Robust Estimation, Regression and Ranking with Applications in Portfolio Optimization

by

Tri-Dung Nguyen

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Sloan School of Management
May 14, 2009

Certified by .....
Andrew Lo
Harris & Harris Group Professor
Director, MIT Laboratory for Financial Engineering (LFE)
Thesis Supervisor

Accepted by .....
Dimitris Bertsimas
Boeing Professor of Operations Research
Co-director, Operations Research Center

Robust Estimation, Regression and Ranking with Applications in Portfolio Optimization

by

Tri-Dung Nguyen

Submitted to the Sloan School of Management
on May 14, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Classical methods of maximum likelihood and least squares rely a great deal on the correctness of the model assumptions. Since these assumptions are only approximations of reality, many robust statistical methods have been developed to produce estimators that are robust against the deviation from the model assumptions. Unfortunately, these techniques have very high computational complexity that prevents their application to large scale problems. We present computationally efficient methods for robust mean-covariance estimation and robust linear regression using special mathematical programming models and semi-definite programming (SDP).

In the robust covariance estimation problem, we design an optimization model with a loss function on the weighted Mahalanobis distances and show that the problem is equivalent to a system of equations and can be solved using the Newton-Raphson method. The problem can also be transformed into an SDP problem from which we can flexibly incorporate prior beliefs into the estimators without much increase in the computational complexity.

The robust regression problem is often formulated as the least trimmed squares (LTS) regression problem where we want to find the best subset of observations with the smallest sum of squared residuals. We show the LTS problem is equivalent to a concave minimization problem, which is very hard to solve. We resolve this difficulty by introducing the "maximum trimmed squares" problem that finds the worst subset of observations. This problem can be transformed into an SDP problem that can be solved efficiently while still guaranteeing that we can identify outliers.

In addition, we model the robust ranking problem as a mixed integer minimax problem where the ranking is in a discrete uncertainty set. We use mixed integer programming methods, specifically column generation and network flows, to solve the robust ranking problem.

To illustrate the power of these robust methods, we apply them to the mean-variance portfolio optimization problem in order to incorporate estimation errors into the model.

Thesis Supervisor: Andrew Lo
Title: Harris & Harris Group Professor
Director, MIT Laboratory for Financial Engineering (LFE)

Acknowledgments

First of all, I would like to thank Professor Andrew Lo for his continued advice and support. I am very fortunate to work with Professor Lo and I am thankful for his flexibility in letting me choose the topic of research in portfolio optimization that I like most and for introducing to me the world of financial engineering and hedge fund strategies. The ideas presented in this thesis and in future publications are from or are strongly influenced by the discussions I have had with Professor Lo. I enjoyed very much learning his philosophy of doing research, his ways of communicating ideas, and his teaching methodology and these experiences surely influence my career. I would also like to thank Professor Lo for his generous funding support for my PhD program.

I would like to thank Professors Roy Welsch and Dimitris Bertsimas for being in my thesis committee. I am grateful to Professor Welsch for introducing to me the exciting problems in robust statistical analysis. I also really appreciate the teaching opportunities that Professor Welsch has given me. I am thankful to Professor Welsch for his contributions to the third chapter of this thesis. I would like to thank Professor Bertsimas for asking tough questions and for giving valuable comments when I first showed him the proposal and for his insights into the problems I faced. This help makes significant changes in the contents and the quality of this thesis.

I would also like to express my deepest gratitude to Professor Gilbert Strang for his guidance and encouragement. Professor Strang's dedication to research and teaching has influenced my choice to follow an academic career. I was very lucky to meet Professor Strang and I will never forget about our first research collaboration during my three week visit to MIT and the subsequent conversations.

I would like to thank Professor Georgia Perakis for her guidance and support during my first years at MIT. I am thankful for her patience in our robust competitive pricing paper. I also want to thank Professor Vivek Farias for the advice in the job search process and in communicating ideas in general.

I would like to thank the ORC staff: Andrew Carvalho, Paulette P. Mosley and Laura A. Rose and the LFE staff: Sara Salem and Svetlana Sussman for their great

help.

I would like to thank Vinh as one of my closest friends. Vinh has been with me since we were in the same undergraduate program. He is a reliable friend, a very nice roommate and a very professional colleague. I am grateful to have such a great friend like him. I am also very thankful to have many great friends at ORC like Ilan, Kelly, Premal, Rajiv, David, Kostas, Ruben, Dan and many others. I would like to thank Rajiv for his great help in adding many valuable comments and in editing the third chapter of this thesis and I would like to thank David for his advice in my job search. I am also thankful for the friendship with my Vietnamese groups at MIT, in Boston and in the VEFFA. I would also like to specially thank VEF for the generous fellowship during my first years at MIT.

Words cannot express my gratitude to my family and I would like to dedicate this thesis to them. My Father has always been with me in all my journeys. His encouragement, guidance and support are always the infinite sources of inspiration for my career. My Father has kept searching for better education for me and I am sure he is very happy with my fruitful PhD thesis. I would like to thank my Mother, my brother and sister for their love and continued support. Finally, I would like to thank my wife for her unconditional love, her caring and her understanding. Thank you for your patience and for being with me during my toughest time. I specially thank my daughter, Jennifer, for being so cheerful and inspirational. All my stresses in work disappear when seeing Jennifer smiling or laughing.

Contents

1 Introduction **13**

1.1 Motivations for Robust Estimation and Robust Regression 13

1.2 Motivations for Robust Estimation, Regression and Ranking in Portfolio Optimization 15

1.3 Contributions 19

1.4 Thesis Structure 20

2 Robust Estimation **21**

2.1 Introduction 21

2.1.1 The Outlier Detection and Robust Estimation Problems . . . 21

2.1.2 Literature Review 23

2.1.3 Model Setting 26

2.1.4 The Iterative Reweighted MLE Method 27

2.1.5 Our Approach and the Chapter Structure 28

2.2 Minimizing Weighted Mahalanobis Distances 30

2.2.1 Optimization Model 30

2.2.2 Optimal Solution Properties 32

2.2.3 Numerical Computation 36

2.3 Numerical Results 38

2.3.1 Data Generating Processes 38

2.3.2 Outlier Detection Examples 39

2.3.3 Outlier Detection Accuracy and Computational Time 39

2.4 Incorporating Prior Information via Semi-definite Programming Reformulation . . . 44
2.4.1 Semi-definite Programming Reformulation . . . 46
2.4.2 Existence and Uniqueness of the Newton Raphson Solution and the SDP Solution . . . 48
2.4.3 Incorporating Prior Information Example . . . 50
2.5 Conclusion . . . 51

3 Robust Regression 53
3.1 Introduction . . . 53
3.1.1 The Least Squares Regression Problem . . . 53
3.1.2 Outliers and the Need for Robust Estimation . . . 54
3.1.3 Literature Review . . . 54
3.1.4 Contributions . . . 55
3.1.5 Chapter Structure . . . 56
3.2 The Least Trimmed Squares (LTS) Regression Problem . . . 56
3.2.1 Formulating LTS as a Nonlinear Programming Problem . . . 57
3.2.2 LTS Reformulation . . . 58
3.2.3 An Iterative Method for LTS . . . 59
3.2.4 The Concavity of the LTS . . . 61
3.3 The Maximum Trimmed Squares (MTS) Problem . . . 62
3.3.1 MTS Reformulation as a Semi-definite Programming Problem 65
3.3.2 Performance Guarantee . . . 66
3.3.3 Robust Regression Algorithm using MTS . . . 68
3.4 Numerical Results . . . 69
3.4.1 Classical Examples . . . 69
3.4.2 Examples from Different Contamination Models . . . 71
3.4.3 Extreme Case Example . . . 73
3.4.4 Robust Regression Accuracy . . . 73
3.4.5 Computational Time . . . 74

3.5 Future Research 75
3.6 Conclusion 76

4 Robust Ranking and Portfolio Optimization 77
4.1 Introduction 77
4.1.1 Literature Review 78
4.1.2 The Robust Ranking Problem 80
4.1.3 Chapter Structure 81
4.1.4 Contributions 82
4.2 Solving the Robust Ranking Problem 82
4.2.1 The Column Generation Algorithm 82
4.2.2 Solving the Constraint Generation Problem 83
4.2.3 Solving the Relaxed Problem 85
4.2.4 Extension to Control Conservative Levels 85
4.2.5 Extension to Allow Group Ranking 87
4.3 Robust Ranking Models for Portfolio Optimization 88
4.3.1 Model I: Robust Max Weighted Ranking with Nonnegative Weight Constraint 89
4.3.2 Model II: Robust Max Weighted Ranking with Risk Constraint 90
4.4 Numerical Results 91
4.4.1 Computational Time 91
4.4.2 Empirical Study 92
4.5 Conclusion 97

5 Future Directions 100

A Appendices 102
A.1 Proof of Theorem 1 102
A.2 Proof of Theorem 5 103
A.3 Proof of Theorem 6 104
A.4 Stock Universe for Robust Ranking Empirical Tests 107

List of Figures

2-1 An example showing how a few outliers can deteriorate the maximum likelihood estimators. It shows that MLE works well if the model assumption is correct and does NOT work well if the model assumption is incorrect. 23

2-2 Model setting (Input, Output and Assumptions) for our robust estimation method. 26

2-3 Three examples demonstrating how our robust estimation technique performs under different contamination models. They show our technique can detect outliers and recover the distribution of the majority of the observations nicely. 40

2-4 An example showing how the data generated from the Fully Independent Contamination Model looks like. 41

2-5 An example illustrating the effect of incorporating prior information into the robust estimators. 52

3-1 An example demonstrating how a single outlier can grossly deteriorate the least squares estimator. 54

3-2 An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst three points. 63

3-3 An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst eight points. 64

3-4 An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst 11 points. 64

3-5 Three classical examples demonstrating the performance of our robust regression technique. 70
3-6 Two examples demonstrating the performance of our robust regression technique with data generated from the One Sided Contamination Model 72
3-7 An example to demonstrate our robust regression technique can identify outliers no mater how extremely they are located. 73

4-1 The constraint generation problem is reformulated as a network flow problem. 84
4-2 The network flow problem when the conservative level is incorporated 87
4-3 The network flow problem when allowing group ranking in the uncertainty set. 88
4-4 Computational time for solving the relaxed problem and the constraint generation problem. 93
4-5 An empirical test showing that post earning announcement does indeed have valuable information. 96
4-6 Performance comparison between robust ranking and its non-robust counterpart for Model I 98
4-7 Performance comparison between robust ranking and its non-robust counterpart for Model II 99

A-1 Figure demonstrating a geometrical intuition for the proof of Theorem 2106

List of Tables

2.1 Computation time of our robust estimation technique for different problem sizes under the Fully Independent Contamination Model . . 42
2.2 Outlier detection accuracies of our robust estimation technique for different problem sizes under the Fully Independent Contamination Model 43
2.3 Computation time of our robust estimation technique for different problem sizes under the All Direction Contamination Model 43
2.4 Outlier detection accuracies of our robust estimation technique for different problem sizes under the All Direction Contamination Model. . 44
2.5 Computation times of our robust estimation technique for different problem sizes under the First Quadrant Contamination Model. 44
2.6 Outlier detection accuracies of our robust estimation technique for different problem sizes under the First Quadrant Contamination Model. 45

3.1 An iterative algorithm for Least Trimmed Squares regression 60
3.2 Robust regression algorithm via Maximum Trimmed Squares 68
3.3 Average trimmed absolute residuals for different problem sizes under the Two Sided Contaminated Model. 75
3.4 Average computational time for different problem sizes under the fully contaminated model. 76

4.1 Computational time for running the constraint generation algorithm with different problem sizes. 92
4.2 Risk and return characteristics for different portfolios (equal weighted versus lowest SUE and highest SUE portfolios). 95

4.3 Risk and return characteristics for different strategies (robust versus non-robust). 97
4.4 Risk and return characteristics when adding additional information. . 97

A.1 List of assets in our sample. 107

Chapter 1

Introduction

1.1 Motivations for Robust Estimation and Robust Regression

The mean-covariance estimation and linear regression problems are the central parts in multivariate statistical analysis. They appear in many applications where we want to parametrize data that arrives from many different application domains such as in economics, finance, e-commerce and engineering. For example, in finance, the covariance matrix between assets' returns is estimated to model their risk.

When dealing with a large data set, we often want to find statistical and mathematical models to simplify its representation. One of the first questions that we often ask is whether we can fit the data with a normal distribution. This involves the estimation of the mean and the covariance matrix of the normal distribution. The mean and the covariance matrix are often estimated using the classical method of Maximum Likelihood Estimation (MLE) by assuming the data follows a multivariate normal distribution. It is well documented that even a single observation that deviates from the normal distribution could distort the MLE estimators (See [35, 20, 28] and our example in Subsection 2.1.1). In fact, under the normality assumption, the log likelihood function contains the sum of the squared Mahalanobis distances. If outliers are present, the squared Mahalanobis distances from them dominate those from the

good observations. Therefore, the mean and the covariance matrix are pulled toward the outliers. Since outliers appear occasionally in practice, many different techniques for outlier detection and for robust estimation have been developed recently as we will show in the literature review Subsection 2.1.2.

