the fixed learning rate is large, the learning rate sequence will be pairwise large. Although this could expedite reaching the optimal value, it may also cause erratic behavior in the updating process and result in non-convergence. Therefore, PWSGD and SGD are quite sensitive to the learning rate, and a misspedified learning rate value could lead to divergence in both cases. We have compared the result using several values of the fixed learning rate value, from 0.07 to 0.12 with an increment of 0.01, when random contamination proportion is applied to the mini-batches. In all tuning parameter timings, the results are quite similar. The MSE curves of SGD and PWSGD using different γ values when λ is tuned only in the first mini-batch are displayed in Figure 2.1. PWSGD appears relatively insensitive to changes in the γ values in the graph, however, a minor variation can still be observed. If we increase the tuning frequency, PWSGD appears even less sensitive to the change of γ value. We observe that as the learning rate increases, the MSE of both SGD and PWSGD increases to respective extend. For all following analysis, we demonstrate the result using a fixed learning rate $\gamma = 0.08$. The learning rate for both the proposed method and the SGD method follow that described in Equation (2.2) with $\alpha = 2/3$, aligning with the conventional setting as those on SGD.

Example 2

Example 2 is an extension of Example 1 where the tuning parameter λ is tuned every k batches. Example 1 illustrates a scenario with a relatively small data set, however, SGD proves more beneficial for handling larger data sets. Therefore, we design Example 2 to have a larger data set than Example 1. For large data sets,

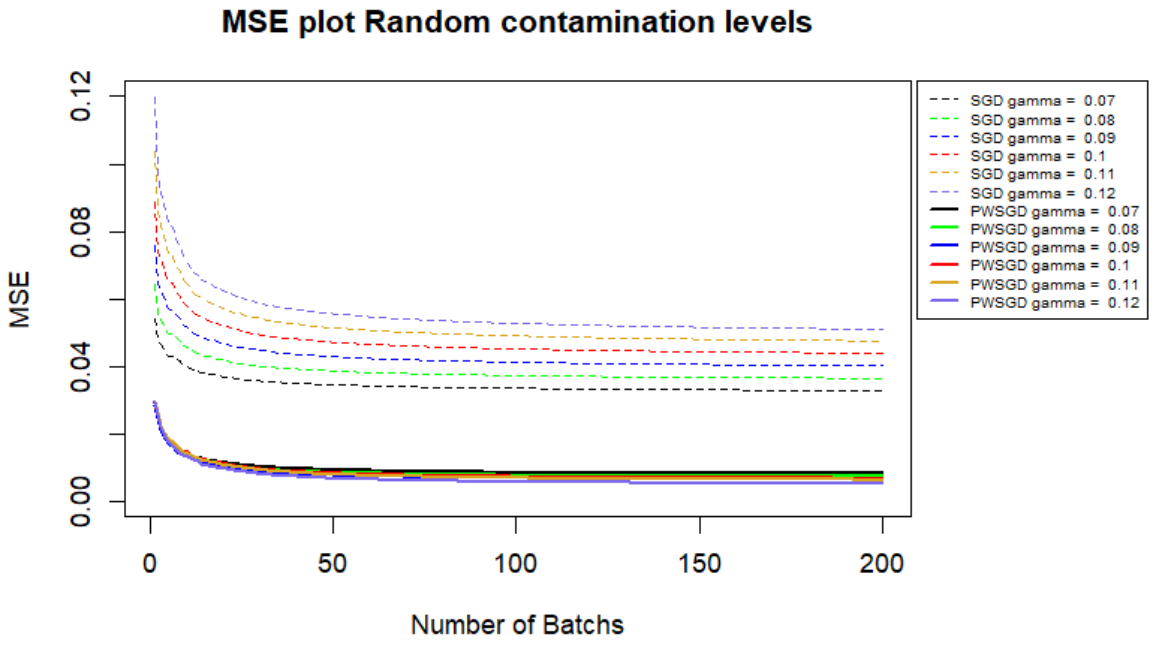


Figure 2.1. MSE of PWSGD and SGD with different fixed learning rate when λ tuned only once with the first batch

tuning the parameter in every batch may be time-consuming, as observed in Example 1. However, tuning the parameter only once for the entire data set or for the first k observations may be overly optimistic. Therefore, in Example 2, we consider only the third tune timing, where λ is tune for every k batches. To accommodate varying data sizes without specifying a fixed k value, we employ a percentage, denoted as p_k. This allows us to tune λ once for every $p_k \times B$ batches, providing adaptability for data sets of different sizes. Notice that a higher p_k implies less frequent tuning for λ over the entire data set. In addition, we increase the initial batch size from 100 to 5000 to maintain the ratio of batch sizes. We investigate the impact of the tuning parameter frequency with several settings in this example:

1. let $p_k = 0.1$, i.e. tune 10% of the entire data set, and generate $B = 2,000$ mini-batches in total with each mini-batch having a size $n = 1,000$;

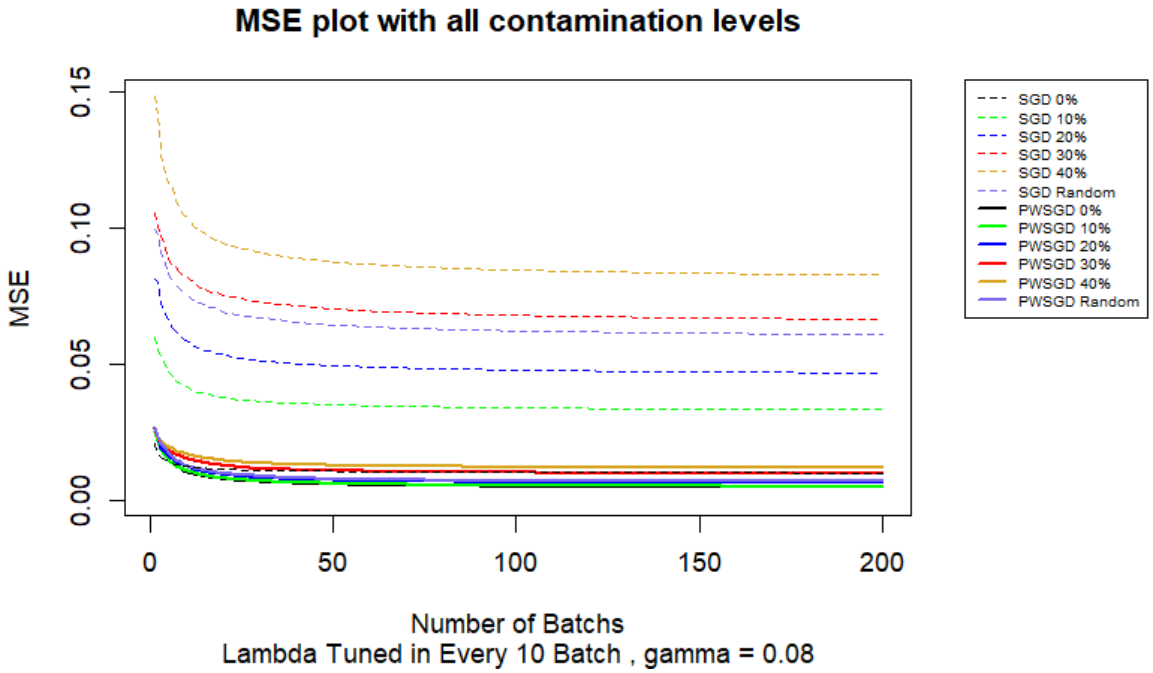
2. let $p_k = 0.2$, i.e. tune 20% of the entire data set, and generate $B = 2,000$ mini-batches in total with each mini-batch having a size $n = 1,000$;

3. let $p_k = 0.3$, i.e. tune 30% of the entire data set, and generate $B = 2,000$ mini-batches in total with each mini-batch having a size $n = 1,000$.

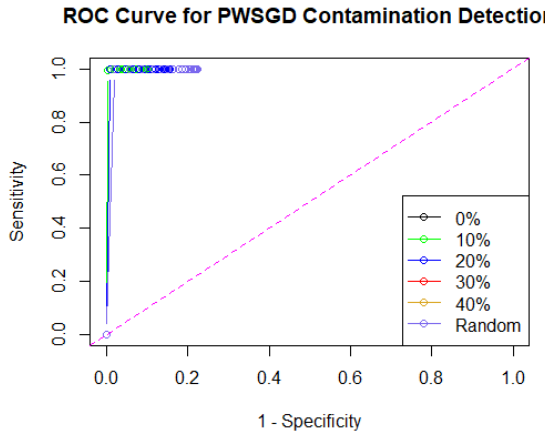
All other settings not explicitly mentioned remain consistent with those of Example 1.

Results

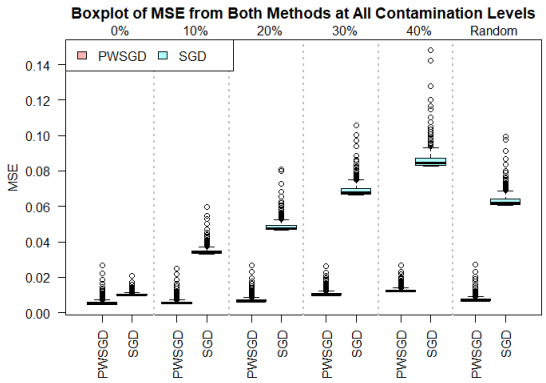
The results obtained from tuning λ with different timings in Example 1, with all other parameters held constant, show subtle differences. This is likely due to the fact that the generated data is very clean, with all observations are from the same distribution and being mutually independent. While this may not be representative of real-life data,



(a) Learning Curve



(b) ROC Curve



(c) Boxplot of MSEs of all batches

Figure 2.2. Performance of PWSGD and SGD at all contamination levels when $\gamma = 0.08$ and λ being tuned every 10 batches in Example 1. (a) is the MSE values of both PWSGD and SGD at five different contamination proportion rates over all mini-batches of size 20; (b) demonstrates the ROC curve of classification of outliers using PWSGD; and (c) shows the boxplot of all MSE's from PWSGD and SGD at all five contamination proportion rates.

we consider tuning every k observations to be the most time-efficient while maintaining adequate performance. Figure 2.2 exhibits the results with $\gamma = 0.08$, where λ value is tuned every 10 batches. Results with other tune timing are demonstrated in Supplementary Material.

Figure 2.2(a) provides the MSE change in both methods over the number of batches at five contamination proportions. The solid line indicates the result of the proposed method and the dashed line presents the performance of SGD. It is evident that our method outperforms SGD by a significant margin when there is data contamination in the data set. Despite SGD initially performing better than our method when there is no contamination in the data set, our method quickly surpasses SGD within the first 10 batches and maintains a consistently superior performance. This can be observed from Figure 2.2(c), where, when the contamination proportion is 0%, there are several MSE's on the higher end of the boxplot for PWSGD that exceed the highest MSE boxplot of SGD, but overall, PWSGD achieves lower MSE than SGD. When there is data contamination, MSE obtained by PWSGD distinctly surpasses SGD. Figure 2.2(b) demonstrates the ROC curve of PWSGD method on outlier recognition. It shows that almost the outliers and non-outliers are correctly being identified at a very high rate. The false positive rate or 1 - Specificity barely exceeds 0.2, indicating that our method rarely misclassifies non-outliers. Assuredly, tuning λ every k batches seems the most reasonable approach for real data analysis. Therefore, it is important that we examine if changing the frequency would affect the performances of the two methods. The comparison of MSEs also turns out to be similar to Figure 2.2(a).

We observed that, across all contamination proportions, PWSGD consistently achieves the highest sensitivity of 1 with the smallest provided cutoff value, 0.05. The AUC of the ROC curves remains very large and similar for all contamination

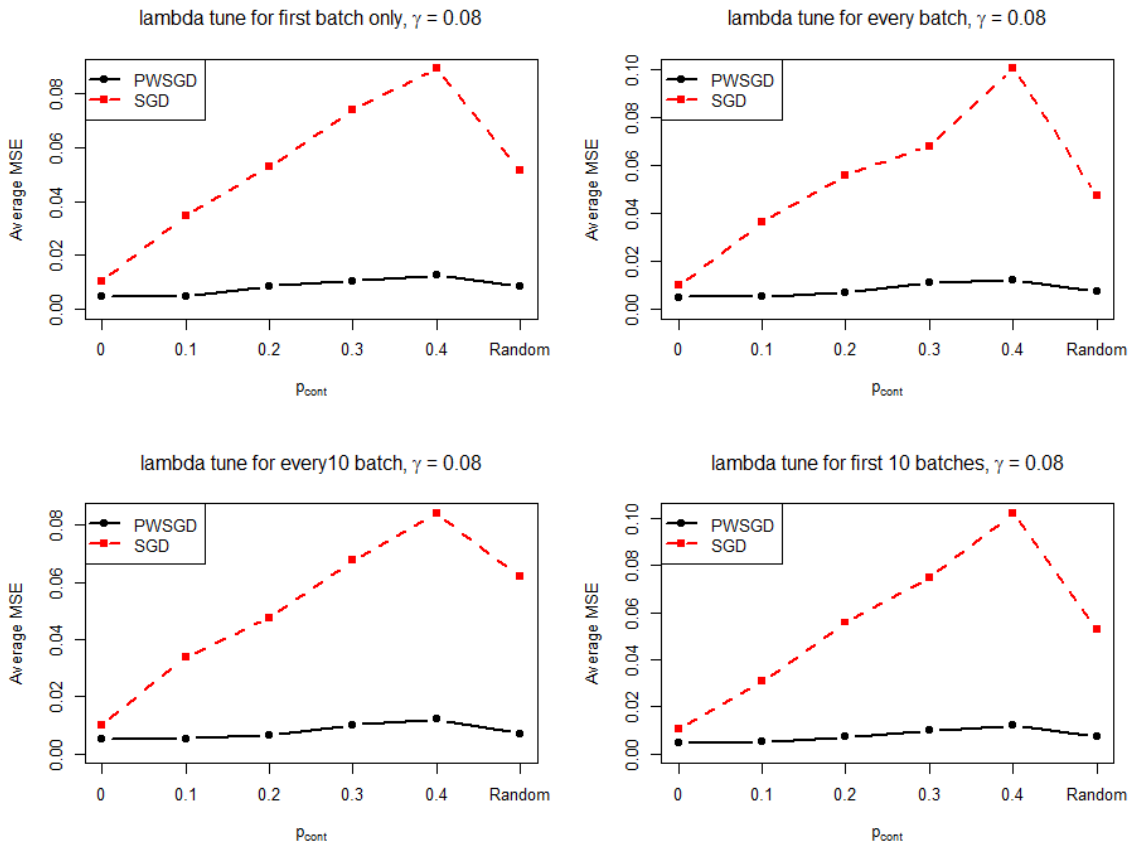


Figure 2.3. Average MSE of PWSGD and SGD in Example 1 at all contamination levels when λ tuned for the first batch only, for all batches, every 10 batches, and for the first 10 batches

proportions. The AUC of the ROC curves remains very large and similar for all contamination proportions. Even misspecifying the contamination proportion, PWSGD demonstrates stability in the face of increasing contamination disturbance.

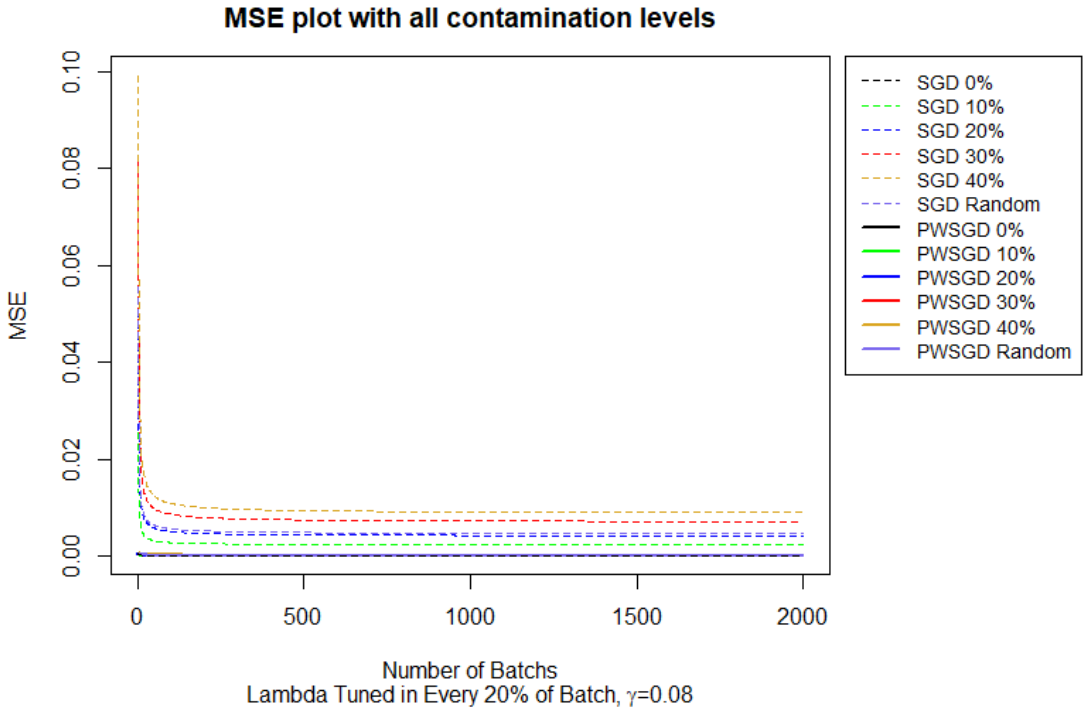
Figure 2.3 demonstrates the comparison of the change in average MSE when applying SGD and PWSGD over contamination levels in all tuning time scenarios mentioned in Example 1 (2.3) with $k = 10$. In the figure, the black solid line is the average MSE of PWSGD and the red dashed line is the average MSE of SGD. All scenarios have a fixed learning rate $\gamma = 0.08$. It is evident that PWSGD exhibits

robustness compared to SGD across all scenarios. The average MSEs of SGD remain relatively consistent, whether dealing with random contamination proportions or a fixed contamination proportion of 20%. This consistency is primarily attributed to the expected contamination proportion rate being 20% when the contamination proportion is uniformly selected.

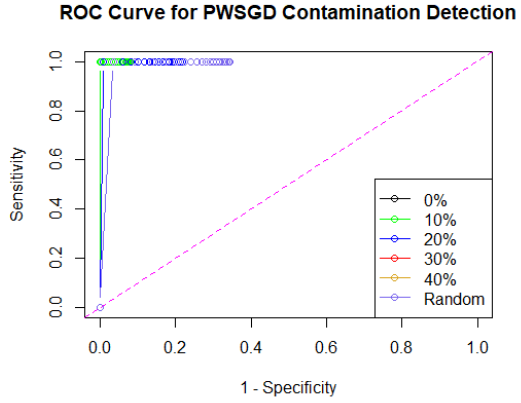
We experimented with different batch sizes to assess their impact on the performance of both methods. Starting with $n = 20$, then slowly increased to $n = 50, 100$, and $n = 1000$ to address the challenges of the big data era in Example 2. Despite the relative similarity in results, it became apparent that SGD is more sensitive to increases in batch size, whereas PWSGD demonstrates remarkable stability. This observation is evident when comparing the MSE learning curve plot and the MSE boxplot.

In our various attempts with different γ values, a noteworthy finding is that PWSGD performs poorly when the γ value is excessively large, while SGD tends to exhibit better learning. This observation aligns with the discussion in Section 2.3, highlighting PWSGD's sensitivity to the choice of the learning rate. Our experiments indicate that PWSGD is more sensitive than SGD to the learning rate, with a preference for smaller values to converge effectively. If the learning rate becomes too large, the performance of PWSGD becomes erratic. Additionally, our experimental findings suggest that the learning rate is influenced by the mini-batch size, the initial batch size to mini-batch size ratio, and the tuning frequency. As the mini-batch size increases, while keeping other settings, including initial batch size and the tune timing, constant, an optimal γ value should be smaller to maintain stable performance.

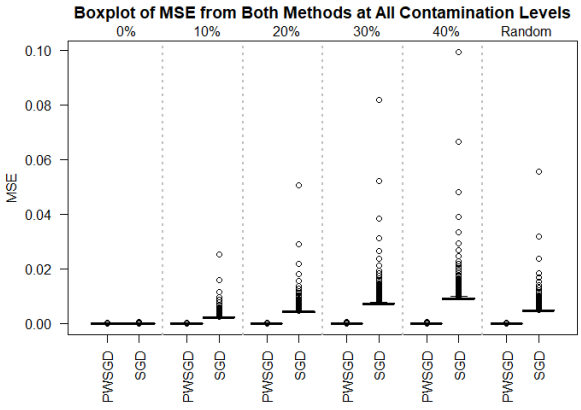
Similar to Example 1, the variance in tuning frequency did not yield significant differences in outcomes. Figure 2.4 presents the result with $\gamma = 0.08$, and λ value is



(a) Learning Curve



(b) ROC Curve



(c) Boxplot of MSEs of all batches

Figure 2.4. Performance of PWSGD and SGD at all contamination levels when $\gamma = 0.08$ and λ being tuned every 20% batches in Example 2. (a) is the MSE values of both PWSGD and SGD at five different contamination proportion rates over all mini-batches of size 1000; (b) demonstrates the ROC curve of classification of outliers using PWSGD; and (c) shows the boxplot of all MSE's from PWSGD and SGD at all five contamination proportion rates.

tuned every 20% batches, the result with other tuning frequencies are demonstrated in Supplementary Material. In Figure 2.4(a), it appears that the overall and converged MSE are smaller than those in Example 1, and the MSE of SGD converges faster than it does in Example 1. This faster convergence may be attributed to a larger mini-batch size. In accordance with Example 1, PWSGD consistently outperforms SGD in all contamination proportion settings by acquiring smaller MSEs. The boxplot of the MSEs in Figure 2.4(c) shows that when the data set is large PWSGD and SGD performs very similarly when there is no contamination in the data set. However, PWSGD achieves a slightly smaller MSE compared to SGD. With contamination present, the performance of SGD deteriorates, whereas PWSGD maintains a high level of accuracy. The outlier detection accuracy is outstanding, as indicated by the ROC curve having a very large AUC in Figure 2.4(b). This suggests that PWSGD detects outliers correctly with a very low false positive rate.

We compare the average MSEs of both methods in the three settings in Example 2. The comparison mirrors that in Example 1, although we observe minimal variations in the average MSE from PWSGD as the contamination proportion increases in Example 2. Through the simulations with both small and large data sets, we demonstrate the robust stability of the proposed method across different data scenarios. Furthermore, the outlier detection performance remains consistently stable.

2.4 Real data analysis

In this section, we apply PWSGD to the gas sensor array under dynamic gas mixtures data set available on UCI machine learning repository [22]. The data was collected at the ChemoSignals Laboratory in the BioCircuits Institute, University of California San

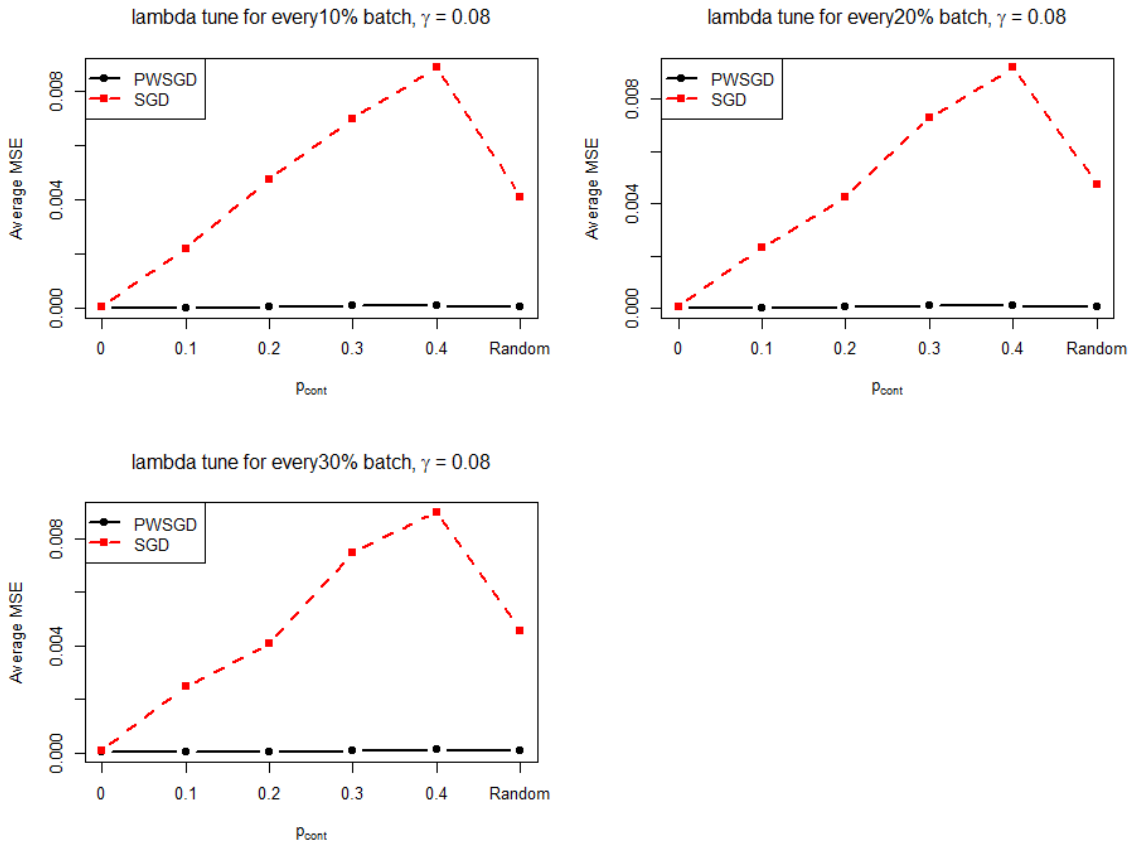


Figure 2.5. Average MSE of PWSGD and SGD in Example 2 at all contamination levels when λ tuned every 10% batches, every 20% batches, and every 30% batches

Diego by Fonollosa et al. [23]. The purpose was to improve the accuracy for detecting chemical mixtures from the sensor measurements as early as possible in time. We take the data set that contains the time series readings from 16 chemical sensors exposed to ethylene and Carbon Monoxide (CO) gas mixtures at different concentration levels, with concentration levels changing randomly. The chemical sensor arrays were placed in a 60 ml measurement chamber, where the gas sample was injected at a constant flow of 300 ml/min, and the readings are continuous signals over approximately 12 hours without interruption, resulting in $4,208,261$ measurements in total from all sensors. The 16 chemical sensors contains four different types: TGS-2600, TGS-2602, TGS-2610, TGS-2620, and there are four of each type.

The use of the chemical sensor data set is inspired by Wang et al. [72], who applied a novel subdata method to overcome potential covariance issues and computational challenges associated with large-sized data when employing linear regression analysis. Eo et al. [17] also leveraged this data set in their research, with a primary focus on data pre-processing for neural network techniques. In their study, SGD was adopted as the optimizer. The data set was utilized to investigate their method for improving gas concentration estimation accuracy using various neural network methods and the proposed data pre-processing method. For this real data analysis, we compare the estimation performance of our proposed method to that of SGD assuming a linear regression model.