From the data, we also often want to study the relationship between different attributes and this is often done through regression analysis. In this case, linear regression with the method of least squares is the most common class in practice because it is very simple to compute and to explain the results. However, it is well-known that the least squares fit can be grossly influenced by outliers as we will show in Subsection 3.1.2. This issue again motivates the development of many different techniques for robust regression.

There is an extensive literature in robust estimation and robust regression as we will review in Chapters 2 and 3. In fact, the robust estimation and regression are among the most popular problems in the robust statistics community. The robust mean-covariance estimation is often viewed in the context of outlier detection since if we can detect outliers, we can use the maximum likelihood estimators on the remaining observations for the mean and the covariance matrix. On the other hand, if we can find the robust mean and the robust covariance matrix, we can classify outliers as those whose Mahalanobis distances are abnormally large. This view motivates the development of many different methods that aim to find the best subset of observations which satisfy some criteria such as having the minimum covariance determinant or the minimum volume [34, 36, 37]. However, the methods proposed suffer either from computational complexity when the problem size increases or from giving up desirable properties. In Chapter 2, we take a different approach, where we find the optimal probability of occurrence for all the observations, such that at the optimal solution, outliers are set with smaller probabilities and can be detected. Our robust mean-covariance estimators have the following properties: First, they are affine equivariant. Second, they are computationally efficient even for large problem sizes. Our method also makes it easy to incorporate prior belief into the estimators. We test the accuracy of our method for different contamination models, including

the most recently proposed ones. We found our method to be faster than the Fast-MCD method [36] for high dimensional data while still obtaining the same levels of accuracy.

The robust linear regression is often conducted via least trimmed squares, which minimizes the sum of the k smallest squared residuals. Least trimmed squares has desirable properties and forms the basis on which several recent robust methods are built, but is very computationally expensive due to its combinatorial nature. In Chapter 3, we reformulate least trimmed squares as a minimization problem of a concave function over a convex set. We then introduce the "maximum trimmed squares" problem, an "almost complementary" problem that maximizes the sum of q smallest squared residuals, in direct pursuit of the set of outliers rather than the set of clean points. Maximum trimmed squares (MTS) can be formulated as a semi-definite programming problem, which can be solved efficiently in polynomial time using interior point methods. In addition, under reasonable assumptions, the maximum trimmed squares problem is guaranteed to identify outliers no mater how extreme they are.

1.2 Motivations for Robust Estimation, Regression and Ranking in Portfolio Optimization

In mean-variance (MV) portfolio optimization, we want to allocate wealth among different assets to maximize some utility functions on the portfolio return and portfolio risk [27]. The inputs for the portfolio optimization model typically include the expected returns and the covariance matrix, which need to be estimated from historical returns and sometimes through a factor model. There have been many critiques about the poor performance of the MV portfolio optimization model due to estimation errors on the mean and the covariance matrix such as the papers by Michaud [30] and Chopra & Ziemba [15]. In order to demonstrate the mean-variance estimation challenges and the effect of estimation errors on the optimal portfolio, we will show a very simple example. Suppose today's date is 01/01/2009 and we want to form a

portfolio of two assets: Goldman Sacks stock and Microsoft stock. We also suppose the following simple utility maximization model is used:

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \boldsymbol{\omega}^t \boldsymbol{\mu} - \frac{\lambda}{2} \boldsymbol{\omega}^t \boldsymbol{\Sigma} \boldsymbol{\omega} \\ s.t. \quad & \boldsymbol{\omega}^t \boldsymbol{e} = 1 \end{aligned}$$

Here, $\boldsymbol{\omega}$ is a vector of the percentages of wealth to be allocated into the assets, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the estimated expected return and covariance matrix and $\lambda = 1$ is our choice for the risk aversion ratio.

If we look back at one year historical returns of the two assets and find their sample estimates, we would get $\boldsymbol{\mu} = \begin{bmatrix} -70.8\%, & -50\% \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 61\% & 21\% \\ 21\% & 23\% \end{bmatrix}$ and this would produce an optimal portfolio $\boldsymbol{\omega} = \begin{bmatrix} -45\%, & 145\% \end{bmatrix}$.

Now suppose we exclude one single observation on 10/09/2008, we would obtain new estimates for the expected return and the covariance matrix and a new corresponding optimal portfolio as follows: $\boldsymbol{\mu} = \begin{bmatrix} -61\%, & -47\% \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 60\% & 21\% \\ 21\% & 23\% \end{bmatrix}$ and $\boldsymbol{\omega} = \begin{bmatrix} -28\%, & 128\% \end{bmatrix}$

This simple example shows us that the estimation of $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is very sensitive to the inclusion of a only a few observations while a small change in $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ could lead to significant changes in the optimal $\boldsymbol{\omega}$. Notice that we could have more or less severely sensitive results depending on our choice of the objective function and the constraint set. However, it is clear that, in general, it is a very difficult task to estimate the input parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ while the optimal solutions are very sensitive to the changes in the inputs because optimization models often produce optimal solutions at the corners of the feasible space. In fact, many researchers including Michaud [30] and Chopra & Ziemba [15] have shown that the mean-variance optimal portfolio performs poorly out-of-sample. Since then, the problem has attracted the attentions of researchers from many disciplines and there have been many different techniques for incorporating estimation errors into the MV portfolio optimization

model. These methods range from robust optimization approaches by Goldfarb & Iyengar [18], Tütüncü & Koenig [40] ..., Bayesian approaches by Jorion [21], Black & Litterman [11], and robust estimation approaches by Welsch & Zhou [44], DeMiguel & Nogales [16]... There are many other methods such as resampling by Michaud [29], using ordering information by Almgren & Chriss [1], or using random matrix theory by Laloux et al. [24, 25], Burda et al. [13], Pafka and Kondor [31] ...We refer the readers to the complete review by Brennan et al. [12] and the references therein for detail. These methods present many nice ideas for tackling the portfolio optimization problem. However, our tests in [12] show that none of the methods work well for different data sets. The empirical data sensitivity issue shown in the aforementioned example motivates us to find the answers for the following questions of interest:

1. *First, which historical return period should we use for estimating the mean and covariance matrix? Should we use one year data, two year data or any other window lengths?*

2. *Second, how do we identify influential observations from the data? (This might not be an easy task if the number of assets is large)*

3. *Third, once we have identified the influential observations, should we include or exclude them in our estimates?*

4. *Fourth, how do we find estimators* $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *that represent the majority of the data?*

5. *Fifth, since the estimation of the expected return is too difficult, can we avoid it?*

6. *Finally and most importantly, how do we find optimal portfolios that are robust against estimation errors?*

We believe that our robust estimation methods would provide the answers for questions 2 and 4. In Chapter 4, we will present the robust ranking method that could provide the partial answers for questions 5 and 6. We have not found the

complete answers for all the above questions in this thesis. However, we believe robust estimation alone will not solve the problem since the part of data which are treated as outliers from the robust statistics view actually contains the observations that should be given more careful analysis. Instead, we believe robust estimation in combination with other techniques such as robust optimization would be very powerful. Nevertheless, our method of using mathematical programming models for the robust estimation and robust regression would make the inclusion of robust estimation into portfolio optimization models easier. We will have a discussion about future directions in Chapter 5.

We also propose another method of robust ranking for portfolio optimization. We notice that the estimation of the expected return is a very difficult task and a small error in the estimation could lead to significant change in the optimal portfolio. We can avoid the estimation of the expected return by using the assets' ranking. Using ranking is very relevant in portfolio construction because managers usually have preferences on their assets. They might prefer one stock or one sector of stocks more than others. These preferences are usually expressed in the form of ranking. In addition, ranking usually contains uncertainty and it is more common to see the ranking to be expressed in an uncertainty set rather than point estimates. For example, it is more common to predict an asset to be in the top 5, top 10 or bottom 5, bottom 10 but not to say an asset to be the best one, the 5th best or the worst one. This means the ranking is often expressed as a discrete uncertainty set. The robust ranking then becomes a mixed integer minimax problem. We will present the problem formulation and the solution approach using column generation and network flows in Chapter 4. For empirical tests, we use post announcement earning drifts to obtain the ranking uncertainty set for the stocks in the DJIA index in the period from 2000 to 2007. We show our robust ranking method produces portfolios with smaller risk and higher Sharpe ratios compared to their non-robust counterparts.

1.3 Contributions

- We present a mathematical programming model for the robust estimation problem. The complexity of our algorithm is $O(n^3)$, which is independent of the problem dimension.

- We show the mathematical programming model to be equivalent to a semi-definite programming model so that many types of additional constraints can be added without much increase in the computational complexity.

- We show the Least Trimmed Squares regression problem to be equivalent to a concave minimization problem.

- We introduce the idea of "Maximum Trimmed Squares" (MTS) to solve the robust regression problem. We show that the MTS is guaranteed to detect outliers. In addition, we show that the MTS is equivalent to a semi-definite programming model and can be solved efficiently using interior point methods.

- We develop a generic robust ranking model and use the constraint generation method to solve the general robust ranking problem.

- We show that the problem can be solved efficiently with large classes of objective functions by using a network flow model.

- We apply our robust ranking model to portfolio optimization and use post announcement earning drifts to obtain the ranking uncertainty set. We also provide computational results to demonstrate the algorithm performance and show empirical tests to demonstrate how our robust ranking method produces portfolios with smaller risk and higher Sharpe ratios compared to their non-robust counterparts..

1.4 Thesis Structure

This thesis consists of papers on robust estimation in Chapter 2, robust regression in Chapter 3 and robust ranking in Chapter 4. Each of the three papers is self-contained and can be read separately without the need for reading other papers. Nevertheless, as we have presented in Section 1.2, the development of the three papers was motivated from the same problem of how to incorporate estimation errors into the mean-variance portfolio optimization problem. Each of the chapters will have its own literature review, mathematical model and computational results. Finally, the three papers are connected back in Chapter 5 when we discuss about the future directions from our work on robust estimation, regression and ranking.

Chapter 2

Robust Estimation

2.1 Introduction

2.1.1 The Outlier Detection and Robust Estimation Problems

Outlier detection is one of the most common problems that appears in many different application domains which involve statistical data analysis, such as in health care, engineering, data mining, image processing and fraud detection. There has been a rich literature with many different methods for outlier detection. Most of these methods were evolved from the fields of computer science, data mining and robust statistics. In robust statistics, the outlier detection problem and the robust covariance estimation problem are interchangeable. If we can detect outliers, we could use the maximum likelihood estimators for the location (the mean) and the covariance matrix. On the other hand, if we can find the robust location and the covariance matrix, we could classify outliers as those whose Mahalanobis distances are abnormally large. The covariance estimation problem is a central part of multivariate analysis. It appears in many applications where we want to parametrize multivariate data. For example, in finance, the covariance matrix between assets' returns is used to model their risk.

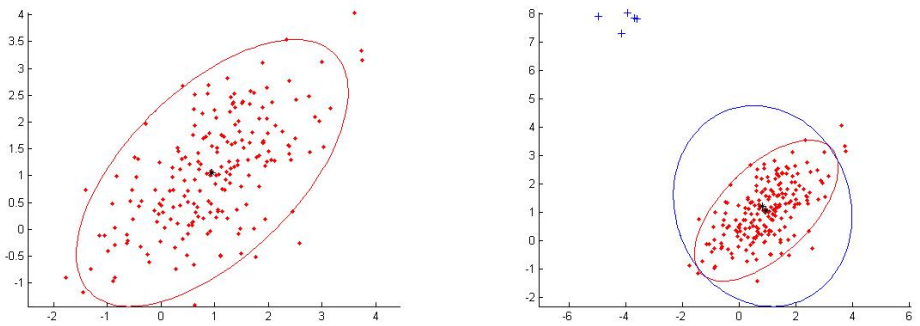
The covariance matrix is classically estimated using the Maximum Likelihood Estimator (MLE) by assuming data follows multivariate normal distribution. Let

$R \in \mathbb{R}^{m \times n}$ be a matrix of size $m \times n$ that contains n observations in n columns, each of size m. Under the normality assumption, the Maximum Likelihood Estimators for the mean and the covariance of these m attributes are:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{MLE} &= \frac{1}{n} R * \boldsymbol{e} \\ \hat{\boldsymbol{\Sigma}}_{MLE} &= \frac{1}{n}(R - \hat{\boldsymbol{\mu}}_{MLE}\boldsymbol{e}^t)(R - \hat{\boldsymbol{\mu}}_{MLE}\boldsymbol{e}^t)^t \\ &= \frac{1}{n}\sum_{i=1}^{n}(R_i - \hat{\boldsymbol{\mu}}_{MLE})(R_i - \hat{\boldsymbol{\mu}}_{MLE})^t \end{aligned}$$

where R_i is the column vectors of size m for observation i and $\boldsymbol{e}$ is a column identity vector of size n of all 1.

The above estimation is based on the assumption of normality for all the observations. It is well documented that even a single observation that deviates from the normal distribution could deteriorate the MLE estimators of the mean and the covariance matrix . In Subfigure (a) of Figure 2-1, we generate 1000 observations that follow a normal distribution. The 95% confidence ellipsoid is drawn using the MLE estimators. Subfigure (a) shows that, if the observations follow a normal distribution, the MLE estimators fit the data very well. However, if we introduce only a few new observations as shown in Subfigure (b), the new MLE ellipsoid is changed significantly and it does not represent the majority of the data very well. In fact, under the normality assumption, the log likelihood function contains the sum of the squared Mahalanobis distances. If outliers are present, the squared Mahalanobis distances from them dominate those from the good observations, which pull the location (the mean $\hat{\boldsymbol{\mu}}_{MLE}$) and the scatter (covariance matrix $\hat{\boldsymbol{\Sigma}}_{MLE}$) toward the outliers. This example shows that we need robust estimators that can represent the majority of the data better. There is a whole literature on methods to deal with outlier removal. In the next section, we will review the current approaches for outlier detection and robust covariance estimation in the field of robust statistics.



(a) When assumption is correct (b) When assumption is Incorrect

Figure 2-1: An example showing how a few outliers can deteriorate the maximum likelihood estimators. In Subfigure (a), the data follows a normal distribution and hence the MLE ellipsoid fits it very well. In Subfigure (b), we introduce only a few observations which could change the MLE from the red ellipsoid (the smaller one) to the blue ellipsoid (the larger one). The outliers pull the location and the scatter of the MLE ellipsoid toward them. This example shows that MLE works well if the model assumption is correct and does NOT work well if the model assumption is incorrect.

2.1.2 Literature Review

There is an extensive literature on outlier detection and robust covariance matrix estimation. In fact, the outlier detection problem has evolved mostly in the fields of computer science, data mining and robust statistics. For the most recent development in outlier detection techniques in general, we refer the readers to the Chandola et al. review in [14]. Our focus is on the robust statistics side, where we care more about the robust estimation of the parameters via outlier detection.

The most common approach for outlier detection is to use the classical MLE estimators on the entire data set to compute the Mahalanobis distances for all the observations and finally remove those with unexpectedly large distances (e.g. those with $d^2 > \chi^2_{0.95,m}$). However, this approach does not work well because the outliers have already created a strong bias in the MLE estimators. Thus, the Mahalanobis distances are very much different from those if we used the true parameters, i.e. the MLE parameters, which come from the good observations only.

The next stream of research on robust statistics modifies the MLE by using different functions on the Mahalanobis distances, other than the squared Mahalanobis

distances, so that the log-likelihood function is less affected by outliers. Examples of these functions include the Huber functions, Turkey's biweight functions, etc. (see [20] or [28] for more detail). This class of estimators is called the M-Estimator. The S-Estimators are those which result from the generalized maximum likelihood estimators, where the objective functions contain the scales. There are other types of estimators, such as the MM or GM estimators. The problem in using these lies in the computational complexity when the number of observations or the dimension increase.

For one dimensional data, the most common approach is to use the median as an estimator for the location and the median absolute deviation (MAD) as the estimator for the standard deviation. For higher dimensional data, the affine equivariant property is sacrificed for speed by using the pair-wise correlation estimation approaches. The most advanced one is probably the bivariate winsorization method proposed by Khan et al. in [22] which improves the one-dimensional Huber winsorization by taking into consideration the orientation of the bivariate data. Pair-wise approaches would improve the computational efficiency. However, they face the problem of arriving at a non-positive definite and non-affine equivariant covariance matrix which is undesirable. This could be serious in finance because the risk could be negative due to the the matrix being non-positive definite.

A completely different approach for robust covariance estimation is the shrinkage estimator proposed by Ledoit and Wolf in [26]. This approach basically shrinks the MLE estimator toward some known matrices, such as the identity matrix, to minimize the expected distance to the true covariance matrix. This approach is appealing for applications where the number of observations n is not significantly larger than the number of parameters to be estimated $\frac{m(m+1)}{2}$. However, this approach does not really try to remove outliers and the effects caused by outliers is still present in the shrinkage estimators.

The last and probably the most advanced robust covariance estimators are the Minimum Volume Ellipsoid (MVE) and the Minimum Covariance Determinant (MCD) proposed by Rousseeuw in [34]. The MVE approach aims to find the subset whose

covering ellipsoid has the smallest volume while the MCE approach aims to find the subset whose corresponding MLE covariance matrix has the smallest determinant. Both of thee iterative methods that move from one subset to another improving subset until a local optimal solution is found. These approaches also start at many different starting points with the hope that one of them might arrive at the global optimal solution. However, these methods face computational complexity and local optimum. The Fast-MCD by Rousseeuw and Driessen [36] is an improvement of the MCD where the observations are cleverly divided into subsets to reduce the number of iterations. However, the Fast-MCD is not really fast for high dimension problems and is still facing the local optimum.

Our approach is the result of two ideas: First, instead of viewing observations as two separate sets of good observations and outliers, we view all observations as good observations except that each of them has a different probability of occurrence. This normalized probability is proportional to the probability densities of the observations, given the true parameters. If we can find the probability of occurrence for each observation, we then can simply treat those with very small probabilities as outliers. Using this idea, we can transform the combinatory nature of the outlier detection problem into a continuous problem. Second, we observe that the robust covariance estimation problem is closely related to semi-definite programming (SDP) through linear matrix inequalities. SDP has emerged since the development of the interior point method. It provides the tools to model nonlinear constraints in one-dimensional space as linear matrix inequalities in a higher dimensional space. SDP also provides an excellent tool for relaxing combinatorial problems (see [41] and [42] for examples). In fact, we will show that the robust estimation problem can be reformulated as an SDP problem.

Our approach has the following features: First, it has all the good properties of the current best approaches, which include a) affine equivariant b) works fast for large n and c) performs well even for high dimensional data. Besides, it is easy for our method to incorporate additional belief into the estimators. It turns out that our method not only meets the above objectives but also performs slightly more accurately than the

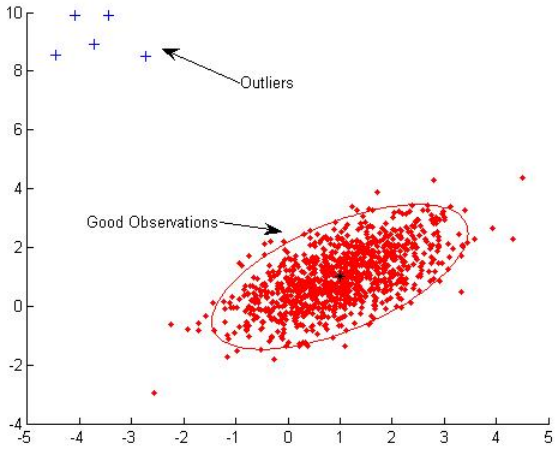


Figure 2-2: Model setting for our robust estimation method: A majority of the observations (the red dotted ones) follow a normal distribution with unknown parameters. The rest of the observations could follow any random process. The task is to find the distribution that represent the majority of the observations.

Fast-MCD for our tested contamination models.

2.1.3 Model Setting

Input: The input is any multivariate data R of size $(n \times m)$. For example, in Figure 2-2, we generate 1000 data points in two dimensional space and hence $n = 1000$ and $m = 2$.

Assumption: A majority of the observations follow a normal distribution with unknown parameters, i.e. there is a majority subset $\boldsymbol{\mathcal{S}}$ such that $R_j \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),\ \forall\, j \in \boldsymbol{\mathcal{S}}$ with $|\boldsymbol{\mathcal{S}}| = \bar{n} > \frac{n}{2}$, where $\boldsymbol{\mathcal{S}}, \bar{n}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ are unknown. For example, in Figure 2-2, the red dotted observations are those that belong to the majority while the blue crossed points are those we called the outliers (or anomalies).

Output: We want to recover the set of good observations $\boldsymbol{\mathcal{S}}$ and to estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$.

2.1.4 The Iterative Reweighted MLE Method

In the maximum likelihood estimation, we want to find the parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ that maximize the likelihood of obtaining the sample data R.

$$\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \quad \sum_{i=1}^{n} log\left[\ f\left(R_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\ \right]$$

Under the normality assumption, the closed form solutions $(\boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)$ are expressed as follows:

$$\begin{aligned} \boldsymbol{\mu}_e &= \sum_{i=1}^{n} \frac{1}{n} R_i \\ \boldsymbol{\Sigma}_e &= \sum_{i=1}^{n} \frac{1}{n} R_i^t R_i - \boldsymbol{\mu}_e^t \boldsymbol{\mu}_e \end{aligned}$$

We notice that all the observations are viewed equally important in the formulas for $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This means the MLE estimators are influenced by both good and bad observations. The weighted MLE method avoids this issue by putting weights $\boldsymbol{\omega}$ to the observations with the hope that bad observations are set with smaller weights.

$$\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \quad \sum_{i=1}^{n} \omega_i log\left[f\left(R_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\right]$$

The corresponding closed form solutions $(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ are expressed as follows:

$$\begin{aligned} \boldsymbol{\mu}_w &= \sum_{i=1}^{n} \omega_i R_i \\ \boldsymbol{\Sigma}_w &= \sum_{i=1}^{n} \omega_i R_i^t R_i - \boldsymbol{\mu}_w^t \boldsymbol{\mu}_w \end{aligned}$$

If we knew the right weight, we could use the weighted MLE method to eliminate the effects of bad observations. However, in practice we do not know the right weight since we do not know which observations are good and which ones are bad. The iterative reweighted MLE method resolves this issue by updating the weights inversely proportional to the Mahalanobis distances d_i from the observations to the weighted

mean and rescaled by the inverse weighted covariance matrix.

$$d_i = (R_i - \boldsymbol{\mu}_w)\boldsymbol{\Sigma}_w^{-1}(R_i - \boldsymbol{\mu}_w)^t$$

One popular way to reset ω_i from d_i is to notice that d_i follows a Chi squared distribution and hence we can take only observations that belong to the 95% confidence interval by setting the weights as follows:

$$\omega_i = \begin{cases} 1, & \text{if } d_i \leq \chi^2_{m,0.95} \\ 0, & \text{otherwise} \end{cases}$$

The problem with the iterative reweighted MLE method is its slow convergence and its trap into local optimal solutions. In the next Subsection, we will show our approach of using a mathematical programming model to resolve the issues faced by the iterative reweigted MLE method.

2.1.5 Our Approach and the Chapter Structure

The outlier detection problem is essentially a combinatorial problem where we want to find a subset of observations such that the normal distribution with parameters that come from the MLE estimators of that subset would classify all the observations in the subset as normal and the remaining observations as outliers. Suppose there are k outliers out of n observations, there would be $\binom{n}{k}$ possible subsets to be tested. This combinatorial nature of the problem makes it very challenging to solve. We avoid the combinatorial characteristics of the outlier detection problem by viewing all the observations as good ones except each of them have a different probability of occurrence. We introduce a probability measure on all the observations and formulate an optimization problem whose objective function is designed to make sure outliers are set with smaller weights at the optimal solution. This idea is similar to the idea used in weighted least squares or the iterative bivariate-winsorization approaches. Once we have introduced a probability measure x on the observations, we can find

the corresponding weighted MLE estimators:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{rob} &= \sum_{i=1}^{n} x_i R_i \\ \hat{\boldsymbol{\Sigma}}_{rob} &= \sum_{i=1}^{n} x_i (R_i - \hat{\boldsymbol{\mu}}_{rob})(R_i - \hat{\boldsymbol{\mu}}_{rob})^t\end{aligned}$$

Given these parameters, we can find out the corresponding Mahalanobis distances d_i as well as the actual probability of occurrence of the observations given the truth parameters $\left(\hat{\boldsymbol{\mu}}_{rob}, \hat{\boldsymbol{\Sigma}}_{rob}\right)$.

$$\begin{aligned}d_i &= (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_i - \hat{\boldsymbol{\mu}}_{rob}) \\ Prob(R_i) &= (2\pi)^{-m/2} \left|\hat{\boldsymbol{\Sigma}}_{rob}\right|^{-1/2} \exp(-\frac{d_i}{2})\end{aligned}$$

Ideally we would like to find the probability measure x on the observations such that $x_i = Prob(R_i)$ and the distances d_i are as small as possible. However, it is very difficult to find x that solves the system of equations due to the high non-linearity of the system and the sensitiveness of the exponentials. Instead, we just want to find an approximation on x such that x_i is inversely proportional to the corresponding Mahalanobis distances d_i. From that, we can set all observations with smallest x_i to be outliers.

We relax the exponential constraints $x_i = (2\pi)^{-m/2} \left|\hat{\boldsymbol{\Sigma}}_{rob}\right|^{-1/2} exp(\frac{-d_i}{2})$ and introduce a loss function $L(x, d)$ such that by minimizing the loss function, we naturally arrive at optimal solution $x^*,\ d^*$ where x_i is inversely proportional to d_i.