Following Wang et al. [72], all readings from the second sensor were excluded due to mainly containing negative readings for unknown reasons. Additionally, only the readings after the first 4 minutes, which is the system run-in time, were used. A log-transformation was applied to all readings since trace concentration generally follows a lognormal distribution. The readings from the last sensor (`sensor 16`) were

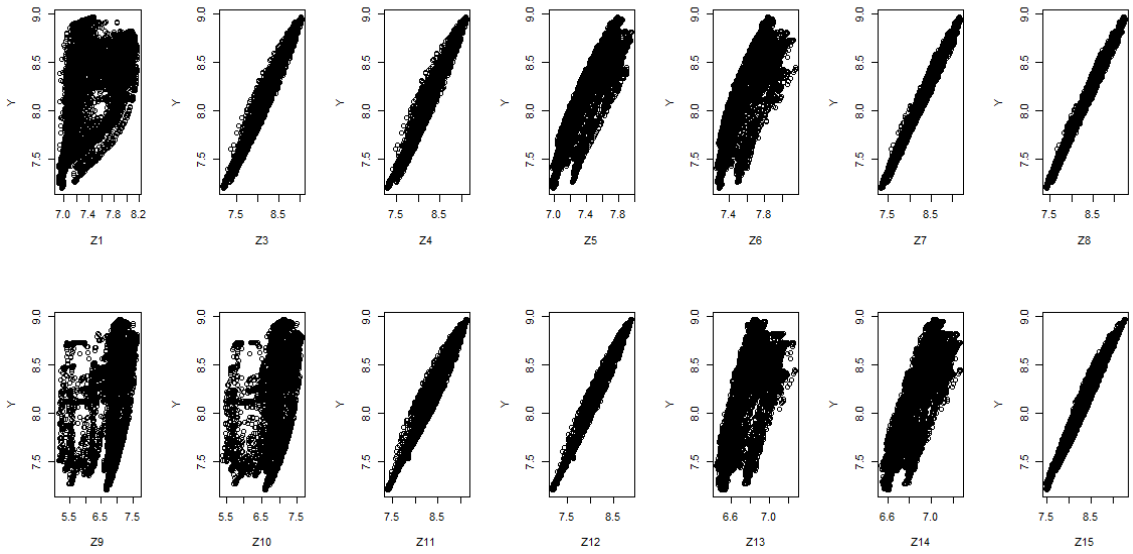


Figure 2.6. Scatter plots of simple random sample of size $10,000$ from the chemical sensors data. Z1 represent sensor 1, Z3 represent sensor 3, etc..

selected as the response variable, while the rest were considered as covariates. Applying the same data preprocessing steps, the data set used for data analysis consists of $4,188,261$ observations with 14 covariates from the other sensors except `sensor 16`. Randomly select $10,000$ observations from the data set, Figure 2.6 demonstrate the scatter plot of these obervations. Notably, some sensor readings exhibit strong linear relationships with `sensor 16`. This may be attributed to these sensors being of the same type as `sensor 16` or similar types. The other sensor readings do not hold strong linear associations with `sensor 16`, but a moderate linear association may exist.

We apply both PWSGD and SGD to implement the linear regression analysis, investigating the correlation between `sensor 16` with the other sensors. The estimation performance of the two methods is compared using the mean squared prediction error

(MSPE). MSPE is calculated in the following method:

$$\text{MSPE} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2, \tag{2.19}$$

where n is the number of observations being estimated, $\widehat{y}_i$ is the estimated logged reading of `sensor 16` by either the proposed method or SGD for observation $i = 1, 2, ..., n$, and y_i is the true logged reading of `sensor 16` for the i^{th} observation. The initial batch contains the $8,261$ instances and each mini-batch after the first batch contains 2000 observations, resulting in $2,090$ mini-batches. We assume the initial estimation for the model parameters is zero for both methods, let $\gamma = 0.08$, and assume a contamination proportion of 20% in each batch. The tuning parameter λ is tuned every 20% of the batches, that is, we tune λ once for every 418 batches. With each mini-batch, we update the estimation of the model parameters and use the updated model parameter to estimate the logged reading of `sensor 16` for the next mini-batch. The true logged reading from `sensor 16` of the next mini-batch is compared against the estimations from the two methods, and a MSPE value is calculated using Equation (2.19).

Figure 2.7 demonstrates the MSPE of both methods over all iterations in varying scales. In the top left corner, all MSPEs from both methods are displayed, highlighting the substantial difference in the initial estimation using SGD from the underlying logged readings of `sensor 16`. The top right corner provides a zoomed-in view of the MSPEs of both methods. Notably, the MSPE behavior from SGD appears erratic, and it remains challenging to observe the variation in MSPE using PWSGD. Moving to the bottom left corner, we present the MSPEs from both methods zoomed in to the scale of PWSGD only. In this view, the information from SGD is absent. The

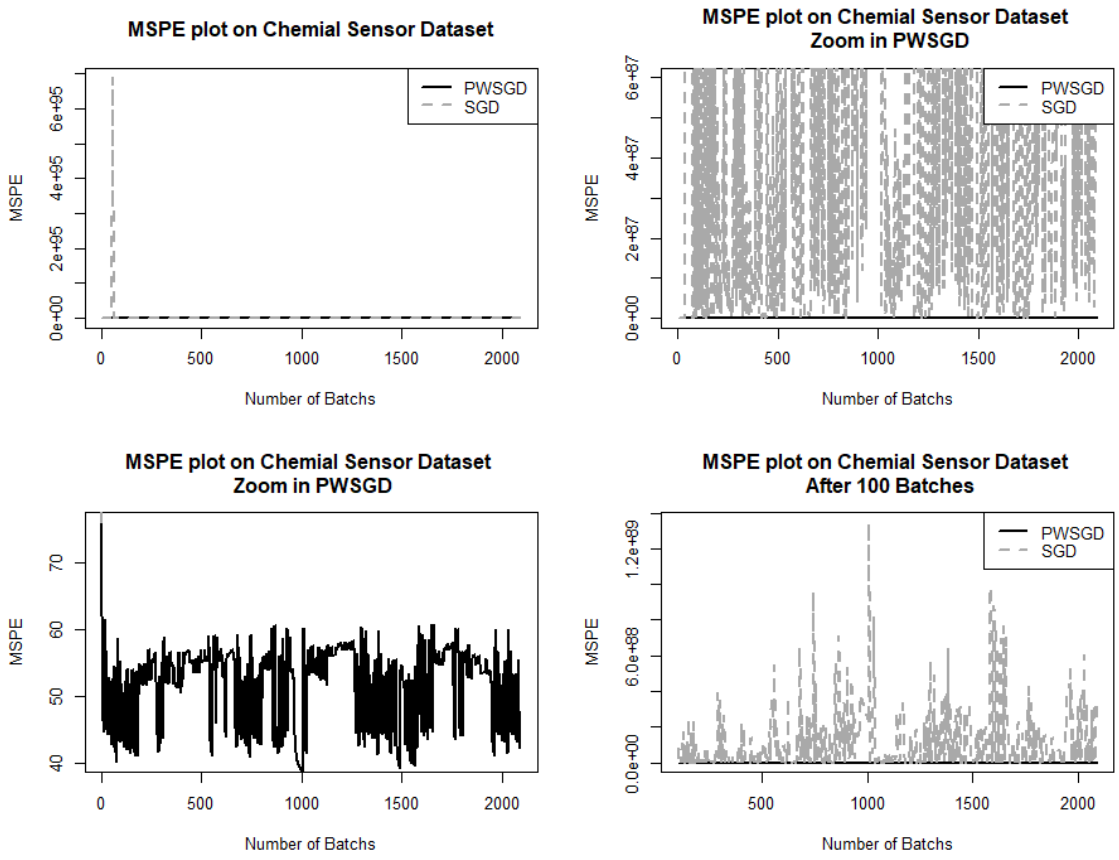


Figure 2.7. MSPE of PWSGD and SGD applied to Ethylene CO Analysis when $\gamma = 0.08$, tune very 20% batches.

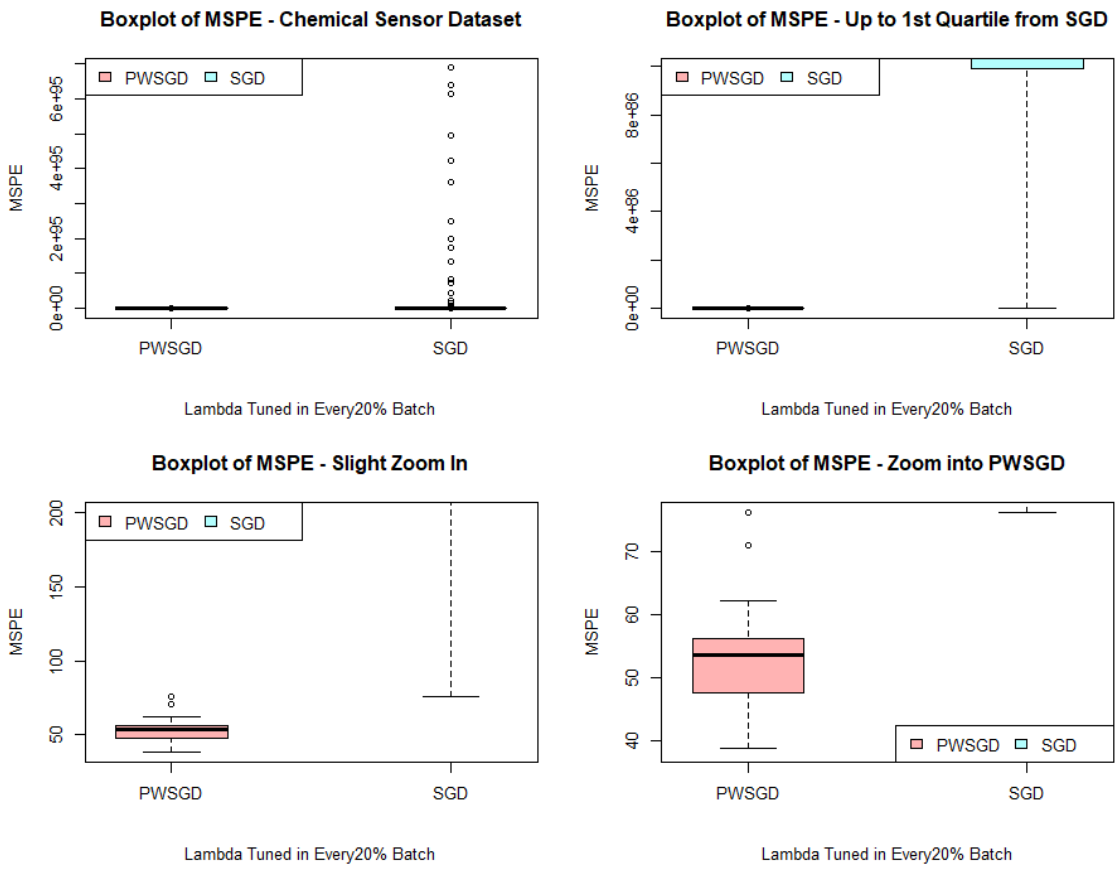


Figure 2.8. Box plot of MSPE of PWSGD and SGD from all iterations applied to Ethylene CO Analysis when $\gamma = 0.08$, tune very 20% batches.

maximum MSPE is observed from the first batch. Given that we assumed all zero values for the model parameters in the initial batch, the spike in MSPE in the first mini-batch is reasonable. Finally, the bottom right graph illustrates the MSPEs from both methods after 100 batches. The MSPEs from SGD exhibit increased stability compared to the top left graph. However, the scale remains large.

Figure 2.8 displays MSPEs using both methods to estimate the logged readings of `sensor 16` in boxplots. The top left graph illustrates all MSPEs from the two methods, where it is evident that the variation using the proposed method is significantly lower than with original SGD. From the bottom right graph, it can be seen that the largest MSPE using PWSGD is approximately the same as the smallest MSPE applying SGD.

It is evident that even when the relationships between the response variable and the covariates appear to be linear, PWSGD outperforms SGD when assuming a linear regression model in a real-life data set.

2.5 Summary

PWSGD builds upon SGD, which is a potent optimization tool for handling large amounts of data or streaming data. In a least squares regression setting, SGD alleviates the challenges of inverting covariance matrices caused by huge data sets or the inability of least squares regression models to handle data streams. However, SGD is susceptible to the impact of outliers or data contamination. PWSGD incorporates penalized weighted methods by assigning an individual weight to each observation and enforcing a lasso-like penalty on the individual weight. The individual weight acts as a regulator to control the influence of each observation on the estimation of the model parameters based on the residual. Thus, when an observation appears to

be a potential outlier, its contribution to the estimation is limited. Meanwhile, when the individual weight is very small, indicating a potential outlier, the corresponding observation is labeled by our algorithm, allowing for further analysis and actions based on this information. The penalization on the individual weight regulates the amount of data contamination identified by the algorithm, with the penalization strength controlled by a hyperparameter λ. These modifications based on SGD enhance its robustness to data contamination, thereby improving the estimation of model parameters. Consequently, predictions using the estimated model parameters become more accurate. This improvement is evident from our experiments with simulated data and real data analysis using the chemical sensor data set, where in all examples, PWSGD outperforms SGD in terms of MSE and MSPE.