In this Chapter, we will first show our choice of the loss function $L(x, d)$. The choice of the loss function and the corresponding optimization model are presented in Subsection 2.2.1. Since the optimization model in the current form is highly nonlinear and nonconvex, we will not solve it directly. Instead, we will show some interesting properties of the optimal solution and will prove that the optimization problem is equivalent to solving a simpler system of non-linear equations in Subsection 2.2.2. We will then show the Newton-Raphson algorithm for finding out the solution of the

system of equations in Subsection 2.2.3. The Newton-Raphson algorithm turns out to perform very well where the convergence is achieved within less than 10 iterations even for very large problems. This is quite surprising because the equivalent optimization model seems to be very hard to solve. It turns out that under our choice of the loss function, we can reformulate the optimization problem as a semi-definite programming (SDP) problem which can be solved efficiently using interior point methods. We will show this SDP reformulation in Section 2.4. The SDP reformulation not only helps us explain the well-behaved property of the Newton-Raphson algorithm but also provides us the means to add prior knowledge to the robust estimation problem. We also show an example on how prior information can be incorporated and its effect on the estimator in Subsection 2.4.3. Finally, we show the performance of our robust estimator with different contamination models and compare it with the current most advanced method of the Fast-MCD in Section 2.3.

2.2 Minimizing Weighted Mahalanobis Distances

2.2.1 Optimization Model

Consider the general optimization model for the robust estimation problem in Model 1. In this optimization model, the first two constraints represent the weighted MLEs given the probability measure x. The third constraint corresponds to the equation for the squared Mahalanobis distances. The fourth constraint specifies the normalization of the weights x, while the firth constraint restricts the non-negativity of the weights. While the constraint setting is straightforward, the choice of the loss function needs to fulfill several objectives: First, the loss function should be designed such that those observations with larger distances are set with smaller weights at the optimal solution. Under this setting, the outliers are those with the smallest weights. Second, the loss function should be designed such that the total of the Mahalanobis distances is small at the optimal solution such that the log likelihood function is large. Third, the choice of the loss function should make the optimization problem computationally

tractable.

Model 1 : General Optimization Model for Robust Covariance Estimation

$$
\begin{aligned}
\min_{x,d,\hat{\boldsymbol{\mu}}_{rob},\hat{\boldsymbol{\Sigma}}_{rob}} \quad & L(x,d) \\
s.t \quad & \hat{\boldsymbol{\mu}}_{rob} = \sum_{i=1}^{n} x_i R_i \\
& \hat{\boldsymbol{\Sigma}}_{rob} = \sum_{i=1}^{n} x_i R_i R_i^t - \hat{\boldsymbol{\mu}}_{rob}\hat{\boldsymbol{\mu}}_{rob}^t \\
& d_i = (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_i - \hat{\boldsymbol{\mu}}_{rob}) \\
& x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}
$$

The loss function would be any function of the following type:

$$L(x,d) \;=\; \sum_{i=1}^{n} f(x_i, d_i),$$

where $f(x_i, d_i)$ is a monotonically nondecreasing function of x_i and d_i. The monotonicity will ensure that observations with larger d_i are set with smaller weight x_i to drive the objective function low. In the minimal sum of the squared residuals loss function, we have $L(x,d) = \sum_{i=1}^{n} d_i$. However, this loss function is non-robust because it does not distinguish between the weights for outliers and good observations. The most natural and typical loss function is $L(x,d) = \sum_{i=1}^{n} x_i d_i$. However, this loss function turns out to be equal to m, the problem dimension, and there is nothing to optimize (the proof is skipped for the purpose of clarity but it is available from the author on request). Instead, in our formulation, we will choose $L(x,d) = \sum_{i=1}^{n} x_i^2(d_i+1)$ because of the following reasons: The first and the foremost reason is that, $f(x_i, d_i)$ does satisfy the monotonically nondecreasing properties with respects to x_i and d_i. Another very important reason is that with this loss function, the problem can be cast into a SDP problem and can be solved efficiently as we will show in Subsection 4. The choice of $x_i^2(d_i+1)$ instead of $x_i^2 d_i$ is for the purpose of convenient reformulation. We have the following corresponding robust estimation formulation:

Model 2 : Our Optimization Model for Robust Covariance Estimation

$$
\begin{aligned}
\min_{x,d,\hat{\boldsymbol{\mu}}_{rob},\hat{\boldsymbol{\Sigma}}_{rob}} \quad & \sum_{i=1}^{n} x_i^2(d_i+1) \\
s.t \quad & \hat{\boldsymbol{\mu}}_{rob} = \sum_{i=1}^{n} x_i R_i \\
& \hat{\boldsymbol{\Sigma}}_{rob} = \sum_{i=1}^{n} x_i R_i R_i^t - \hat{\boldsymbol{\mu}}_{rob}\hat{\boldsymbol{\mu}}_{rob}^t \\
& d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1}(R_i - \hat{\boldsymbol{\mu}}_{rob}) \\
& x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}
$$

We notice that in the third constraint, we have changed from equality to "≥" without changing the optimal solution since the optimization problem would drive the distances small so that the equality would naturally be satisfied.

2.2.2 Optimal Solution Properties

We will rewrite the robust estimation slightly to study its optimal solution properties. Notice that the reformulation in this Subsection is only for the purpose of understanding the optimal solution. This Subsection does not study the model from the computational point of view.

Theorem 1. *Let* $v_i = \begin{bmatrix} 1 \\ R_i \end{bmatrix}$ *and* $Z = \begin{bmatrix} 1 & \sum_{i=1}^n x_i R_i^t \\ \sum_{i=1}^n x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} = \sum_{i=1}^n x_i v_i v_i^t$, *then the constraint* $d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1}(R_i - \hat{\boldsymbol{\mu}}_{rob})$ *is equivalent to* $d_i + 1 \geq v_i^t Z^{-1} v_i$.

The proof for Theorem 1 is in the Appendix. At the optimal solution, we always have $d_i + 1 = v_i^t Z^{-1} v_i$ to drive the objective function to minimum. Therefore, the problem can be rewritten as.

Theorem 2. *Let* x^* *be the optimal solution. Then,* $x_j^* > 0, \ \forall \ j$.

Proof. Let's define $h_{ij} = v_i^t Z^{-1} v_j$ and let x be a feasible solution and consider a perturbation $\omega_j = x_j + \epsilon$, $\omega_{-j} = \beta x_{-j}$, where the indexes $-j$ denote those indexes

Model 3 : Equivalent Optimization Model for Robust Covariance Estimation

$$\begin{aligned} \min_{x} \quad & f(x) = \sum_{i=1}^{n} x_i^2 v_i^t \left(\sum_{i=1}^{n} x_i v_i v_i^t \right)^{-1} v_i \\ s.t \quad & x^t \boldsymbol{e} = 1 \\ & x \geq 0 \end{aligned}$$

other than j. We set $\epsilon = (1-\beta)(1-x_j)$ such that the constraint $\boldsymbol{\omega}^t \boldsymbol{e} = 1$ is preserved. We have:

$$\begin{aligned} S &\equiv \sum_{i=1}^{n} \omega_i v_i v_i^t \\ &= (x_j + \epsilon) v_j v_j^t + \beta x_{-j} v_{-j} v_{-j}^t \\ &= \beta Z + (1-\beta) v_j v_j^t \\ \Rightarrow S^{-1} &= \frac{1}{\beta} \big(Z^{-1} - \frac{1-\beta}{\beta} \frac{Z^{-1} v_j v_j^t Z^{-1}}{1 + \frac{1-\beta}{\beta} v_j^t Z^{-1} v_j} \big) \end{aligned}$$

Thus, the perturbed Mahalanobis distances g_j, g_{-j} and perturbed objective function satisfy:

$$\begin{aligned} g_j + 1 &= v_j^t S^{-1} v_j \\ &= \frac{1}{\beta} \big(h_{jj} - \frac{1-\beta}{\beta} \frac{h_{jj}^2}{1 + \frac{1-\beta}{\beta} h_{jj}} \big) \\ &= \frac{1}{\beta} \frac{h_{jj}}{1 + \frac{1-\beta}{\beta} h_{jj}} \\ g_{-j,-j} + 1 &= v_{-j}^t S^{-1} v_{-j} \\ &= \frac{1}{\beta} \big(h_{-j,-j} - \frac{1-\beta}{\beta} \frac{h_{-j,j}^2}{1 + \frac{1-\beta}{\beta} h_{jj}} \big) \end{aligned}$$

$$
\begin{aligned}
f(\boldsymbol{\omega}) &= \omega_j^2(g_{jj}+1)+\omega_{-j}^2(g_{-j,-j}+1) \\
&= \big((1-\beta)^2+\beta^2x_j^2+2\beta(1-\beta)x_j\big)\frac{1}{\beta}\frac{h_{jj}}{1+\frac{1-\beta}{\beta}h_{jj}} \\
&+ \beta^2x_{-j}^2\frac{1}{\beta}\big(h_{-j,-j}-\frac{1-\beta}{\beta}\frac{h_{-j,j}^2}{1+\frac{1-\beta}{\beta}h_{jj}}\big) \\
&= \beta f(x)-(1-\beta)\left(x_{-j}^2h_{-j,j}^2+\frac{\beta}{\beta+(1-\beta)h_{jj}}(h_{jj}x_j-1)^2-1\right) \\
\Rightarrow \quad & \frac{f(\boldsymbol{\omega})-f(x)}{1-\beta} = -f(x)-\left(x_{-j}^2h_{-j,j}^2+\frac{\beta}{\beta+(1-\beta)h_{jj}}(h_jx_j-1)^2-1\right)
\end{aligned}
$$

Suppose there exist $x_j^* \le 0$. We consider a perturbation on x^* and let β approach 1. We have:

$$
\begin{aligned}
\frac{f(\boldsymbol{\omega})-f(x)}{1-\beta} &= -f(x^*)-\left((x_{-j}^*)^2h_{-j,j}^2+\frac{\beta}{\beta+(1-\beta)h_{jj}}(h_{jj}x_j^*-1)^2-1\right) \\
&\le -f(x^*) \le -x^{*t}x^* \le -\frac{1}{n} < 0
\end{aligned}
$$

Therefore $f(x^*)$ is not optimal. Thus, we can remove the constraint $x \ge 0$ and the optimal solution always satisfies $x^* > 0$. □

Theorem 3. *At the optimal solution, we have:* $f(x)+\sum_{k=1}^n x_k^2h_{k,j}^2-2h_{jj}x_j=0 \ \ \forall\ j.$

Proof. With $x_j > 0$, $\boldsymbol{\omega}$ is feasible when β is in a vicinity of 1. Thus, at the optimal solution, the left hand side must approach zero when β approaches 1 , i.e.

$$
\begin{aligned}
& f(x)+x_{-j}^2h_{-j,j}^2+(h_{jj}x_j-1)^2-1=0 \\
\Leftrightarrow \quad & f(x)+\sum_{k=1}^n x_k^2h_{k,j}^2-2h_{jj}x_j=0
\end{aligned}
$$

Remark: Once we have removed the sign constraint $x \ge 0$, the result of $\sum_{k=1}^n x_k^2h_{k,j}^2-2h_{jj}x_j$ being constant for all j can be proven by setting the derivative of the La-

grangian function $L(x,\lambda) = \sum_{k=1}^{n} x_k^2 v_k^t Z^{-1} v_k - \lambda(x^t \boldsymbol{e} - 1)$ equal to zero:

$$\begin{aligned} 0 &= \frac{\partial L(x,\lambda)}{x_i} \\ &= \frac{\partial \sum_{k=1}^{n} x_k^2 h_{kk}}{\partial x_i} - \lambda \\ &= 2x_i h_i - \sum_{k=1}^{n} x_k^2 h_{ik}^2 - \lambda \quad [1] \end{aligned}$$

Notice also that we can solve either the system of n equations $2x_i h_i - \sum_{k=1}^{n} x_k^2 h_{ik}^2 - f(x) = 0,\ \forall\, j$ with n variables or the system of $(n+1)$ equations of $2x_i h_i - \sum_{k=1}^{n} x_k^2 h_{ik}^2 - \lambda = 0,\ \forall\ j$ and $x^t \boldsymbol{e} = 1$ with $(n+1)$ variables. In our numerical approach shown in Section 2.2.3, we choose to solve the former system of equations as it is smaller.

□

Theorem 4. ***Affine Equivariant Property****: Let x be an optimal solution that corresponds to data R. Consider the affine transformation $Y = A * R + b * \boldsymbol{e}^t$ where $A \in \mathbb{R}^{m \times m}$ is a non-singular matrix, $b \in \mathbb{R}^m$ is a column vector and $\boldsymbol{e}$ is the column identity vector of size n of all 1. We will prove that x is also an optimal solution for Y.*

Proof. We have the corresponding location and covariance matrix for data Y given

[1] We have used the following results:

$$\begin{aligned} \frac{\partial Z}{\partial x_i} &= v_i v_i^t \\ \frac{\partial Z^{-1}}{\partial x_i} &= -Z^{-1} v_i v_i^t Z^{-1} \\ \frac{\partial h_{kk}}{\partial x_i} &= -v_k^t Z^{-1} v_i v_i^t Z^{-1} v_k = -h_{ik}^2 \end{aligned}$$

the weight x:

$$\begin{aligned}
\bar{Y} &= \sum_{i=1}^{n} x_i(A * R_i + b) \\
&= A * \hat{\boldsymbol{\mu}}_{rob} + b \\
\boldsymbol{\Sigma}_Y &= \sum_{i=1}^{n} x_i(A * R_i + b - \bar{Y})(R_i * A + b - \bar{Y})^t \\
&= A * \hat{\boldsymbol{\Sigma}}_{rob} * A^t
\end{aligned}$$

The distances corresponding to data Y and weight x are:

$$\begin{aligned}
d'_{ij} &= (Y_i - \bar{Y})^t \boldsymbol{\Sigma}_Y^{-1} (Y_j - \bar{Y}) \\
&= (A * R_i + b - \bar{Y})^t \boldsymbol{\Sigma}_Y^{-1} (A * R_j + b - \bar{Y}) \\
&= (R_i - \hat{\boldsymbol{\mu}}_{rob})^t A^t \left(A * \hat{\boldsymbol{\Sigma}}_{rob} * A^t \right)^{-1} A (R_j - \hat{\boldsymbol{\mu}}_{rob}) \\
&= (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_j - \hat{\boldsymbol{\mu}}_{rob}) = d_{ij}, \quad \forall\ i, j
\end{aligned}$$

From here we can show $h'_{ij} = h_{ij}, \quad \forall\ i, j$ (the proof detail is skipped). Thus, the system of equations $f(x) + \sum_{k=1}^{n} x_k^2 h'^2_{k,j} - 2h'_{jj} x_j = 0, \quad \forall\ j$ still satisfied. Hence, x is also an optimal solution for the data Y. Another way to prove is as follows: Since $d'_j = d_j$, the optimal value of the optimization problem corresponding to data R is equal to $\sum_{i=1}^{n} x_i^2(d'_i + 1)$ which is the objective value of the optimization problem corresponding to data Y taken at the weight x. Thus, the optimal value of the optimization problem corresponding to data R is always greater than or equal to that corresponding to data Y. Under the same argument, the reverse is true. Thus, the optimal value of the two problems are the same and hence x is an optimal solution for both problem. □

2.2.3 Numerical Computation

We use the Newton-Raphson method to solve the system of equations:

$$\Gamma_j(x) = f(x) + \sum_{k=1}^{n} x_k^2 h_{k,j}^2 - 2d_j x_j = 0, \quad \forall\ j \tag{2.1}$$

The corresponding Jacobian matrix can be calculated as follows:

$$\begin{aligned} J_{j,i} &= \frac{\partial \Gamma_j}{x_i} \\ &= -2x_j d_{ji}^2 + 2d_j 1_{[i=j]} + 2d_{ji} \sum_{k=1}^{n} x_k^2 d_{kj} d_{ki} \\ &\quad -2x_i d_{ji}^2 - 2x_i d_i + \sum_{k=1}^{n} x_k^2 d_{ki}^2, \quad \forall\ j, i \end{aligned} \tag{2.2}$$

The Jacobian matrix can also be written in the following form:

$$J = -(X + X') \cdot D \cdot D + 2 * diag(D) + 2 * D \cdot (D * diag(x^2) * D) - 2 * D + \boldsymbol{e} * (x^2)^t * (D \cdot D),$$

where $X = x * \boldsymbol{e}^t$, D is a matrix of element h_{ji}, the dot product $\cdot$ is the element-wise matrix multiplication while the $*$ operation is the usual matrix multiplication. $diag(D)$ is the diagonal matrix of diagonal elements of D and $diag(x)$ is the diagonal matrix of elements from vector x. This vectorization procedure can make the algorithm run faster in Matlab.

Algorithm 1 Newton Raphson method

1. Set $k = 0$ and start with $x_j^k = \frac{1}{n},\ \forall\ j$
2. Find $\Gamma_j(x^k)$ according to Equation 2.3 and Jacobian matrix $J(x^k)$ according Equation 2.2
3. Set $x^{k+1} = x^k - J(x^k)^{-1}\Gamma(x^k)$ and $k = k + 1$
4. If $\|\Gamma(x^k)\| \leq 10^{-6}$, go to step **5**, otherwise back to step **2**
5. Set $x^* = x^k$.

In the case we know the pre-specified number of outliers p, we set those p observations with the smallest weights x_j^* as outliers once we have found x^*. Otherwise, we could find the weighted maximum likelihood estimators $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w$ and the corresponding Mahalanobis distances d_j. We can then set those observations that are outside the 95% confidence interval, i.e. those with $d_i \geq \chi^2_{m,0.95}$, to be outliers. In our numerical results, we assume we do not know the number of outliers p. The robust location and covariance matrix are then set as the MLEs for the non-outlier subset.

2.3 Numerical Results

2.3.1 Data Generating Processes

Fully Independent Contamination Model (FICM)

We use the popular contamination model by Alqallaf et al. [2]. In this model, observational data is generated by setting $X = (I - B)Y + BZ$, where Y and Z are multivariate normal random variables and B is a diagonal matrix whose diagonal elements come from independent Bernoulli random variables with the probability of success equal to $(1 - p)$. Let the distribution for Y be that of the good observations. Then an observation is in the good set if B is equal to zero (with probability of p^m). Otherwise, the observation is contaminated by Z in at least one attribute.

All Direction Contamination Model (ADCM)

In this model, we have good observations concentrated in the center while outliers lie randomly surrounding good observations by setting their Mahalanobis distances larger than those from the good observation. From the true covariance matrix $\mathbf{\Sigma}$ and the true location $\boldsymbol{\mu}$, we generate n samples R_i from the multivariate normal distribution $N(\boldsymbol{\mu}, \mathbf{\Sigma})$. Let the Mahalanobis distances from the center $\boldsymbol{\mu}$ to these samples be $d_i = (R_i - \boldsymbol{\mu})^t \mathbf{\Sigma}^{-1} (R_i - \boldsymbol{\mu})$ and let $d_{max} = \max(d_i)$. We make sure the last k samples among n samples to be outliers by setting: $R_j = \boldsymbol{\mu} + (1.05 + \xi_j)\sqrt{\frac{d_{max}}{d_j}}(R_j - \boldsymbol{\mu}),\ \forall\ j \geq n - k + 1$, where ξ_j are some nonnegative random numbers. This step essentially scales up the last k samples such that their distances to the center $\boldsymbol{\mu}$ become $(1.05 + \xi_j)^2 d_{max}$. Notice that the higher the expected value of $(1.05 + \xi_j)^2$, the clearer we can distinguish between outliers and non-outliers and the easier the algorithm can detect outliers. In our simulation, we set ξ_j to be uniformly distributed in the range $[0, 0.5]$.

First Quadrant Contamination Model (FQCM)

This model is basically the same with the ADCM model except that outliers lie only in the first quadrant relative to the location. We also use the same method shown

in the previous subsection except that, in the last step, we set: $R_j = \boldsymbol{\mu} + (1.05 + \xi_j)\sqrt{\frac{d_{max}}{d_j}}\,|R_j - \boldsymbol{\mu}|\,,\ \forall\ j \geq n-k+1$. The absolute operation essentially set all the outliers to be in the first quadrant of the coordinates relative to the location $\boldsymbol{\mu}$.

2.3.2 Outlier Detection Examples

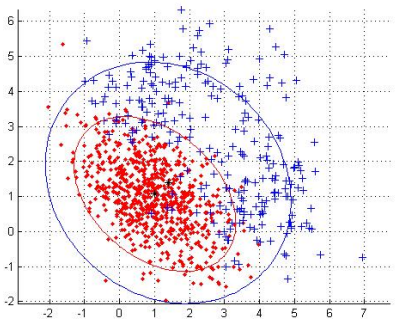
We present three examples to demonstrate our robust estimators under different contamination models and for different choices of the true locations and covariance matrices. In the first example (the first plot shown in Figure 2-3), we use the FICM model where the data were generated from:

$$X = \begin{bmatrix} 1-b_1 & 0 \\ 0 & 1-b_2 \end{bmatrix} * \mathcal{N}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right) + \begin{bmatrix} b_1 & 0 \\ 0 & b_2 \end{bmatrix} * \mathcal{N}\left(\begin{bmatrix} 4 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

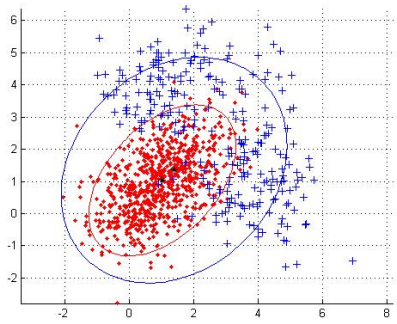
In order to produce 25% contamination, we set b_1 and b_2 as independent Bernoulli random variables with the probability of success equal to $1 - \sqrt{0.75}$. The data in the second plot is essentially the same with that in the first plot except that the covariance matrix of Y was changed to $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. The data in the third plot in Figure 2-3 was generated from the FQCM model with the true distribution being: $\mathcal{N}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}\right)$. We can see that in all the three examples, our robust estimators recover the location and the covariance matrix very well.

2.3.3 Outlier Detection Accuracy and Computational Time

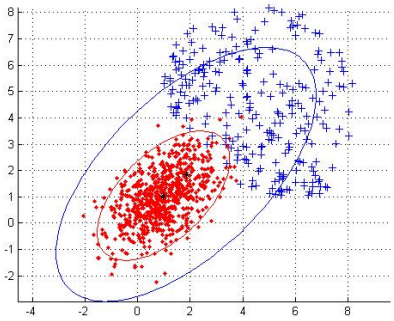
We check the accuracy of our robust estimators with the number of observations n ranging from 100 to 5000 and the problem dimension m ranging from 2 to 50. For each problem size, we generate K sample paths where K is equal to 20 for $n \leq 500$ and $K = 10$ for $n \geq 1000$. The sample paths are generated with fixed random seeds in Matlab so that they can be regenerated for future comparison. Each sample path corresponds to a matrix of size $n \times m$ data generated from the FICM model with Y comes from m-multivariate normal distribution $N(0, I)$ and Z comes from



(a) FICM I



(b) FICM II



(c) FQCM

Figure 2-3: Three examples demonstrating how our robust estimation technique performs under different contamination models. They show our technique can detect outliers and recover the distribution of the majority of the observations nicely. In the first plot, 1000 points were generated from the fully independent contamination model, where the Bernoulli random variables have probabilities of success of $p = 1 - \sqrt{0.75}$ such that the expected percentage of non-contaminated observations is 75%. The expected percentage of observations being contaminated in one attribute is $p(1-p)$ while that of observations being contaminated in both attributes is $(1-p)^2$. In this example, we have 750 good observations which are the red, dotted points. The contaminated observations are the blue, crossed points. The larger ellipsoid (in blue) contains 95% of the observations that are conformed with the classical MLE covariance, i.e. those points whose distances d are less than or equal to $\chi^2_{0.95,2}$: $d_j = (R_i - \mu_{MLE})^t \mathbf{\Sigma}^{-1}_{MLE}(R_i - \mu_{MLE}) \leq \chi^2_{0.95,2}$. The smaller ellipsoid (in red) contains 95% of the observations that are conformed with the robust covariance $\mathbf{\Sigma}$ and the robust location $\bar{R}$, i.e. those points whose distances d are less than or equal to $\chi^2_{0.95,2}$: $d_j = (R_i - \bar{R})^t \mathbf{\Sigma}^{-1}(R_i - \bar{R}) \leq \chi^2_{0.95,2}$. Similarly, in the second plot, 1000 points were generated from the fully independent contamination model but using a different covariance matrix with the same contamination parameters. In the third plot, 1000 points were generated from the first quadrant contamination model with 750 good observations and 250 outliers.

m-multivariate normal distribution $N(4\boldsymbol{e}, I)$, where I is the identity matrix and $\boldsymbol{e}$ is the column identity vector of size m of all 1. The Bernoulli random variables are independent with the probability of success being $1 - \sqrt[m]{p}$ such that the probability of having a non-contaminated observation is p where p is a uniform random number in the range $[0.75, 0.95]$. Figure 2-4 shows an instance generated by the FICM model with $n = 1000$ and $m = 2$. Notice that although we use the identity matrix to generate the sample data, the affine equivariant property implies that we can transform the data to any type of covariance matrix we want and still be able to obtain the same results.

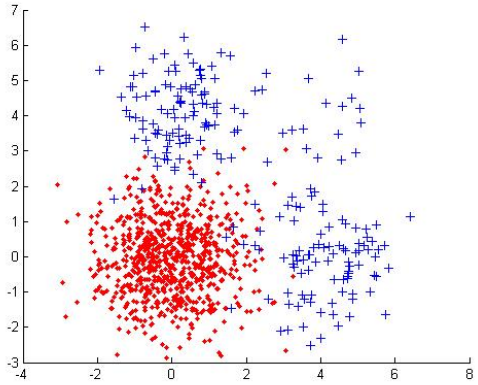


Figure 2-4: An example showing how the data generated from the Fully Independent Contamination Model looks like with $n = 1000$ and $m = 2$. The red dotted points are good observations and the blue crossed points are contaminated observations.