We believe refined algorithm is valuable, given the widespread use of SGD across various machine learning methods and its broad adoption in many fields. PWSGD has demonstrated its effectiveness in improving estimation or prediction under a linear regression setting, as well as in the estimation or prediction of other algorithms. A future avenue for exploration involves applying PWSGD to a heterogeneous model with a variance function, moving beyond the homogeneous setting covered in this chapter. This extension is particularly relevant for scenarios where outlying responses may not be solely due to error but rather stem from intrinsic variability in the response variable.

Chapter 3: Robust Random Forest

3.1 Overview

Random Forest (RF) by Breiman [7] is very popular in the industry, especially in finance, where complex and messy data exist. This popularity is attributable to its flexibility, which allows it to adapt to various types of data. RF consists of many individual regression or classification trees, each trained on a bootstrapped sample with a randomly selected subset of explanatory variables. The RF prediction for a new observation is the average of the results from all trees in RF. Let θ_t be the random parameter set for tree t, where there are m trees in total in RF. Let (X_i, y_i) be observation i in the training set, where there are n training observations in total. Meinshausen and Ridgeway [46] demonstrated that the prediction from RF for a new observation with $X = \mathbf{x}$ can be written as the weighted average response of the training observations as follows:

$$\widehat{Y}_{RF}(\mathbf{x}) = \sum_{i=1}^{n} w(X_i, \mathbf{x}) y_i, \tag{3.1}$$

$$w(X_i, \mathbf{x}) = \frac{1}{m} \sum_{t=1}^{m} w(X_i, \mathbf{x}, \boldsymbol{\theta}_t), \tag{3.2}$$

where $w(X_i, \mathbf{x}, \boldsymbol{\theta}_t)$ is the weight of training observation i from tree t.

The prediction of RF is highly local and, therefore, robust to data from heavily tailed distributions. However, RF is not robust if the training responses and the testing responses are from different distributions. One phenomenon that could lead to different distributions is when the response variable is contaminated in the training set. In this chapter, we consider the situation where there exists mean-shift contamination on the response variable in the training set. A mean-shift contaminated response variable follows a distinct distribution that has a constant difference in the mean compared to the true distribution. The observations with this type of contamination are normally called outliers. Consequently, the prediction from the RF for a new observation may be affected by the contamination if the outliers are not addressed. According to Hample et al. [32], a data set can contain 1% to 10% contaminated observations. She and Owen [64] discussed that traditional methods to identify outliers, such as studentized residuals or leave one out methods (with Cook's distance or with difference in fits (DFFITS)), work well when dealing with a single outlier but might fail when there are multiple outliers. These traditional methods fail in two ways. One is called masking, where outliers are not identified. The other way is called swamping, where non-outliers are misclassified as outliers. As data is collected from various resources and in different forms at an increasing speed in the era of big data, the issue of outliers becomes more conspicuous. Therefore, identifying outliers and being resistant in the face of data contamination becomes remarkably important.

Numerous scholars have endeavored to enhance the robustness of RF against diverse data contamination. Roy and Larocque [56] showed that applying robust aggregation methods for RF often assists in mitigating the impact of outliers. They illustrated that replacing the mean of the response variable with its rank improves the robustness of RF.

Meinshausen and Ridgeway [46] introduced quantile regression forest (QRF), which substitutes the mean of the response variable with quantiles. Additionally, Meinshausen established that the distribution of QRF prediction approximates, point-wise, the underlying distribution of the response variable. Li and Martin [42] demonstrated that RF predictions can be formulated in generalized loss function form:

$$\widehat{Y}_{RF}(\mathbf{x}) = \underset{\mathcal{S}\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} w(X_i, \mathbf{x})\rho(y_i, \mathcal{S}(\mathbf{x}_i)), \tag{3.3}$$

where $\mathcal{F}$ is a family of functions and $\rho(\cdot)$ is a general loss function. Li and Martin showed that the original RF and the QRF can be expressed by the generalized loss function with squared error loss function and quantile loss function, respectively. Furthermore, Li and Martin proposed robust RF with adaptive weights using pseudo Huber loss function and using Tukey's biweight loss function (hereinafter referred to as RFHuber and RFTukey, respectively). Sage [59] proposed the method that introduces a weight for each training case to enhance the robustness of classification RF. Unlike the approach by Li and Martin, where adaptive weights are recalculated for each new observation, Sage's method computes the weights for the training observations only once, making it applicable to all new observations.

In this chapter, we propose the penalized weighted robust random forest (PWRF) method drawing inspiration from the penalized weighted method from Gao [27], Gao and Fang [26], Gao and Feng [28], and Luo et al. [45]. These studies have demonstrated that the penalized weighted method robustly identifies outliers in the training set, thereby enhancing the estimation and prediction of the corresponding algorithms. PWRF modifies the loss function in Equation (3.3) with a residual based individual weight, d, and introduces a Lasso-like penalty on the logged weight term to regulate the

individual weight assignment. Additionally, given that our method is residual-based, it exhibits versatility in handling various types of data. Moreover, it can be extended to enhance the robustness of other algorithms.

Moreover, PWRF is computational efficient, in theory, compared to RFHuber and RFTukey introduced in Li and Martin [42], when continuously predicting for streaming in new observations in different batches. This efficiency stems from the fact that the individual weight for a training observation in the proposed method is based on the difference between the its estimated response and its underlying response. In contrast, Li and Martin's method derives from the deviation between the estimated response of the new observation and each training observation. Consequently, our method evaluates individual weights and hyperparameters only once for all new cases, while other robust RF methods recalculate their corresponding hyperparameters each time new cases arrive. In experiments conducted in Section 3.3, PWRF consistently achieves highly competitive or even superior results compared to the two robust RF methods, RFHuber and RFTukey. Particularly in simulation studies, PWRF demonstrates stability across different data settings.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce our proposed method, PWRF, demonstrate the framework of the algorithm, and discuss how the tuning parameter is selected. In Section 3.3, we conduct numerical studies to compare the performance of PWRF to three other RF methods (RFHuber, RFTukey, and the original RF). In Section 3.4, we apply all four methods to a real-life financial data set to further investigate and compare their performance. Finally, in Section 3.5 we conclude by summarizing the observations from both the numerical studies and real data analysis.

3.2 PWRF Regression

PWRF alleviates the impact from the outliers in the training set based on the residuals of the training observations. We employ the squared error loss function, $||\mathbf{y} - \hat{\mathbf{y}}||_2^2$, in Equation (3.3), and introduce a weight, $d \in [0, 1]$. This weight, distinct from the weight in the original RF, w, in Equation (3.3),is referred to as the individual weight, denoted as d. The individual weight d_i assigned to an observation $\mathbf{x}_i$ is directly related to its training squared error, $(y_i - \widehat{y}_i)^2$, where $\widehat{y}_i$ is the estimated response for observation $\mathbf{x}_i$. The assignment of the individual weight d to each training observation is described in Equation (3.4):

$$\widehat{d}_i = \begin{cases} 1 & \text{if } (y_i - \widehat{y}_i)^2 \leq \lambda, \\ \frac{\lambda}{(y_i - \widehat{y}_i)^2} & \text{if } (y_i - \widehat{y}_i)^2 > \lambda, \end{cases} \tag{3.4}$$

where $\widehat{y}_i$ is the estimated response for observation $\mathbf{x}_i$. In an ideal scenario, $d_i = 0$ for outliers and $d_i = 1$ for non-outliers. Nonetheless, the individual weight typically range between zero and one: the larger the error, the smaller the individual weight, resulting in less impact on the prediction from new observations Additionally, an ℓ_1 penalty is applied to the log-transformed individual weights to control the number of training observations considered as outliers. The estimation of the individual weight, $\widehat{\mathbf{d}}$, can be summarized as in Equation (3.5):

$$\widehat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} w(X_i, \mathbf{x}) d_i (y_i - \widehat{y}_i)^2 + \lambda \sum_{i=1}^{n} |\log(d_i)| \right\}. \tag{3.5}$$

Furthermore, the prediction from PWRF is a modified version of Equation (3.1):

$$\widehat{Y}_{PWRF}(\mathbf{x}) = \sum_{i=1}^{n} w(X_i, \mathbf{x}) d_i y_i. \tag{3.6}$$

3.2.1 Our Algorithm

Algorithm 2 outlines the framework of the proposed method. With a given training set, we first use the original RF to obtain the Out-of-Bag (OOB) prediction weights, $\mathbf{w}_{OOB}$, and the initial OOB prediction, $\widehat{y}_{OOB}^{(0)}$, for all training observations. Then, the initial OOB prediction is computed as the weighted average of the training observations as described in Equation (3.1). Assuming that the observations are not outliers – assigning 1 to the individual weight d – we update the OOB predictions with Equation (3.6) . At the same time, d is being updated using Equation (3.4) with a given λ. The update process continues until both the prediction and the individual weights converge. The prediction for a new observation $X = \mathbf{x}$ is calculated using the prediction weights w_k from the original RF and the converged individual weights.

3.2.2 Tuning parameter selection

The hyperparameter, λ, plays a crucial role in the PWRF algorithm. This parameter controls both the individual weight in Equation (3.4) and the strength of lasso-like penalty in Equation (3.5). Achieving optimal performance in our method hinges on selecting an appropriate value for λ. In Equation (3.4), λ essentially functions as an outlier classification rule, determining the assignment of individual weights for training observations. If the squared residual of a training observation is less than λ, we classify it as a non-outlier by assigning an individual weight of 1. However, if

Input: out-of-bag (OOB) prediction weights $w_{OOB_{k\neq i}}(i)$ and RF prediction weights w_k from RF, and λ.
Data: x
Result: $d_i = d_i^{(j)}$ and $\widehat{Y}_{PWRF}(\mathbf{x})$
Compute initial OOB predictions by
$\widehat{y}_{OOB}^{(0)}(i) \leftarrow \sum_{k\neq i} w_{OOB_k}(i) y_k,\ for\, i = 1, 2, ..., n.$
let $j = 1$, $d_i^{(0)} = 1$;
while *not converge* **do**
  [Update $\widehat{y}_{OOB}$]
  $\widehat{y}_{OOB}^{(j)}(i) \leftarrow \frac{\sum_{k\neq i} w_{OOB_k}(i) d_i^{(j-1)} y_k}{\sum_{k\neq i}^{n} w_{OOB_k}(i) d_i^{(j-1)}}$;
  [Update d]
  if $(y_i - \widehat{y}_{OOB}^{(j)}(i))^2 > \lambda$ **then**
    $d_i^{(j)} \leftarrow \frac{\lambda}{(y_i - \widehat{y}_{OOB}^{(j)}(i))^2}$;
  else
    $d_i^{(j)} \leftarrow 1$;
  end
  $converge \leftarrow ||\widehat{y}_{OOB}^{(j)} - \widehat{y}_{OOB}^{(j-1)}||_\infty < \epsilon$;
  $j \leftarrow j + 1$;
end
$\widehat{Y}_{PWRF}(\mathbf{x}) \leftarrow \frac{\sum_{k=1}^{n} w_k(\mathbf{x}) d_i y_k}{\sum_{k=1}^{n} w_k(\mathbf{x}) d_i}$

Algorithm 2: The PWRF

the squared residual exceeds λ, indicating a higher potential for contamination or being an outlier, we set the individual weight as the ratio of λ to the squared residual. The magnitude of λ influences the classification rule's strictness. A larger λ increases the likelihood of an observation being classified as a non-outlier, resulting in a looser rule. Conversely, a smaller λ raises the chances of an observation being considered a contaminated case, leading to smaller individual weights for such observations and a stricter rule. Moreover, λ controls the penalty strength on the individual weight in Equation (3.5). A larger λ results in a larger individual weight but a weaker penalty, $|\log(d_i)|$, and vice versa. In essence, λ acts as a lever to balance the behavior of individual weights and the penalty, making it crucial to select an appropriate value for accurate predictions in the algorithm.

Cross-validation is prevalent in machine learning to determine optimal tuning parameters. However, it operates under the assumption that both training and testing observations are drawn from the same distribution. This assumption becomes invalid when outliers are present in the training set but absent in the testing set. To mitigate the impact from the outliers, Sage [59] employed a weighted cross validation method which involves training two additional smaller trees, which requires more computation power. To attain the optimal value for λ given a set of fine grid λ values, we perform a k-fold cross-validation on the training set with all λ values using trimmed mean squared validation error. First, we randomly assign the training observations into k subsets, or k folds, and designate one of the folds as the validation set while the rest compose the "training set". Subsequently, we train a PWRF model on the "training set" with each λ value from the fine grid and evaluate its performance on the validation set to obtain the validation error for this fold. Here, we deviate from the conventional cross-validation method by computing the trimmed mean squared validation error for

each fold instead of the mean squared validation error. This adjustment helps mitigate the impact of outliers. We iterate through each fold to compute k trimmed mean squared validation errors, and then calculate the average trimmed mean validation error across the k folds for each λ. This process is repeated for all λ values several times to reduce randomness. The λ value that yields the smallest average trimmed mean squared validation error across repetitions is selected as the optimal λ value.

3.3 Simulation studies

We conduct numerical studies to compare the performance of our method with three other methods: RFHuber (Li and Martin [42]), RFTukey Li and Martin ([42]), and the original RF (Breiman [7]). These studies comprise three simulation examples, inspired by similar ideas from Sage [59]. The first two examples are based on work by Roy and Larocque [56], and the third example originated from Li and Martin [42]. Example 1 simulates a tree-like mechanism, Example 2 creates a combined non-linear response variable, and Example 3 generates linear response variable. In Example 3, the covariates are correlated with a correlation coefficient of 0.7, unlike in the first two examples, where the covariates are independent. Across all examples, we vary the proportion of data contamination in the training set, demonstrating the prediction performance of the methods under different scenarios. In contrast to the mean (or "shift") contamination from Roy and Larocque [56], where only the random error is contaminated with a larger mean value, in this chapter, we define a mean-shift type of contamination in the response variable. Specifically, we add three times the maximum of the responses to randomly selected observed responses, with the number of selected observations varying by the contamination proportion level. The subsequent sections

provide detailed descriptions of the setting for each of the three examples in detail.