By keeping track of the realization of the Bernoulli variables, we record those contaminated observations. For each sample path, we run our robust estimation algorithm to find out those observations that lie outside the 95% confidence ellipsoid and set those as outliers. The accuracy of the method is calculated as follows:

$$\text{Accuracy} \quad = \quad 1 - \frac{E_1 + E_2}{n}$$

where E_1 is the number of unidentified outliers and E_2 is the number of good observations that are misclassified to be outliers. We record the average accuracy over K sample paths.

Table 2.1 shows the average computation time (in seconds) for different problem sizes. [1] We can see that when n increases, the computation time increases at the rate of $\mathrm{O}(n^3)$. This is what we expected because the Newton-Raphson algorithm takes around 7 iterations for most of the simulations. Each iteration takes $\mathrm{O}(n^3)$ operations to compute the Jacobian matrix and its inverse to find a new direction. It is also very interesting to see that the computational time for each fixed n is almost the same for all problem dimensions, i.e. the numbers in each column are about the same.

	100	**200**	**500**	**1000**	**2000**	**5000**
2	0.0265	0.146	1.286	6.74	31.78	364.49
5	0.0235	0.140	1.149	6.11	30.82	386.83
10		0.136	1.170	6.13	29.72	351.26
20			1.103	5.48	29.86	355.57
30				5.09	28.75	332.66
50						304.35

Table 2.1: Computation time of our robust estimation technique for different problem sizes under the Fully Independent Contamination Model. The top row is for the number of observations n while the first column is for the problem dimensions m. The numbers in the table show the average computation time (in seconds).

Table 2.2 shows that, overall, the accuracy is around 93% which is quite high. Notice that since the distribution of Y and Z are not very far from each other, there is a crossover between the good observations and the contaminated observations simply because of chance.

We obtain similar computational time for the FQCM and the ADCM models compared to that from the FICM model as shown in Tables 2.3 and 2.5. The average accuracy is around 97% for the ADCM model and around 90% for the FQCM model as shown in Tables 2.4 and 2.6.

We also compare the computational time between our algorithm and the Fast-MCD using the FICM, ADCM and FQCM models. We used the LIBRA library (see [43]) for the Fast-MCD algorithm. it is not clear about the exact formula for the

[1] All the computation was done through Matlab version 7.3.0.298 (R2006b) under the Linux 2.6.15-51-amd64-xeon system. Notice that all the test data was generated randomly in Matlab using fix seeds such that they can be regenerated for testing with other techniques. Matlab codes for the algorithm and for generating test data are available on request.

	100	**200**	**500**	**1000**	**2000**	**5000**
2	93.45%	90.35%	94.14%	95.02%	94.68%	94.87%
5	93.95%	93.2%	93.77%	94.39%	94.48%	94.29%
10		93.53%	93.72%	93.81%	93.525%	93.88 %
20			93.27%	92.29%	92.77%	92.45%
30				92.27%	92.85%	91.71%
50						91.01%

Table 2.2: Outlier detection accuracies of our robust estimation technique for different problem sizes under the Fully Independent Contamination Model. The top row contains the numbers of observations n while the first column contains the problem dimensions m. The numbers in the table show the average detection accuracies which measure the percentage of the accurately classified points over n.

	100	**200**	**500**	**1000**	**2000**	**5000**
2	0.035435	0.15149	1.1415	7.2779	39.597	418.01
5	0.023327	0.15311	1.4436	7.0569	36.595	469.4
10		0.1653	1.293	7.1696	34.935	547.12
20			1.4099	6.661	34.822	1044.9
30				6.4929	33.61	563.68
50						819.02

Table 2.3: Computation time of our robust estimation technique for different problem sizes under the All Direction Contamination Model. The top row is for the number of observations n while the first column is for the problem dimensions m. The numbers in the table show the average computation time (in seconds).

complexity of the Fast-MCD. However, its complexity grows exponentially when we increase the problem dimension m and grows linearly when we increase n. On the other hand, the complexity of our algorithm is $O(n^3)$ which is independent of the problem dimension m. Therefore, our method outperforms the Fast-MCD significantly when either n is small or m is large. For large n and small m, the Fast-MCD algorithm is faster than ours. We also compare the accuracies between our algorithm and the Fast-MCD. We found the two methods produce about the same levels of accuracy at around 93% for data generated from the FICM model. For the ADCM model, our method produces an average accuracy of around 97% while that of the Fast MCD is around 92.5%. For the FQCM model, our method produces an average accuracy of around 90% while that of the Fast-MCD is around 94%. In conclusion, overall, both method produces around the same levels of accuracies. The Fast-MCD algorithm is

	100	200	500	1000	2000	5000
2	96.60%	97.23%	97.48%	98.19%	97.86%	97.31%
5	96.00%	94.80%	99.01%	96.31%	98.32%	97.28%
10		97.83%	97.77%	96.02%	97.44%	95.07%
20			96.45%	97.79%	97.24%	97.14%
30				95.43%	98.40%	96.36%
50						95.60%

Table 2.4: Outlier detection accuracies of our robust estimation technique for different problem sizes under the All Direction Contamination Model. The top row contains the numbers of observations n while the first column contains the problem dimensions m. The numbers in the table show the average detection accuracies which measure the percentage of the accurately classified points over n.

	100	200	500	1000	2000	5000
2	0.047549	0.27515	1.5284	9.6296	51.118	662.31
5	0.04619	0.22515	1.8701	9.0582	58.188	667.62
10		0.2346	1.8062	9.5765	50.899	618.47
20			1.6024	10.205	46.112	613.42
30				9.197	41.011	586.26
50						558.34

Table 2.5: Computation times of our robust estimation technique for different problem sizes under the First Quadrant Contamination Model. The top row are for the number of observations n while the first column is for the problem dimensions m. The numbers in the table show the average computation time (in seconds).

more favorable for problems with large number of observations and smaller problem dimensions. Our method is more favorable for high dimension problems or small numbers of observations.

2.4 Incorporating Prior Information via Semi-definite Programming Reformulation

Our robust covariance estimation model is most attractive in the way it can incorporate additional constraints into the optimization model. In general, we can add constraints as long as they do not add additional complexity to the model. We will show that the robust covariance estimation problem can be reformulated as a semi-definite programming problem, any SDP-like constraints can be added to the model.

	100	**200**	**500**	**1000**	**2000**	**5000**
2	91.45%	91.23%	92.30%	94.56%	93.20%	93.09%
5	90.25%	88.03%	91.40%	90.42%	91.91%	90.93%
10		90.73%	89.19%	87.11%	88.31%	85.49%
20			90.95%	89.96%	89.68%	89.04%
30				87.86%	88.27%	87.20%
50						85.68%

Table 2.6: Outlier detection accuracies of our robust estimation technique for different problem sizes under the First Quadrant Contamination Model. The top row contains the numbers of observations n while the first column contains the problem dimensions m. The numbers in the table show the average detection accuracies which measure the percentage of the accurately classified points over n.

These constraints could be additional beliefs that we have on the location, the covariance matrix and the observations. This capability is very important in many applications such as in finance where one strategy is distinguished from others by the capacity of the managers to incorporate their beliefs into the model. For example, we might have a strong belief that the mean (location) $\hat{\boldsymbol{\mu}}_{rob}$ lies within some known ellipsoid $E(c, S)$ such that $(\hat{\boldsymbol{\mu}}_{rob} - c)^t S^{-1} (\hat{\boldsymbol{\mu}}_{rob} - c) \leq 1$. Then we can add the following constraint into the model:

$$\begin{bmatrix} 1 & \sum_{i=1}^{n} x_i R_i^t - c^t \\ \sum_{i=1}^{n} x_i R_i - c & S \end{bmatrix} \succeq 0$$

In finance, we might want to restrict the expected return to be non negative and this can be done by setting $\hat{\boldsymbol{\mu}}_{rob} \geq 0$. We might have additional information that the correlation between two assets is positive or negative. If we knew an observation R_i to be in the good set, we can set $d_i < 0.975$. We can even add a constraint like observation i has a higher probability of being a good observation than observation j does by setting $x_i \geq x_j$. We might add constraints on the shape of the covariance matrix. For example, in order to make sure the dispersion (the covariance matrix) is not so flat, we can set the minimum eigenvalue of the covariance matrix to be greater than some bound or the ratio between the maximum eigenvalue and the minimum eigenvalue to be within some ranges. We skip the mathematical formulation for these

constraints for the purpose of clarity. The detail is available from the author on request.

2.4.1 Semi-definite Programming Reformulation

We have already shown that solving the optimization model is equivalent to solving the system of nonlinear equations. The optimization model is nonconvex in variables (x, d) and is highly nonlinear in the inverse of $\hat{\mathbf{\Sigma}}$ and in the term $\bar{R}\bar{R}^t$ in the constraints, and it seems very difficult to solve. However, we will show in the computation results that the system of equations can be efficiently solved using the Newton-Raphson method. It has been shown experimentally that it often takes fewer than 10 iterations for the Newton-Raphson to converge to $||\Gamma|| \leq 10^{-6}$ for even large scale problems with n as high as 10000 and m can be up to hundreds. These results seem conflicting. However, we can show that the optimization model is actually equivalent to a semi-definite programming problem in a different space and can be solved efficiently. By showing the reformulation of the robust covariance estimation as a semi-definite programming problem, we not only explain why we can simply use the Newton-Raphson method to solve an equivalent system of equations but we can also incorporate many different types of prior beliefs by adding additional constraints to the model as long as the constraints can be transformed into SDP forms and do not increase the problem complexity. We will first show how we can reformulate the problem in SDP form. Let $y_i = x_i^2(d_i + 1)$, the objective becomes $\sum_{i=1}^{n} y_i$, while the

Mahalanobis distance constraint can be reformulated as follows:

$$
\begin{aligned}
& d_i \geq (R_i - \bar{R})^t \mathbf{\Sigma}^{-1} (R_i - \bar{R}) \\
\Leftrightarrow \quad & y_i - x_i^2 \geq x_i^2 (R_i - \bar{R})^t \mathbf{\Sigma}^{-1} (R_i - \bar{R}) \\
\Leftrightarrow \quad & \begin{bmatrix} y_i - x_i^2 & x_i (R_i - \bar{R})^t \\ x_i (R_i - \bar{R}) & \mathbf{\Sigma} \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & \begin{bmatrix} y_i - x_i^2 & x_i R_i^t - x_i \bar{R}^t \\ x_i R_i - x_i \bar{R} & \sum_{i=1}^n x_i R_i R_i^t - \bar{R}\bar{R}^t \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & \begin{bmatrix} y_i & x_i R_i^t \\ x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} - \begin{bmatrix} x_i^2 & x_i \bar{R}^t \\ x_i \bar{R} & \bar{R}\bar{R}^t \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & \begin{bmatrix} y_i & x_i R_i^t \\ x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} - \begin{bmatrix} x_i \\ \sum_{i=1}^n x_i R_i \end{bmatrix} \begin{bmatrix} x_i & \sum_{i=1}^n x_i R_i^t \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & \begin{bmatrix} 1 & x_i & \sum_{i=1}^n x_i R_i^t \\ x_i & y_i & x_i R_i^t \\ \sum_{i=1}^n x_i R_i & x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq 0
\end{aligned}
$$

Notice that this constraint is a typical SDP constraint where all the terms in the SDP matrix are linear functions of the decision variables x, y. The corresponding SDP reformulation is shown in Model 4:

Model 4 : Semi-definite Programming Model for Robust Covariance Estimation

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i=1}^n y_i \\
s.t \quad & \begin{bmatrix} 1 & x_i & \sum_{i=1}^n x_i R_i^t \\ x_i & y_i & x_i R_i^t \\ \sum_{i=1}^n x_i R_i & x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq 0, \qquad \forall\, i \\
& x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}
$$

Remark 1: We did not show every step in the above proof. The detail is similar to the proof of Proposition 1 shown in the Appendix. The SDP problem can be solved

by interior point methods using standard software packages like the SDPT3 [39]. If k is known, we sort the weights x_i and take the smallest k weights as outliers. If k is unknown, we only take those observations whose Mahalanobis distances are smaller than $\chi^2_{m+1,0.95}$ as good observations.

2.4.2 Existence and Uniqueness of the Newton Raphson Solution and the SDP Solution

In Subsection 2.2.3 and Section 2.4, we present two different approaches for solving the robust estimation problem. In the first approach, we prove that the optimal solution of the robust optimization model 2 satisfies a system of equations. We then use the Newton Raphson method to solve the system. Our numerical results show the Newton Raphson method behaves well. However, we did not present any theoretical result about the existence and uniqueness of the optimal solution. In the second approach, we prove that our robust optimization model 2.2.3 can be transformed into the semi-definite programming model 4. However, we did not show whether the original model and the SDP model are equivalent in the sense that the set of solutions produced by these models are the same. In this Subsection, we will bridge the gaps by showing all the methods lead to the same unique solution.

First, we prove that the original robust optimization model 2 is equivalent to the simplified model 3. This is done by noticing the inequality constraint of $d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_i - \hat{\boldsymbol{\mu}}_{rob})$ to be always tight at the optimal solution. Therefore, we can replace $d, \hat{\boldsymbol{\mu}}_{rob}, \hat{\boldsymbol{\Sigma}}_{rob}$ from the original model by their functions of x and arrive at the simplified model.

Original Robust Model 2

$$\begin{aligned}
\min_{x,d,\hat{\boldsymbol{\mu}}_{rob},\hat{\boldsymbol{\Sigma}}_{rob}} \quad & \sum_{i=1}^{n} x_i^2(d_i+1) \\
s.t \quad & \hat{\boldsymbol{\mu}}_{rob} = \sum_{i=1}^{n} x_i R_i \\
& \hat{\boldsymbol{\Sigma}}_{rob} = \sum_{i=1}^{n} x_i R_i R_i^t - \hat{\boldsymbol{\mu}}_{rob}\hat{\boldsymbol{\mu}}_{rob}^t \\
& d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1}(R_i - \hat{\boldsymbol{\mu}}_{rob}) \\
& x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}$$

Simplified Model 3

$$\begin{aligned}
\Leftrightarrow \quad \min_{x} \quad & \sum_{i=1}^{n} x_i^2 v_i^t \left(\sum_{i=1}^{n} x_i v_i v_i^t \right)^{-1} v_i \\
s.t \quad & x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}$$

We then prove that the constraint $x \geq 0$ is redundant and can be removed. Therefore, we can use the KKT conditions to arrive at the following system of equations:

$$\Gamma_j(x) = f(x) + \sum_{k=1}^{n} x_k^2 h_{k,j}^2 - 2 d_j x_j = 0, \quad \forall\ j \tag{2.3}$$

In Subsection 4, we show that each of the constraints $y_i \geq x_i^2 v_i^t \left(\sum_{i=1}^{n} x_i v_i v_i^t\right)^{-1} v_i$ is an SDP-typed constraint. This implies the level sets of the function $x_i^2 v_i^t \left(\sum_{i=1}^{n} x_i v_i v_i^t\right)^{-1} v_i$ are convex. The level sets of the objective function are therefore convex. Since the objective function is also a continuously differentiable function, the KKT conditions are necessary and sufficient conditions. Therefore, the simplified model is equivalent

to the system of equations. We then use the Newton-Raphson method to solve that system of equations.

In addition, by introducing the additional variable y_i to the simplified model, we arrive at the semi-definite programming model:

$$\begin{aligned}
\min_{x,y} \quad & \sum_{i=1}^{n} y_i \\
s.t \quad & \begin{bmatrix} 1 & x_i & \sum_{i=1}^{n} x_i R_i^t \\ x_i & y_i & x_i R_i^t \\ \sum_{i=1}^{n} x_i R_i & x_i R_i & \sum_{i=1}^{n} x_i R_i R_i^t \end{bmatrix} \succeq 0, \qquad \forall\ i \\
& x^t \boldsymbol{e} = 1 \\
& x \geq 0
\end{aligned}$$

We can see the set of optimal solutions of the simplified model and the set of optimal solutions of the SDP model are exactly the same. In addition, since the objective function for the simplified model is a non-linear continuously differentiable function and the constraint set is a bounded simplex, there exists an unique solution. Therefore, the solutions from the Newton-Raphson method and the SDP model are exactly the same.

2.4.3 Incorporating Prior Information Example

In this example, we demonstrate the effect of having prior information to the robust estimator, which could produce a breakpoint smaller than 50%. In Figure 2-5, we generate 1000 points from the following two normal distributions: $\mathcal{N}\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}\right)$ and $\mathcal{N}\left(\begin{bmatrix}5\\1\end{bmatrix}, \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}\right)$ with 400 points from the first distribution and 600 points from the second. Suppose the 400 points are good observations. Since the good observations account for only the minority, typical robust methods without any prior information should find the parameters that correspond to the majority of 600 points. However, if we have additional information that the true location $\bar{R}$ lies within the

ellipsoid:

$$(\bar{R} - \boldsymbol{\mu}_{prior})^t \boldsymbol{\Sigma}_{prior}^{-1} (\bar{R} - \boldsymbol{\mu}_{prior}) \leq 1,$$

where $\boldsymbol{\mu}_{prior} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\boldsymbol{\Sigma}_{prior} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, then we could recover the good observations and find the robust estimators corresponding the 400 points.

2.5 Conclusion

In conclusion, we have studied the robust covariance estimation problem via outlier detection. Our approach is different from others in the way we introduce a probability measure on the observations and in the design of an optimization problem on that measure such that outliers are set with smaller probabilities at the optimal solution. This optimization problem was proven to be equivalent to solving a system of nonlinear equations. We use the Newton-Raphson method to solve the system. The algorithm behaves very well computationally because the optimization model turns out to be equivalent to a semi-definite programming problem. This SDP reformulation is very flexible such that we can incorporate prior belief into the robust estimators by adding additional constraints into the optimization model. We believe that using SDP is a promising choice for robust statistics applications since they both closely involve matrix operations. We demonstrate our robust estimators through different examples. We also compare our method with the Fast-MCD method in terms of accuracy and computational time for different contamination models at different problem sizes. Overall, our estimators are slightly more accurate and the algorithm runs much faster than the Fast-MCD for large problem dimensions.

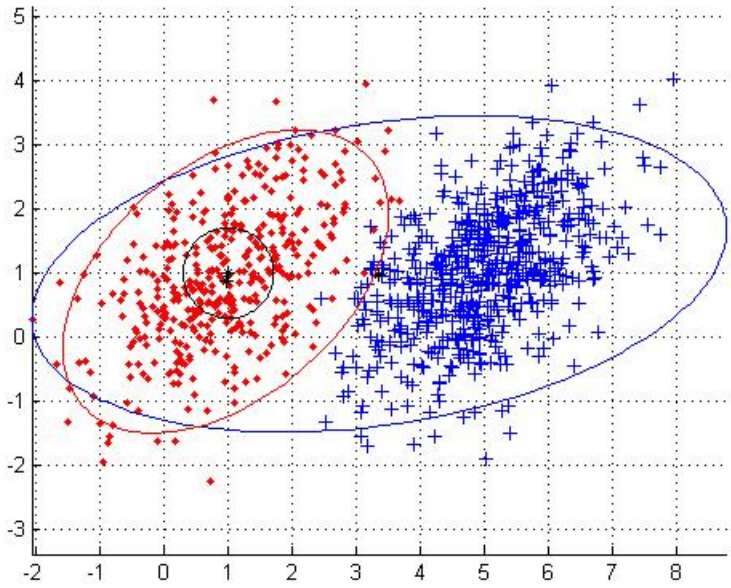


Figure 2-5: An example illustrating the effect of incorporating prior information into the robust estimators. There were 1000 points generated from a mixture of two normal distributions. The 400 red dotted points are considered good observations while the 600 blue crossed points are considered outliers. The largest ellipsoid (in blue color) is the ellipsoid that contains 95% of the observations using the MLE estimators. The second largest ellipsoid is that one that contains 95% of the good observations using the robust estimators. The smallest ellipsoid represents the prior belief that the location must be contained in this ellipsoid. Since the number of good observations accounts for only 40% of all observations, a robust method without prior information would classify the majority of 600 points as good observations. However, with the additional knowledge that the truth location is within the ellipsoid $(\bar{R} - \boldsymbol{\mu}_{prior})^t \boldsymbol{\Sigma}_{prior}^{-1} (\bar{R} - \boldsymbol{\mu}_{prior}) \leq 1$, the robust estimator can recover up to 98.5% of the good observations. (notice that the two distributions cross over such that there are some unclassifiable points and there is no way to recover 100% good observations.)

Chapter 3

Robust Regression

3.1 Introduction

3.1.1 The Least Squares Regression Problem

Let $X \in \mathbb{R}^{n \times p}$ be the data matrix of realizations for the independent (explanatory) variable and $y \in \mathbb{R}^n$ be the vector of realizations for the dependent (response) variable. Least squares regression assumes:

$$y = X\beta + \epsilon$$

where $\beta \in R^p$ is some unknown coefficient vector and $\epsilon \in \mathbb{R}^n$ is a vector of i.i.d. realizations for an error term which is normally distributed with mean 0.

In ordinary least squares, we estimate β as the minimizer of the sum of squares:

$$\hat{\beta}_{LS} = \arg\min_{\beta} \qquad (y - X\beta)^t (y - X\beta)$$

Taking the derivative of the objective function and setting it to zero, we get:

$$\hat{\beta}_{LS} \;\; = \;\; \left(X^t X\right)^{-1} X^t y$$

3.1.2 Outliers and the Need for Robust Estimation

It is well-known that the least squares fit can be grossly influenced by outliers. Figure 3-1 for instance, shows a case where the fit is pulled towards a single outlier. The red dashed line shows the least squares fit using all observations. The blue solid line shows the least trimmed squared (LTS) fit where the the outlier is excluded. We can see the dashed line does not fit the majority of the data as well as the solid line.

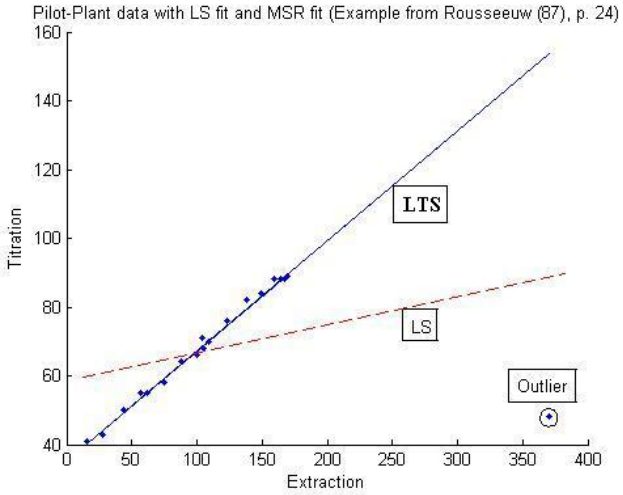


Figure 3-1: An example demonstrating how a single outlier can grossly deteriorate the least squares estimator. The outlier pulls the least squares line of the subset without the outlier from the solid line to the dashed line. This example shows that least squares does not work well under the presence of outliers. In general, least squares performs poorly if the model assumption is incorrect, i.e. if the residuals do not follows a normal distribution.

We will review different methods for handling outliers in the next Subsection.

3.1.3 Literature Review

An intuitive approach for outlier detection is to use the classical least squares estimator on the entire data set and then remove those observations with unexpectedly large squared residuals. However, this approach does not work well because the outliers have already corrupted the least squares fit such that the squared residuals we are comparing are significantly different from those computed using the robust regression

fit.

Researchers have developed several methods to work around this. Holland and Welsch [19] use the iterative reweighed least squares procedure where the weights are set inversely proportional to the squared residuals until convergence occurs. Rousseeuw [34] proposes least median of squares (LMS) regression, which minimizes the median squared residuals, and was originally computed by the PROGRESS algorithm. The least trimmed squares problem was presented in 1987 by Rousseeuw and Leroy [35]. They show that LTS and LMS have desirable properties, including affine equivariance, the highest breakdown possible (50%), asymptotic normality etc. These methods and other methods such as M-Estimates, L-Estimates, MM-Estimates, S-Estimates, and R-Estimates are documented in standard robust statistics textbooks by Huber [20] in 2004, Maronna et al. [28] in 2006.

Since least trimmed squares has very desirable properties and forms the basis for several algorithms that followed, there has recently been interest in making the procedure more computationally feasible. The Fast-LTS was discovered by Rousseeuw and Driessen [37] in 2006 and is probably the best one in practice. This method can be viewed as a combination of a gradient method and a genetic algorithm. From the Operations Research community, Ziouta and Avramidis [45] in 2005 and Bertsimas and Shioda [5] in 2007 present mixed integer programming (MIP) models for the classification and robust regression problems. Ziouta and Avramidis allow the number of outliers in the least trimmed squares problem to be unknown by adding a penalty term to those trimmed observations. The corresponding MIP model is very complex such that the authors can only test with small problems sizes with $p \leq 3$. Bertsimas and Shioda use powerful MIP models for classification and regression. However, these models assume the data follows a special structure in the training, validation and testing sets in order to reduce the complexity of the MIP formulations.

3.1.4 Contributions

- We show the least trimmed squares regression problem to be equivalent to a concave minimization problem. This result is very important as it motivates the

application of mathematical programming, especially concave minimization, to the robust regression problem. Traditionally, the robust regression problem is viewed as a mixed integer programming problem and hence there is a hesitation in studying this problem from the mathematical programming view as it is well-known that integer programming does not scale well. On the other hand, by showing the problem to be equivalent to a concave minimization problem under a very simple cardinality constraint, there is a good chance that we can apply future developments in nonlinear concave optimization to this problem.

- We introduce the idea of "Maximum Trimmed Squares" (MTS) as one way to exploit the nice structure of the problem, i.e. concave objective and cardinality constraint, to solve the robust regression problem. We show that the MTS is guaranteed to detect outliers. In addition, we show that the MTS is equivalent to a semi-definite programming model and can be solved efficiently using interior point methods.