Example 1

In the first example, explanatory variable data is generated from a six-dimensional independent normal distribution, i.e. $X \sim \mathcal{N}_6(0, I_6)$, where I_6 is a six-by-six identity matrix. To mimic a seven-leaf tree mechanism, the response variable is generated as follows:

$$\begin{aligned} Y_i =& m(\mathbb{1}((X_{1i} \leq 0), (X_{2i} \leq 0)) \\ &+ 2\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} \leq 0)) \\ &+ 3\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} \leq 0)) \\ &+ 4\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} > 0)) \\ &+ 5\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0)) \\ &+ 6\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0), (X_{5i} \leq 0)) \\ &+ 7\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0), (X_{5i} > 0)) + \varepsilon_i \end{aligned} \tag{3.7}$$

m is a signal-to-noise ratio, regulating the size of the signal with respect to the random error, ε. As m increases, the signal in the data becomes stronger, resulting prediction is less influenced by the random error. Different levels of signal-to-noise ratio can have varying effect on algorithm performance, and when there is data contamination in the training set, this effect may be magnified. In this example, we utilize four different signal-to-noise ratios: $m = 0.2, 0.4, 0.6, 0.8$, to compare the performance of the methods as the signal in the data set transitions from weak to strong. $\mathbb{1}(\cdot)$ is an indicator function, returning 1 if the logic inside the function is true and 0 otherwise. The random error, ε, is generated from an identical and independent standard normal

distribution, i.e. $\varepsilon \sim \mathcal{N}(0,1)$.

Example 2

Covariates for the second example are generated in the same manner as in Example 1. The response variable is generated with the following function, which creates a combination of linear, non-linear, and tree-like response:

$$\begin{aligned} Y_i = m(X_{1i} + 0.707X_{2i}^2 + \mathbb{1}(X_{3i} > 0) + 0.873\log(|X_{1i}|)X_{3i} \\ + 0.894X_{2i}X_{4i} + 2\mathbb{1}(X_{5i} > 0) + 0.464e^{X_{6i}}) + \varepsilon_i, \end{aligned} \tag{3.8}$$

where m and ε are defined the same as in Example 1, but for Example 2, the values of m are set to $0.15, 0.3, 0.45$, and 0.6.

For both examples, we generate 500 and 1000 observations for the training and the testing set, respectively. We randomly contaminate a proportion, p_{cont}, of the training observations, where $p_{\text{cont}} = 0, 0.1, 0.2, 0.3$, with the following method:

$$Y_{\text{contaminated}} = Y + 3 * Y_{\max}, \tag{3.9}$$

where $Y_{\max}$ is the maximum of all observed responses. $p_{\text{cont}} = 0$ indicates 0% data contamination in the training set, $p_{\text{cont}} = 0.1$ means that 10% of the training observations are contaminated, etc.. While complete absence of contamination is nearly impossible in real-life data, we include this scenario in our study to provide a basis for comparison with situations involving data contamination. Exploring different combinations of signal-to-noise ratios and contamination levels allows us to observe the varying effects when these different levels are applied.

Example 3

In the third example, the explanatory variables are generated from a ten-dimensional normal distribution with the covariance matrix being a Toeplitz matrix with a correlation of 0.7, i.e. $X \sim \mathcal{N}_{10}(0, \Sigma)$, where Σ is a Toeplitz matrix with $\rho = 0.7$. The response variable is generated using the following function, which creates a linear response:

$$Y_i = \sum_{j=1}^{10} X_{ji}^2 + \varepsilon_i, \tag{3.10}$$

where ε, the random error, is generated from an identical and independent standard normal distribution as in the previous two examples, i.e. $\varepsilon \sim \mathcal{N}(0, 1)$.

In Example 3, we generate 1000 training observations and 1000 testing observations. Similar to Example 1 and 2, we once again randomly contaminate a proportion p_{cont} of the training observations using Equation (3.9) with $p_{\text{cont}} = 0, 0.1, 0.2, 0.3$.

Both robust RF methods from Li and Martin [42] contain a hyperparameter similar to our method. Thus, we apply the method mentioned in Section 3.2.2 to obtain the optimal hyperparameter values for all three robust RF methods. The procedure for RFHuber and RFTukey follows very similar steps as described in Algorithm 2, and the package '`grf`' ([66]) is employed to perform the original RF algorithm.

		MSPE				MAPE			
m	p_{cont}	PWRF	RFHuber	RFTukey	Original RF	PWRF	RFHuber	RFTukey	Original RF
0.2	0	1.082	1.079	1.080	**1.079**	0.828	0.828	0.828	**0.827**
0.2	0.1	1.296	1.393	**1.292**	4.103	0.906	0.943	**0.904**	1.723
0.2	0.2	1.650	1.831	**1.437**	12.040	1.036	1.098	**0.954**	3.215
0.2	0.3	**6.867**	9.035	20.620	25.784	**2.313**	2.699	4.303	4.867
0.4	0	1.268	1.255	1.261	**1.252**	0.871	0.869	0.870	**0.869**
0.4	0.1	1.806	1.943	**1.787**	8.507	1.045	1.101	**1.041**	2.569
0.4	0.2	2.897	3.076	**2.313**	27.346	**1.403**	1.454	1.189	4.935
0.4	0.3	**16.256**	21.364	47.879	59.770	**3.637**	4.229	6.626	7.478
0.6	0	1.569	1.540	1.556	**1.527**	0.921	**0.919**	0.919	0.919
0.6	0.1	2.651	2.821	**2.612**	15.211	1.218	1.292	**1.217**	3.494
0.6	0.2	4.841	4.991	**3.722**	50.463	1.847	1.879	**1.476**	6.738
0.6	0.3	**29.611**	40.290	86.179	109.605	**4.931**	5.837	8.850	10.148
0.8	0	1.954	1.935	1.958	**1.889**	**0.969**	0.970	0.970	0.972
0.8	0.1	3.760	4.020	**3.760**	24.202	**1.400**	1.492	1.405	4.443
0.8	0.2	7.506	7.525	**6.404**	81.864	2.336	2.328	**1.859**	8.609
0.8	0.3	**48.156**	63.947	139.848	177.113	**6.297**	7.372	11.286	12.915

Table 3.1. Example 1: Average MSPE and Average MAPE

Once we obtain the predictions from all four methods, we compare the results and evaluate the performance using mean squared prediction error (MSPE) and mean absolute prediction error (MAPE). MSPE is defined as:

$$MSPE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i - \widehat{y}_i)^2, \tag{3.11}$$

and MAPE is defined as:

$$MSPE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} |y_i - \widehat{y}_i|, \tag{3.12}$$

where n_{test} is the testing set size and $\widehat{y}$ is the predicted response. Lower MSPE and MAPE indicate better prediction performance. To avoid bias, each condition in the three examples is repeated 50 times, and the average MSPE and MAPE over the 50 repetitions are calculated for each simulation.

Results

Table 3.1 summarizes the mean MSPE and the mean MAPE of the four methods over the 50 simulations in Example 1. The MSPE change of the four methods in Example 1 is presented in Figure 3.1. Table 3.2 summarizes the mean MSPE and the mean MAPE of the four methods over the 50 simulations in Example 2, and the MSPE change and comparison of the four methods is demonstrated in Figure 3.2. In the MSPE and MAPE tables for both Example 1 and Example 2, the smallest mean MSPE and smallest mean MAPE for each scenario is in bold. The best methods among the four by comparing MSPE and MAPE in these two examples agree most of the time, although in some cases, they could give different judgements. When there is

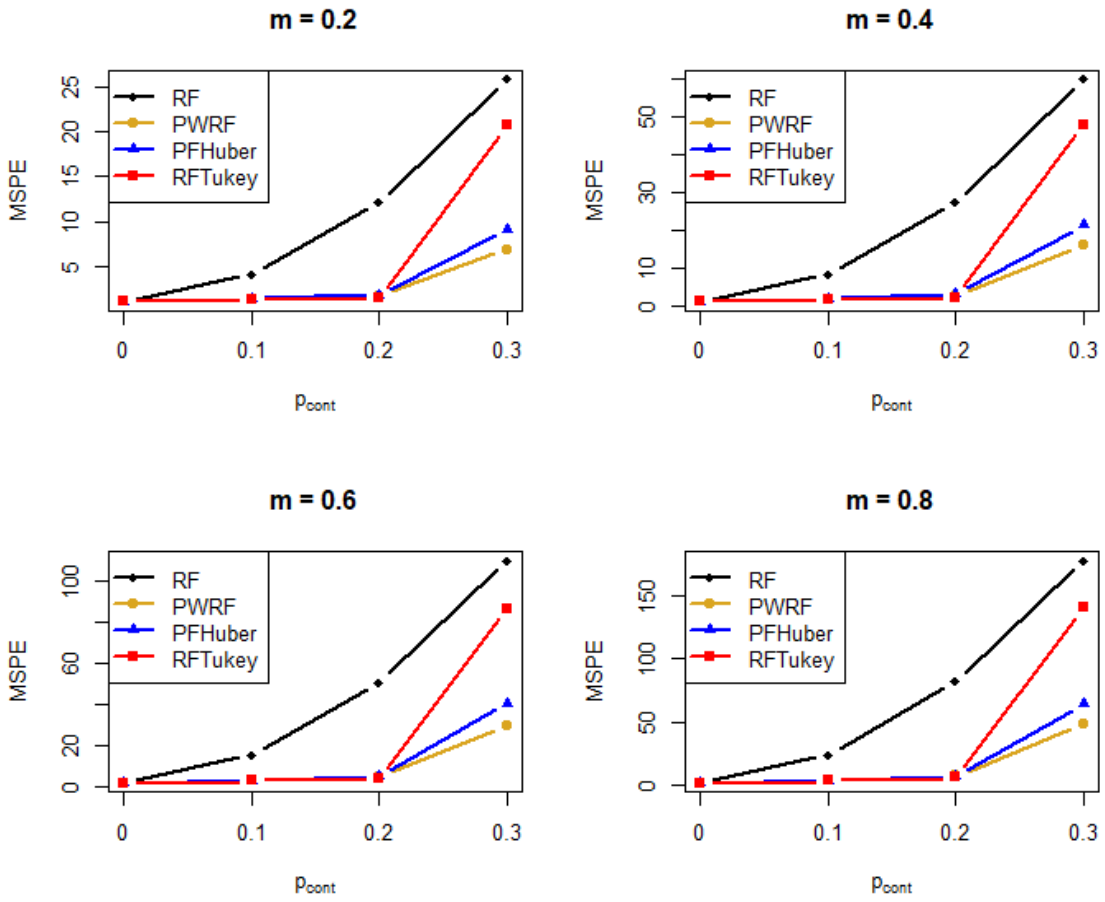


Figure 3.1. Averaged MSPE of the original RF, PWRF, RFHuber and RFTukey in Example 1

no contamination in the training set in both examples, the original RF performs the best in all cases on MSPE but most cases on MAPE. The MSPE and MAPE of the robust methods are extremely comparable with the original RF in the cases where the original RF performs the best. As the contamination proportion increases, the mean MSPE and mean MAPE of all methods begin to increase, and the performance of the original RF worsens exponentially. In contrast, gradually performs better than the other methods. PWRF exhibits the best performance in terms of MSPE and MAPE

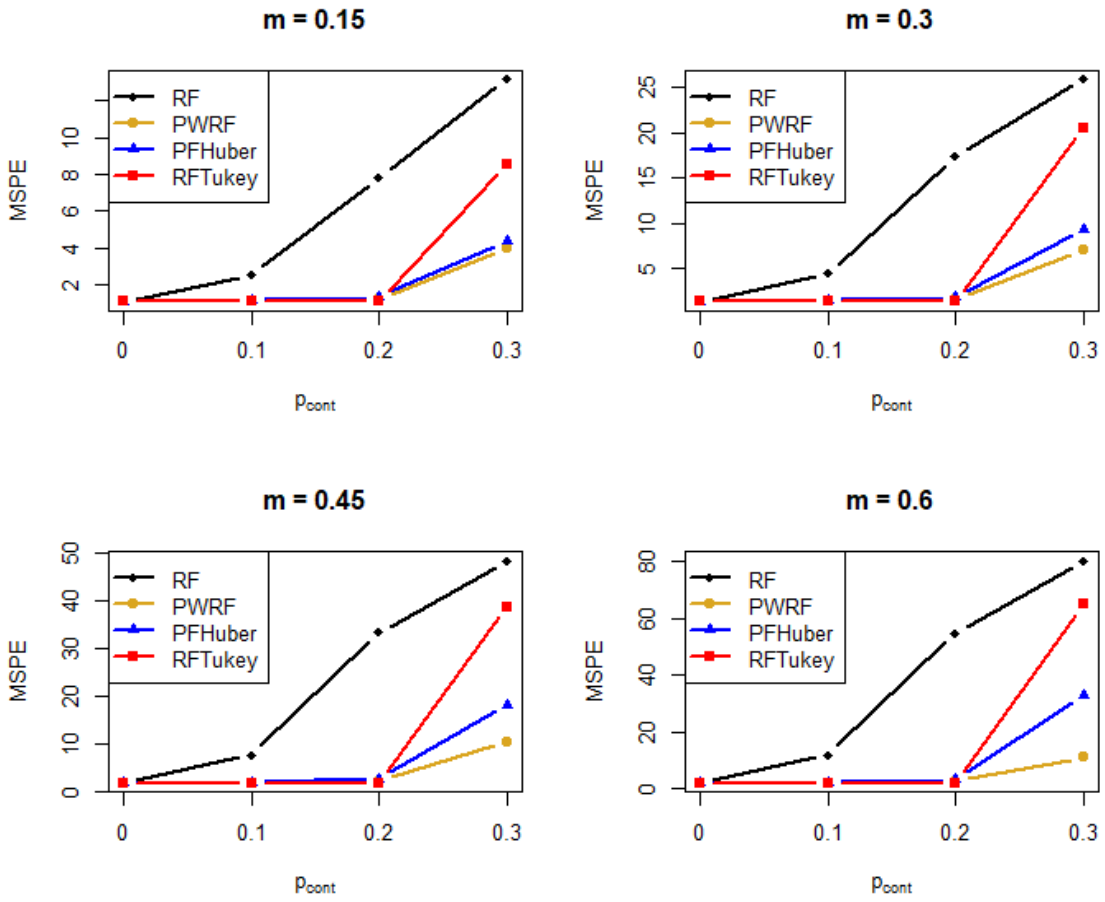


Figure 3.2. Averaged MSPE of the original RF, PWRF, RFHuber and RFTukey in Example 2

when there is high data contamination ($p_{\text{cont}} = 0.3$) in the training set. Additionally, there are occasional cases where PWRF performs the best on MAPE when there is zero contamination and moderate contamination (i.e. $p_{\text{cont}} = 0.1$ and $p_{\text{cont}} = 0.2$) in the training set. In Figure 3.1 and Figure 3.2, PWRF demonstrates significant robustness improvement compared to all other methods, while the original RF completely loses robustness as the data contamination grows.

		MSPE				MAPE			
m	p_{cont}	PWRF	RFHuber	RFTukey	Original RF	PWRF	RFHuber	RFTukey	Original RF
0.15	0	1.096	1.093	1.093	**1.092**	0.835	0.834	0.834	**0.833**
0.15	0.1	1.111	1.158	**1.111**	2.528	**0.839**	0.857	0.839	1.316
0.15	0.2	1.208	1.330	**1.095**	7.794	0.877	0.923	**0.835**	2.449
0.15	0.3	**3.947**	4.360	8.489	13.148	**1.716**	1.811	2.568	3.421
0.30	0	1.312	1.312	1.313	**1.312**	0.904	0.904	0.905	**0.904**
0.30	0.1	1.358	1.427	**1.343**	4.390	0.918	0.945	**0.913**	1.787
0.30	0.2	1.552	1.783	**1.338**	17.307	0.993	1.068	**0.914**	3.557
0.30	0.3	**7.029**	9.293	20.470	25.885	**2.350**	2.732	4.248	4.851
0.45	0	1.659	1.658	1.661	**1.654**	0.999	0.999	0.100	**0.998**
0.45	0.1	1.732	1.849	**1.717**	7.553	1.021	1.064	**1.017**	2.398
0.45	0.2	2.074	2.533	**1.734**	33.205	1.143	1.270	**1.024**	4.872
0.45	0.3	**10.162**	18.031	38.580	48.300	**2.874**	3.894	5.861	6.652
0.60	0	2.131	2.124	2.125	**2.116**	1.110	1.109	1.109	**1.107**
0.60	0.1	2.263	2.422	**2.229**	11.962	1.146	1.202	**1.138**	3.057
0.60	0.2	2.772	3.515	**2.282**	54.659	1.316	1.499	**1.153**	6.217
0.60	0.3	**11.151**	32.681	64.819	80.105	**2.971**	5.261	7.678	8.584

Table 3.2. Example 2: Average MSPE and Average MAPE

	MSPE				MAPE			
p_{cont}	PWRF	RFHuber	RFTukey	Original RF	PWRF	RFHuber	RFTukey	Original RF
0	14.606	14.636	14.538	**14.535**	2.422	2.424	2.418	**2.418**
0.1	18.982	22.549	**18.528**	346.231	2.810	3.195	**2.748**	17.494
0.2	44.509	46.100	**22.817**	1258.845	5.865	5.999	**3.055**	34.388
0.3	**22.772**	32.296	68.127	2738.169	**3.160**	4.523	4.159	51.281

Table 3.3. Example 3 Average MSPE and Average MAPE

The median and the standard deviation of the 50 MSPE and MAPE of both examples are also calculated and exhibited in Supplementary Material. The smallest median MSPE, median MAPE, MSPE standard deviation, MAPE standard deviation are in bold in the tables. The best method, when comparing the median MSPE and median MAPE, aligns with that when comparing the averaged MSPE and averaged MSPE. PWRF obstains comparable MSPE and MAPE standard deviation with the other methods. All methods perform similarly to each other when there is no contamination in the training set. The original RF obtains slightly lower MSPE and MAPE, and the variation in MSPE and MAPE among the 50 simulations are very similar as well. However, once the training set is contaminated, the prediction performance of the original RF cannot compete with the robust methods. As signal becomes stronger in both the tree-like response variable and non-linear response variable, PWRF secures better performance than the other methods with less variation in MSPE and MAPE. The mean and median of MAPE and MSPE of original RF deteriorate significantly compared to the robust methods, and the range of MAPE and MSPE of the original RF is considerably wider than that of the robust methods. When there is ample contamination in the training set, all other methods except PWRF are unable to maintain a stable level of performance, leading to an expansion in their range, and the mean and median of MAPE and MSPE of PWRF are both lower than the other methods. An interesting observation from Example 1 and Example 2 is that as the signal-to-noise ratio increases, all methods worsen in prediction performance in terms of both the center and the scale.

Table 3.3 presents the mean MSPE and mean MAPE over the 50 simulations in Example 3. The result pattern follows what we observed from the other two examples as the contamination proportion increases. All methods preform comparably

in mean MSPE and in mean MAPE when no contamination presents in the training set. RFTukey obtains the least mean MAPE and mean MSPE when there is moderate contamination ($p_{\text{cont}} = 0.1$ and $p_{\text{cont}} = 0.2$) in training set; however, PWRF appears to be very competent, being the second lowest in mean MSPE and mean MAPE with little difference to RFTukey. When there is high data contamination in the training set, PWRF shows its privilege in both mean MSPE and mean MAPE. The performance of the original RF deteriorate significantly as the contamination proportion increases. Figure 3.3 presents the change of average MSPE and average MSPE of the four methods and compares the changes. The average MSPE and MAPE of the original RF worsens so much that the change in the other methods is negligible, although RFTukey seems to be more robust than PWRF and RFHuber in average MAPE when $p_{\text{cont}} = 0.2$. Additional information on median and standard deviation of the 50 MSPE and MAPE for Example 3 are also recorded in the table in Supplementary Material.

From all examples, we observe that our method consistently outperforms the original RF method when there is contamination present in the training set, and PWRF remains robust even as the contamination proportion in the training set increases. Furthermore, PWRF is competent with the other robust methods when it does not outperform the other methods. It is worth noting that PWRF is the most robust in scenarios involving high data contamination across all examples and scenarios. This demonstrates that our method consistently provides better results than the other methods across a broad variety of data settings without sacrificing precision.

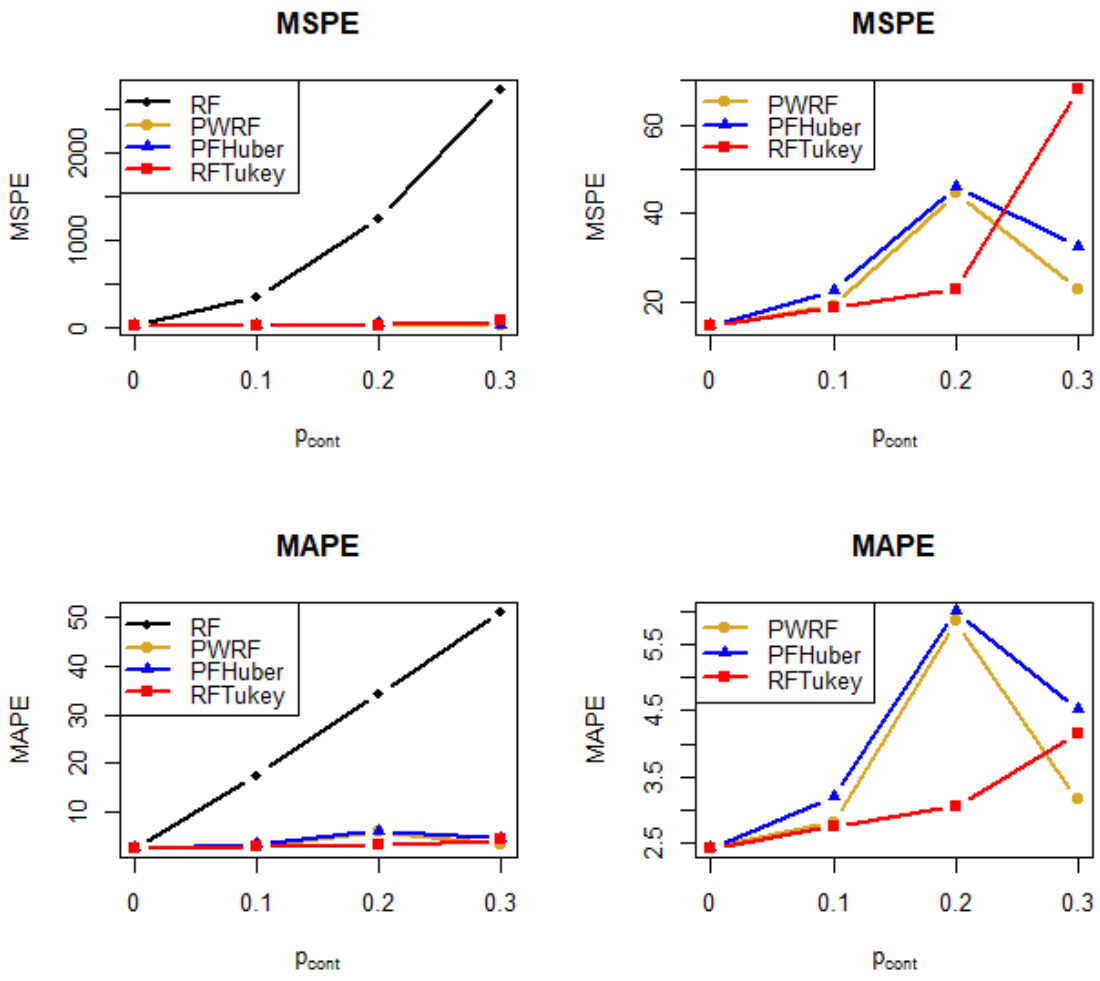


Figure 3.3. Averaged MSPE of the original RF, PWRF, RFHuber and RFTukey in Example 3

3.4 Real data analysis

The result from the simulations studies provides convincing evidence that our method is highly competitive across different contamination levels. It would yield more meaningful results when applying the methods to real-life data, as we may not know if there is contamination in the dataset, what kind of contamination exists, or the level of contamination present. To assess the effectiveness of our method in practical scenarios, we apply PWRF to a financial dataset containing stock exchange records. We then compare its performance to the same methods used in the simulation studies outlined in Section 3.3: RFHuber, RFTukey, and the original RF. In the paper by Gu et al. [31], a similar data set was used. They compared the performance of several machine learning models to the baseline ordinary linear regression model to predict equity risk premium. These machine learning models include, but not limit to, generalized linear model, penalized regression models (eg. elastic-net), dimension reduction methods (eg. principal component regression), tree based models (eg. RF), and layered models (eg. neural network). Our goal in the data analysis is also to predict the future stock return using economic and financial indicators. These indicators include the industry momentum strategies based on Fama-French 49 industries (IND49), national saving (NS), adjusted cost base (ACB), book-to-market ratio (logbm), asset growth (AG), gpta, IKxing, net operating assets (NOA), Ohlson's O-Score (Oscore), return on assets (roaq), Volume (logvolume), stock volatility (lagsvol), beta coefficient (lagbeta), Amihud (lagamihud), collection effectiveness index (lagcei), momentums (mom2, mom2m1, lagindmom), and distress (lagdistress). The data set contains monthly total individual equity returns from CRSP for all firms listed on the NYSE, AMEX, and NASDAQ from July 2004 to December 2020. The variable 'RET,' the realized excess

	PWRF	RFHuber	RFTukey	Original RF
Mean - MAPE	**0.095**	0.096	0.096	0.097
Standard Deviation	**0.089**	0.089	0.089	0.090
Median	**0.071**	0.072	0.071	0.073

Table 3.4. Summary of real data analysis results of absolute prediction error of Penalized Weighted Random Forest (PWRF), Random Forest with Huber loss function, Random Forest with Tukey loss function, and the Original Random Forest.

stock return, is the response variable in the data analysis.

The variable 'idx' is the month index, starting from 31 and ending with 228, with an increment of 1, these month indices correspond to a span of 198 months. The entire data set contains $44,959$ records of monthly stock information in the 198-month span. For our analysis, we select data from January 2017 to December 2018 as the training data set, data from January 2019 to December 2019 as the validation set, and data from January 2020 to December 2020 as the testing data set. We include all 19 economic indicators as the covariates in the analysis. As one of the assumptions of an RF model is that all variables are from the same distribution, we standardize all the covariates using their mean and standard deviation from the training set. To ensure optimal performance, we tune the hyperparameters for all robust methods, including PWRF, RFHuber, and RFTukey. However, our data set contains time-series financial data set, performing cross-validation as discussed in Section 3.2.2 would be impractical. This is because cross-validation randomizes the observations, and we may inadvertently use future data to predict past data during this process. Alternatively, we use the validation set that contains data in 2019 to tune for the hyperparameters to avoid creating time confusion. The optimal hyperparameters should minimize the validation error.

All models are evaluated on their prediction performance by comparing the predic-

	PWRF	RFHuber	RFTukey	Original RF
MSPE	**0.017**	0.017	0.017	0.017

Table 3.5. Mean squared prediction error (MSPE) of Penalized Weighted Random Forest (PWRF), Random Forest with Huber loss function, Random Forest with Tukey loss function, and the Original Random Forest.

tion errors, MAPE, and MSPE. The calculation of MSPE and MAPE is consistent with the simulation studies to compare the performance of all four methods. A summary of the absolute prediction error is presented in Table 3.4, and the MSPE is provided in Table 3.5. All three robust RF methods outperform the original RF by at least 1.06% in MAPE, and 1.33% in MSPE. The best performance comes from our proposed method. Specifically, MAPE from PWRF is 1.895% lower than that of the original RF, and 0.178% lower than that of the second best method, RFTukey. The median of the absolute prediction errors from PWRF is 2.822% better than the original RF and 0.352% better than RFTukey. The results from MSPE in 3.5 and from MAPE in 3.4 are consistent. MSPE from PWRF is 2.553% lower than that from the original RF and 0.334% lower than that from RFTukey. Our method collectively performs better than all other three methods on MAPE and MSPE by aquiring smaller variation in the prediction errors. The standard deviation of the absolute prediction errors of PWRF is the least among all methods, 0.146% smaller than RFTukey. The range of all absolute prediction errors using PWRF (0.442) is the minimum among all methods, 5.506% narrower than the range using original RF and 2.714% narrower than the RFTukey. While the smallest minimum prediction error is obtained by RFHuber (2.42×10^{-6}), PWRF acquires the smallest maximum prediction error (0.442). Both the standard deviation and range provide evidence that PWRF is more stable and more consistent.

Figure 3.4 displays a boxplot of the absolute prediction error from all four methods,

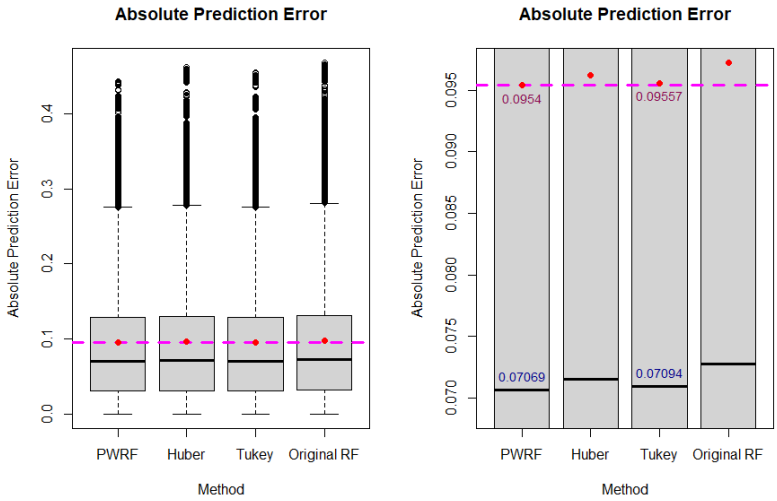


Figure 3.4. Boxplots of Absolute Prediction Error of Penalized Weighted Random Forest (PWRF), Random Forest with Huber loss function, Random Forest with Tukey loss function, and the Original Random Forest for Real Data Analysis

providing a visual representation of the median and range discussed above. The dashed lines indicate the minimum MAPE among the methods, and the red dots label the MAPE for each method. The figure on the left is the boxplot of all absolute prediction errors from the real data analysis. The difference in median among the methods is negligible, nonetheless, it is evident that PWRF exhibits the least range in absolute prediction errors. The boxplot on the right zooms into the boxplot on the left near the median and the mean. Evidently, the median absolute prediction error and MAPE of PWRF is lower than those of the original RF, but they are very close to those of RFTukey. The number in blue above the median lines are the median absolute prediction error of PWRF and that of RFTukey, with the former being lower. The number in red below the red dots are the MAPE of PWRF and that of RFTukey, with PWRF having a slightly lower MAPE. The above observations from the boxplot aligns with the numeric summary discussed earlier, providing empirical evidence that

our method is consistent and stable.

One conclusion from Gu et al. [31] is that the tree-based models, including the original RF, exhibit the most significant performance improvement over linear regression models. This is particularly true for financial data, which is highly sensitive to unforeseeable news, resulting in a low signal-to-noise ratio. Additionally, financial returns and stock-level variables are known to be heavy-tailed. The original RF proves to be robust in scenarios characterized by a low signal-to-noise ratio and heavy-tailed distributions. Its robustness prevents easy overfitting of noise over the signal, as observed in Example 1 and Example 2 in Section 3.3 where the performance of the original RF remains sustainable as the value of m decreases. Despite the inherent robustness of the original RF, the robust RF models outperform it in this real data analysis. This suggests that the robust RF methods effectively alleviate the impact of data contamination, with PWRF demonstrating the best robustness.

3.4.1 The year 2020

The year 2020 was indeed exceptional due to the global pandemic, which significantly altered the way people lived and worked. The financial landscape also experienced turmoil, potentially leading to a dataset for 2020 that follows a different distribution compared to previous years. This discrepancy could significantly affect prediction performance and render the results less representative. We can observe this potential difference in data distribution by examining the histogram of the response variable 'RET', the stock excessive return, and the scatter plot of 'RET' against all other covariates. Figure 3.5 are the histogram and scatter plots using data from 2014 to 2020, with the year 2020 separated. In the overlaying histogram on the top left corner

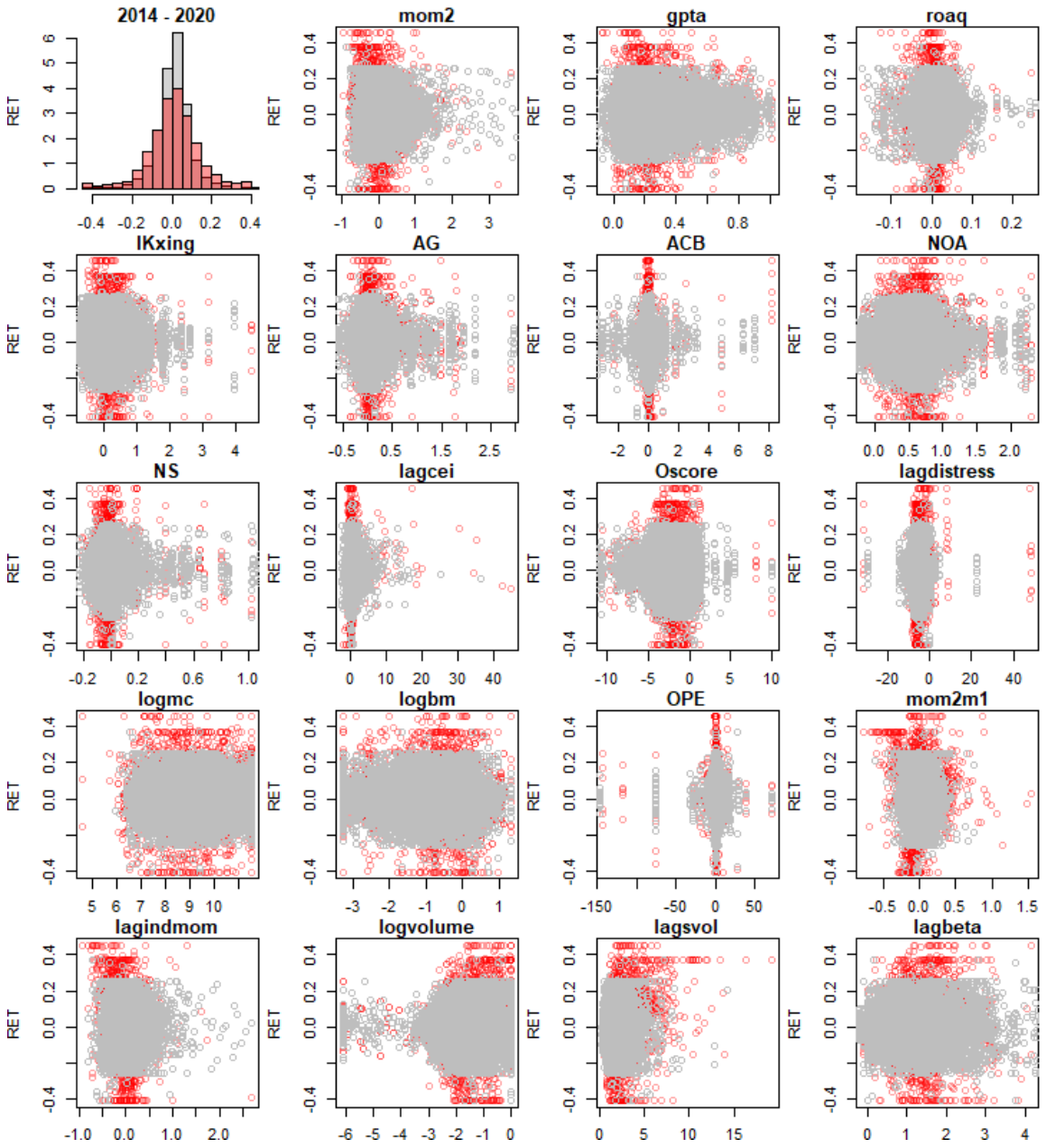


Figure 3.5. Histogram of the response variable 'RET' with scatter plot matrix of 'RET' against other variables with data between 2014 and 2019 in gray shade and data from 2020 with red shade.

of Figure 3.5, the gray histogram represents data from 2014 to 2019, while the red histogram depicts the distribution of data in 2020. Although both histograms appear to be normally distributed, it's evident that the data in 2020 exhibits heavier tails compared to the data from 2014 to 2019. In all the scatter plots, the gray data points correspond to data from 2014 to 2019, and the red shaded data points represent data from the year 2020. These plots further illustrate that the data from 2020 has a larger standard deviation, along with more outliers in both directions. Given the robustness of the original RF to heavy-tailed distributions, using the year 2020 as the testing set may not significantly impact its performance. Consequently, we may not observe substantial improvement with the robust RF methods compared to the original RF. To address this issue, we treat 2020 and the other years as two distinct cases: applying the four methods to the data before 2020 and the data during 2020 separately. This approach allows us to assess the performance of the four methods across different economic periods.

For data preceding 2020, we use information from 2014 to 2016 as the training set, data from 2017 as the validation set, and data from 2018 to 2019 as the test set. The remaining settings remain the same as previously mentioned in Section 3.4. Figure 3.6 demonstrates the boxplot of the absolute prediction errors applying the four methods, indicating comparable results from all four methods. his observation suggests that when there are only few outliers in the data set, the original RF can be quite robust, and therefore, less improvement from the robust methods.

For the data during 2020, we use data from January through July as the training set, from August to September as the validate set, and October through December as test set. The remaining settings remain the same as previously mentioned in Section 3.4. Figure 3.7 are the boxplots of absolute prediction errors applying four methods,

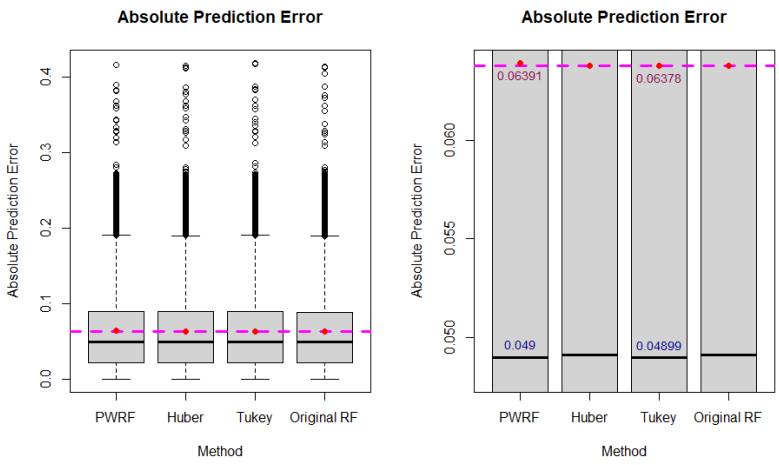


Figure 3.6. Boxplots of Absolute Prediction Error of PWRF, RFHuber, RFTukey, and the Original RF for Real Data Analysis with data before 2020 being split into three years of train set, one year of validate set, and two years of test sets

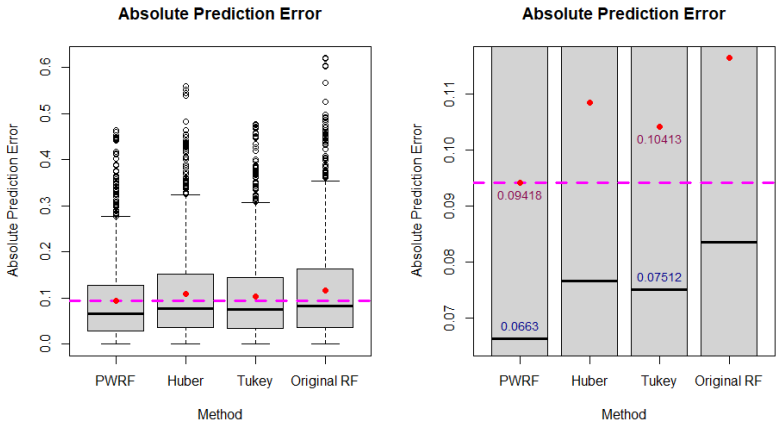


Figure 3.7. Boxplots of Absolute Prediction Error of PWRF, RFHuber, RFTukey, and the Original RF for Real Data Analysis with 2020 data split into seven months of train set, two months of validate set, and three months of test sets

demonstrating that PWRF outperforms all other methods with the lowest MAPE and lowest median absolute prediction error. Additionally, PWRF boasts the smallest standard deviation among the absolute prediction errors. This suggests that PWRF effectively mitigates the impact of abnormal behavior in the data, demonstrating robust predictive performance even in the presence of more outliers and variations in the distribution of the response variable.

3.5 Summary

PWRF inherits the flexibility and the robustness in face of heavy-tailed data from the original RF method. In addition, PWRF introduces individual weights to training observations and enforces a lasso-like penalty on the individual weights to regulate them. This new amendment in PWRF further improves the robustness of a RF model, especially when faced with mean-shift data contamination in the training dataset. Demonstrated in three different simulation study examples, each synthetically contaminated by proportions of 10%, 20%, and 30%, PWRF consistently outperforms the original RF method. Furthermore, PWRF exhibits competence comparable to other two robust RF methods (the RFTukey and the RFHuber). Moreover, PWRF consistently achieves comparable results with the original RF method when there is no contamination present in the training dataset in these examples. In the analysis of stock return data, PWRF consistently outperforms all other RF methods, demonstrating superiority by at least 0.334% in MSPE and at least 0.178% in MAPE, without treating the year 2020 as a special case. Excluding the year 2020, PWRF achieves predictions that are comparable to other methods. It is noteworthy that, when using only the 2020 data, the proposed PWRF method significantly outperforms all other

methods. This empirical evidence underscores the robustness of PWRF when dealing with unexpectedly abnormal data, a phenomenon that is inherently non-predictable.

We believe that PWRF holds broad applicability across various fields, including finance, e-commerce, healthcare, and more. Our simulation studies demonstrate the remarkable stability of PWRF when applied to different types of data characterized by non-linearity, low signal-to-noise ratio, or contamination. In future investigations, it would be worthwhile to explore PWRF's capability to label contaminated observations and examine how it can effectively cooperate with contamination labeling while identifying variable importance. Our method integrates the penalized weighted approach with RF to enhance RF's robustness in the presence of mean-shift type data contamination. The penalized weighted method has proven effective in improving model robustness to data contamination across various algorithms. It holds great potential for pairing with several other methods to address issues related to data variability.

Chapter 4: Conclusion and Future Work

In the big data era, the speed of data generation is astonishingly rapid. The sheer volume of data being generated is colossal, encompassing a diverse array of types and formats. With this abundance of data, transparency is heightened, allowing us to model future events and predict outcomes for unseen data. In this competitive landscape, scholars and practitioners vie with each other to create algorithms that accurately perform prediction and estimation data, accommodating various formats and types. Stochastic Gradient Descent (SGD) emerges as a valuable tool for scenarios involving data streaming or large datasets that surpass the limits of traditional methods due to assumption constraints. SGD excels in handling high-dimensional datasets, as it imposes no limitations on dimensionality. By processing one or a small batch of data at a time and updating iteratively, SGD proves effective in reducing computational costs. A simple application of SGD is on the linear regression model, where it updates parameter estimates using one or a mini-batch of data in each iteration. Random Forest (RF) stands out as a versatile tool for predictions in both classification and regression tasks. At its core, RF leverages the synergistic effect of numerous decision trees with bootstrapped training samples and randomly selected covariates. The use

of bootstrapped training samples and the random selection of covariates in each tree contribute to RF's robustness, particularly in handling heavy-tailed data.

The downside of living in this era is the diminishing assurance of data quality. While data contamination has existed since the inception of data, the big data era is likely to perpetuate this issue unless measures are taken to mitigate its growth. Statistical and machine learning methods are susceptible to the impact of data contamination due to the assumptions inherent in each method. The estimation or prediction of the mean or expectation can be severely affected when data contamination is present, as it violates these assumptions. SGD for linear regression is particularly sensitive to data contamination, given the small amount of data used to update model parameters in each iteration. RF may experience a reduction in robustness if the training observations include outlying data points with covariate vectors similar to those of uncontaminated training observations, while the testing data remains uncontaminated. Despite advanced data wrangling methods aimed at cleaning data before applying existing methods, keeping pace with the speed at which data is generated necessitates the ability to account for data contamination during method execution.

This dissertation discusses the application of the penalized weighted method to two machine learning methods, SGD and RF. The penalized weighted method aims to mitigate the impact of outliers or data contamination on the SGD estimates for model parameters and RF predictions. In Chapter 2, we propose the PWSGD method. This method assigns an individual weight to each training observation, which is calculated based on the residual of the training observation estimations using the model parameters estimated from the previous iteration. The value of this individual weight indicates the likelihood of the observation being an outlier, with smaller values indicating a higher likelihood. Consequently, observations with smaller individual

weights contribute less information to the estimation of model parameters. To control the assignment of individual weights, a Lasso-like penalty is imposed on the individual terms, with a tuning parameter governing the strength of the penalty. We demonstrate the superiority of PWSGD over SGD in terms of model parameter estimation and response estimation using both synthetic and real data examples. Simulation studies on synthetic data, considering both small and large data settings, reveal that as data contamination increases, SGD's performance deteriorates, while PWSGD remains robust to data contamination. Furthermore, real data analysis using the gas sensor dataset [23] highlights the superiority of PWSGD over SGD in utilizing all sensor readings to estimate the last sensor reading.

In Chapter 3, we present PWRF to overcome the limitations of RF when facing data contamination in the training set in a regression task. An individual weight is assigned to each training observation based on residual of the leave-one-out prediction of that observation. The individual weight is smaller for the observations that are potential outliers. PWRF demonstrates comparable results with mild data contamination and exhibits strong robustness with higher data contamination in the simulated examples. Furthermore, it performs comparable to the real data analysis on the financial data set. An interesting observation from the real data analysis is that PWRF outperforms all other methods, including the original RF method and two other robust RF methods when the data appears to be irregular. However, it remains comparable to other methods when the data tends to be "clean".

For future projects, I will focus on proving the stability of the outlier detection properties for both methods. Additionally, there is potential to extend the PWSGD simulation examples by incorporating heterogeneous models. Moreover, we plan to evaluate how PWRF performs when categorical explanatory variables are introduced,

requiring adjustments to the ‘grf‘ library or code modifications to accommodate categorical features. Investigating the application of the two proposed methods to high-dimensional data analysis is also on the agenda. Looking ahead, we aim to extend the application of the penalized weighted method to other machine learning techniques that are sensitive to data contamination.

References

[1] Francis R. Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n). *CoRR*, abs/1306.2119, 2013.

[2] J.A. Bather. *Stochastic Approximation: A Generalisation of the Robbins-Monro Procedure*, volume 89. 1989.

[3] Albert Benveniste, Michel Métivier, and Pierre Priouret. Adaptive algorithms and stochastic approximations. In *Applied Mathematics*, 1990.

[4] Dimitri P. Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163–195, 2011.

[5] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, pages 161–168, 2008.

[7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. Classification and regression trees. *Biometrics*, 40:874, 1984.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[10] Raymond J. Carroll and David Ruppert. *Transformation and Weighting in Regression*. Chapman and Hall, London, 1981.

[11] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997.

[12] Li Cheng, S. V. N. Vishwanathan, Dale Schuurmans, Shaojun Wang, and Terry Caelli. implicit online learning with kernels. In *Neural Information Processing Systems*, 2006.

[13] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[14] R. Dennis Cook. Detection of influential observations in linear regression. *Technometrics*, 19(1):15–18, 1977.

[15] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(99):2899–2934, 2009.

[16] Alexandre Défossez and Francis Bach. Averaged least mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 205–213, 2015.

[17] Moonjung Eo, Jeongyun Han, and Wonjong Rhee. Deep learning framework with essential pre-processing techniques for improving mixed-gas concentration prediction. *IEEE Access*, 11:25467–25479, 2023.

[18] Yixin Fang. Scalable statistical inference for averaged implicit stochastic gradient descent. *Scandinavian Journal of Statistics*, 46(4):987–1002.

[19] Yixin Fang, Jinfeng Xu, and Lei Yang. On scalable inference with stochastic gradient descent, 2017.

[20] Yixin Fang and Lincheng Zhao. Approximation to the distribution of lad estimators for censored regression by random weighting method. *Journal of Statistical Planning and Inference*, 136(4):1302–1316, 2006.

[21] Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

[22] Jordi Fonollosa. Gas sensor array under dynamic gas mixtures. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5WP4C.

[23] Jordi Fonollosa, Sadique Sheik, Ramón Huerta, and Santiago Marco. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, 2015.

[24] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[25] Francis Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[26] X. Gao and Y. Fang. Penalized weighted least squares for outlier detection and robust regression. *Journal of Business Statistics and Economics*, 2016. https://arxiv.org/abs/1603.07427.

[27] Xiaoli Gao. Penalized weighted low-rank approximation for robust recovery of recurrent copy number variations. *BMC Bioinformatics*, 16:407, 2015.

[28] Xiaoli Gao and Yang Feng. Penalized weighted least absolute deviation regression. *Statistics and Its Interface*, 11:79–89, 2018.

[29] Carl Friedrich Gauss. *Theory of Motion of the Heavenly Bodies Moving about the Sun in Conic Sections*. Perthes et Besser, Hamburg, 1809.

[30] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[31] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273, 02 2020.

[32] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. New York: Willey, 1986.

[33] Douglas M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[34] Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, Oct 2021.

[35] Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:492–518, 1964.

[36] Peter J. Huber. *Robust Statistics*, volume 523. John Wiley & Sons, 2004.

[37] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[38] Jyrki Kivinen, Manfred K. Warmuth, and Babak Hassibi. The p-norm generalization of the lms algorithm for adaptive filtering. *IEEE Transactions on Signal Processing*, 54(5):1782–1793, 2006.

[39] Roger Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

[40] Brian Kulis and Peter L. Bartlett. Implicit online learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 575–582, 2010.

[41] Adrien-Marie Legendre. Nouvelles méthodes pour la détermination des orbites des comètes. *Mémoires de l'Académie Royale des Sciences*, Volume Number:Page Range, 1805.

[42] Alexander Hanbo Li and Andrew Martin. Forest-type regression with general losses and robust forest. In *International conference on machine learning*, pages 2091–2100. PMLR, 2017.

[43] Andy Liaw and Matthew Wiener. Classification and regression by random forest. *R News*, 2:18–22, 2002.

[44] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, pages 578–590, 2006.

[45] Bin Luo, Xiaoli Gao, and Susan Halabi. Penalized weighted proportional hazards model for robust variable selection and outlier detection. *Statistics in Medicine*, 2022.

[46] Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of machine learning research*, 7(6), 2006.

[47] Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[48] Jin-Ichi Nagumo and Atsuhiko Noda. A learning method for system identification. *IEEE Transactions on Automatic Control*, 12(3):282–287, 1967.

[49] Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm, 2015.

[50] Neal Parikh and Stephen P. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1:127–239, 2013.

[51] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[52] Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[53] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[54] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. Convergence of stochastic proximal gradient algorithm, 2014.

[55] Peter J. Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *WIREs Data Mining and Knowledge Discovery*, 1(1):73–79, 2011.

[56] Marie-Hélène Roy and Denis Larocque. Robustness of random forests for regression. *Journal of Nonparametric Statistics*, 24(4):993–1006, 2012.

[57] Donald B. Rubin. The bayesian bootstrap. *Annals of Statistics*, 9:130–134, 1981.

[58] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. 1988.

[59] Andrew Sage. Random forest robustness, variable importance, and tree aggregation. *Iowa State University Capstones, Theses, and Dissertations*, 2018.

[60] David J. Sakrison. Efficient recursive estimation; application to estimating the parameters of a covariance function. *International Journal of Engineering Science*, 3(4):461–483, 1965.

[61] Vatsal Shah, Xiaoxia Wu, and Sujay Sanghavi. Choosing the sample with lowest loss makes sgd robust, 2020.

[62] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *CoRR*, abs/1212.1824, 2012.

[63] Jun Shao and Dongsheng Tu. *The jackknife and bootstrap*. Springer Science & Business Media, 2012.

[64] Y. She and A.B. Owen. Outlier detection using nonconvex penalize regression. *JASA*, 106(494):626–639, 2011.

[65] Wei Sun, Junhui Wang, and Yixin Fang. Consistent selection of tuning parameters via variable selection stability. *The Journal of Machine Learning Research*, 14(1):3419–3440, 2013.

[66] Julie Tibshirani, Susan Athey, Erik Sverdrup, and Stefan Wager. *grf: Generalized Random Forests*, 2022. R package version 2.2.0.

[67] Panos Toulis and Edoardo M. Airoldi. Implicit stochastic approximation. *arXiv preprint arXiv:1510.00967*, 2015.

[68] Panos Toulis, Jason Rennie, and Edoardo M. Airoldi. Statistical analysis of stochastic gradient methods for generalized linear models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–667–II–675. JMLR.org, 2014.

[69] Panos Toulis, Dustin Tran, and Edoardo M. Airoldi. Towards stability and optimality in stochastic gradient descent, 2015.

[70] Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 17(3):449–462, 1975.

[71] Chun Wang, Ming-Hui Chen, Elizabeth Schifano, Jing Wu, and Jun Yan. Statistical methods and computing for big data. *Statistics and its interface*, 9(4):399, 2016.

[72] HaiYing Wang, Min Yang, and John Stufken. Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association*, 114(525):393–405, June 2018.

[73] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction, 2014.

[74] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR*, abs/1107.2490, 2011.

[75] George Udny Yule. On the theory of correlation. *Journal of the Royal Statistical Society*, 60:812–854, 1897.

[76] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 116, New York, NY, USA, 2004. Association for Computing Machinery.

[77] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling, 2015.