3.1.5 Chapter Structure

The rest of this Chapter is organized as follows. In section 2, we review least trimmed squares and reformulate it into a nonlinear programming problem with a concave objective function and a convex constraint set. In section 3, we introduce maximum trimmed squares and prove two theoretical results about it. In section 4, we provide numerical results, and lastly, in section 5, we conclude the Chapter.

3.2 The Least Trimmed Squares (LTS) Regression Problem

We have seen that outliers can strongly corrupt the least squares fit due to their dominant effect on the objective function. Least trimmed squares attempts to mitigate this problem by minimizing the sum of the smallest k squared residuals rather than the complete sum of squares. Here, k is a threshold such that the ratio $(1 - k/n)$

represents the percentage of outliers among all the observations. We usually know k based on some prior knowledge we have about the data or we could set the ratio $k/n = 0.75$ as a typical value. In this Chapter, we assume we already know k/n.

3.2.1 Formulating LTS as a Nonlinear Programming Problem

Let z_i be the indicator for whether observation i is a good observation or not. Then, it is easy to see that the least trimmed squares estimation problem can be reformulated as the following mixed integer programming problem:

$$\begin{aligned} \min_{\beta,z} \quad & \sum_{i=1}^{n} \underbrace{z_i(y_i - x_i\beta)^2}_{g(z,\beta)} \\ s.t. \quad & z^t\boldsymbol{e} = k \\ & z_i \in \{0,1\} \end{aligned}$$

where y_i is the response variable and x_i is the row vector explanatory variable for observation i.

We turn this into a nonlinear programming formulation as follows: Let (z^*, β^*) be an optimal solution of the LTS, i.e. (z^*, β^*) minimize $g(z, \beta)$ under the constraint $\{z^t\boldsymbol{e} = k, z_i \in \{0,1\}\}$. Then z^* must be an optimal solution of $g(z, \beta^*)$ under the same constraint set. We notice that, once we fix $\beta = \beta^*$, $z_j^* = 1$ for the smallest k values $(y_j - x_j\beta)^2$ at the optimal solution (in the case of ties, we choose the largest among those tied by randomly picking).

It follows that when β is fixed, the integral constraints $z_i \in \{0,1\}$ can be relaxed to $0 \leq z_i \leq 1$ without losing the optimality. In general, β is not fixed. However, the problem of minimizing $g(z, \beta)$ such that $\{z^t\boldsymbol{e} = k, z_i \in \{0,1\}\}$ is equivalent to:

$$\begin{aligned} \min_{\beta} \min_{z} \quad & \sum_{i=1}^{n} \underbrace{z_i (y_i - x_i \beta)^2}_{g(z,\beta)} \\ s.t. \quad & z^t \boldsymbol{e} = k \\ & z_i \in \{0, 1\} \end{aligned}$$

Hence, on solving the inner optimization problem, β is treated as fixed, so we can replace the integral constraint in general and rewrite the LTS formulation as the following nonlinear programming problem:

$$\begin{aligned} \min_{\beta, z} \quad & \sum_{i=1}^{n} \underbrace{z_i (y_i - x_i \beta)^2}_{g(z,\beta)} \\ s.t. \quad & z^t \boldsymbol{e} = k \\ & 0 \leq z_i \leq 1 \end{aligned}$$

We notice that the objective function is neither convex nor concave and it is quite hard to analyze its structure and to solve it efficiently. In the next Subsections, we will show how the model can be reformulated to a new form with fewer decision variables, a concave objective function and the same constraint set.

3.2.2 LTS Reformulation

We reformulate the problem here by rewriting $\min_{\beta,z} g(z, \beta)$ as $\min_{z}(\min_{\beta} g(z, \beta))$ and solving the inner problem in closed form. In particular, suppose we already know the indicator variables z_i. Let $Z \equiv D(z)$ be the diagonal matrix with diagonal elements equal to z. Then, $g(z, \beta)$ can be written as $(y - X\beta)^t Z (y - X\beta)$ and the constraints are merely true statements since a feasible z is already known, so the solution to the optimization problem is:

$$\hat{\beta}_{LTS} = \left(X^t Z X\right)^{-1} X^t Z y$$

Thus, the total trimmed squared residuals, or $g(z, \beta)$ is:

$$\begin{aligned} f(z) &= \epsilon_{LTS}^t Z \epsilon_{LTS} \\ &= \left(y - X\hat{\beta}_{LTS}\right)^t Z \left(y - X\hat{\beta}_{LTS}\right) \\ &= y^t Z y - y^t Z X \left(X^t Z X\right)^{-1} X^t Z y \end{aligned}$$

The problem is thus reformulated as follows:

Model 5 : Least Trimmed Squares Reformulation

$$\begin{aligned} \min_{z} \quad & \underbrace{y^t D(z) y - y^t D(z) X \left(X^t D(z) X\right)^{-1} X^t D(z) y}_{f(z)} \\ s.t. \quad & z^t \boldsymbol{e} = k \\ & 0 \leq z_i \leq 1 \end{aligned}$$

This model is less complicated than the original one because we have removed the variable β from the model while still keeping the constraint set unchanged. In the next Subsection, we will present an iterative procedure to solve LTS with this new formulation.

3.2.3 An Iterative Method for LTS

The standard gradient method is to set those k indicators z_i with the smallest first order partial derivative to one. We iteratively perform this step until the solution is stable. The partial derivatives are found as follows:

$$
\begin{aligned}
\frac{\partial f(z)}{\partial z_i} &= \frac{\partial \left(y^t Z y - y^t Z X \left(X^t Z X \right)^{-1} X^t Z y \right)}{\partial z_i} \\
&= y_i^2 - 2 \frac{\partial (y^t Z X)}{\partial z_i} \left(X^t Z X \right)^{-1} X^t Z y - y^t Z X \frac{\partial \left((X^t Z X)^{-1} \right)}{\partial z_i} X^t Z y \\
&= y_i^2 - 2 y_i x_i \left(X^t Z X \right)^{-1} X^t Z y + y^t Z X \left(X^t Z X \right)^{-1} x_i^t x_i \left(X^t Z X \right)^{-1} X^t Z y \\
&= (y_i - x_i \underbrace{\left(X^t Z X \right)^{-1} X^t Z y}_{\beta(z)})^2 \qquad (3.1)
\end{aligned}
$$

Remark: It is interesting to see that the gradient for an observation j is exactly the squared residual of that observation. This result actually makes sense because, by definition, the gradient is the relative change of the objective function given a small change in the decision variable. The change in the objective function when adding a new observations is exactly the squared residual of the new observation.

We have the following iterative algorithm:

Input:	Data $X \in \mathcal{R}^{n \times p}$, $y \in \mathcal{R}^n$, $k \in \mathcal{R}$.
Output:	Robust regressor β^*.
Algorithm:	**1**. Set $t = 0$ and start with random z^0. **2**. Find derivatives $\frac{\partial f(z)}{\partial z_i}$ at z^t according to Equation 3.1. **3**. Set $z_i^{t+1} = 1$ for those indices i, for which $\frac{\partial f(z)}{\partial z_i}$ is among the smallest k derivatives and set $z_i^{t+1} = 0$ otherwise. **4**. If $z^{t+1} = z^t$, go to step **5**, otherwise, set $t = t + 1$ and go back to step **2**. **5**. Set β^* to be the least squares regressor for the set of observations j with $z_j^t = 1$.

Table 3.1: An iterative algorithm for Least Trimmed Squares regression

We keep iterating from one subset to a better subset. At each step, the new subset contains the k observations with the smallest squared residuals.

3.2.4 The Concavity of the LTS

We will show the objective function of the LTS problem is concave by proving $\frac{\partial^2 f(z)}{\partial z^2} \preceq 0$, where the notation $\preceq$ (or $\succeq$) denotes the negative (or positive) definite of a matrix. From the result in the previous Subsection, we have:

$$\frac{\partial f(z)}{\partial z_i} = (y_i - x_i\beta(z))^2 \tag{3.2}$$

In order to find the second derivatives $\frac{\partial^2 f(z)}{\partial z_i \partial z_j}$, we will first find $\frac{\partial \beta(z)}{\partial z_i}$ as follows:

$$\begin{aligned}
\frac{\partial \beta(z)}{\partial z_i} &= \frac{\partial \left((X^t Z X)^{-1} X^t Z y \right)}{\partial z_i} \\
&= \left(X^t Z X\right)^{-1} \frac{\partial (y^t Z X)}{\partial z_i} + \frac{\partial \left((X^t Z X)^{-1} \right)}{\partial z_i} X^t Z y \\
&= \left(X^t Z X\right)^{-1} (y_i x_i^t) - \left(X^t Z X\right)^{-1} x_i^t x_i \left(X^t Z X\right)^{-1} X^t Z y \\
&= \left(X^t Z X\right)^{-1} x_i^t (y_i - x_i \left(X^t Z X\right)^{-1} X^t Z y) \\
&= \left(X^t Z X\right)^{-1} x_i^t (y_i - x_i \beta(z)) \qquad (3.3)
\end{aligned}$$

From equalities 3.2 and 3.3, we have:

$$\begin{aligned}
\frac{\partial^2 f(z)}{\partial z_i \partial z_j} &= \frac{\partial (y_i - x_i\beta(z))^2}{\partial z_j} \\
&= -2(y_i - x_i\beta(z)) x_i \frac{\partial \beta(z)}{\partial z_j} \\
&= -2(y_i - x_i\beta(z)) x_i \left(X^t Z X\right)^{-1} x_j^t (y_j - x_j\beta(z)) \\
&= -2(x_i \epsilon_i) \left(X^t Z X\right)^{-1} (x_j \epsilon_j)^t \\
\Rightarrow \frac{\partial^2 f(z)}{\partial z^2} &= -2M \left(X^t Z X\right)^{-1} M^t \preceq 0
\end{aligned}$$

where $M = X diag(\epsilon)$. The last inequality is due to the fact that $\left(X^t Z X\right)^{-1} \succeq 0$. Thus, the objective function $f(z)$ is a concave function on the decision variable z. Thus, the LTS problem is in fact a concave minimization problem. The concavity property is the part that makes the LTS problem very hard to solve in general. In the

next section, we will show how we resolve this difficulty by considering the "almost complementary" problem of the LTS. We call this the "Maximum Trimmed Squares" problem since this problem aims to find the worst subset of observations instead of trying to find the best subset as the LTS does.

3.3 The Maximum Trimmed Squares (MTS) Problem

We notice that the objective function in the LTS problem is concave and hence is very hard to solve in general. We also notice that the outlier set would be the set with large total squared residuals. Hence, instead of trying to find the non-outlier set, we could try to find the outlier set. We could get around this concavity problem by trying to maximize this concave objective.

The least trimmed squares regression problem with k good observations and the maximum trimmed squares regression problem with q bad observations are formulated as follows:

LTS		**MTS**	
$\min_z$	$f(z)$	$\max_z$	$f(z)$
$s.t.$	$z^t\boldsymbol{e} = k$	$s.t.$	$z^t\boldsymbol{e} = q$
	$0 \leq z_i \leq 1$		$0 \leq z_i \leq 1$

Figure 3-2 illustrates the idea of using maximum trimmed squares. In this figure, 100 points were generated randomly with the first 90 blue dotted points following a linear model. The least squares regression line corresponding to the blue dotted points (the blue solid line) fits those 90 points very well. The last 10 red crossed points were generated to be far above and below the blue solid regression line.

For any 3 points from the entire data set, there is a corresponding least squares fit, 3 residuals, and a sum of total squared residuals. The maximum trimmed squares problem with $q = 3$ aims to find the worst 3 points whose sum of squared residuals is maximized. The circled points are the 3 optimal points found by the MTS problem. The red dashed line is their corresponding least squares regression line. We observe that all these 3 points are outliers. This example shows MTS did a great job in detecting outliers. Figures 3-3 and 3-4 show the output of the MTS problem for $q = 8$ and $q = 11$ respectively. We can see that, in each cases, the MTS can identify many outliers.

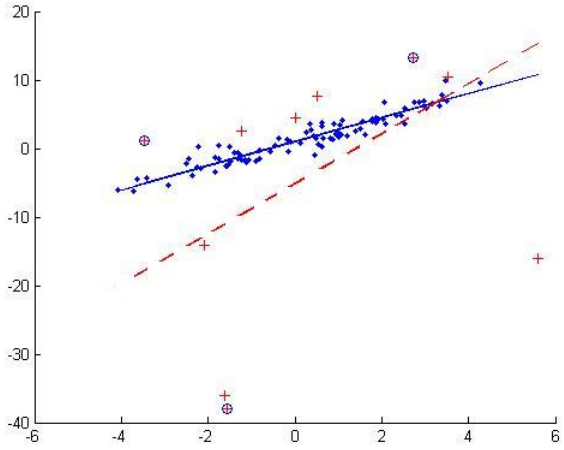


Figure 3-2: An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst three points. In this figure, 100 points were generated by using the two sided contamination model with 10 outliers. The blue solid line is the LTS regression line that corresponds to the 90 good observations. The circled points are those detected when running maximum trimmed squares with $q = 3$ worst points. The red dashed line corresponds to the regression line of these q worst points.

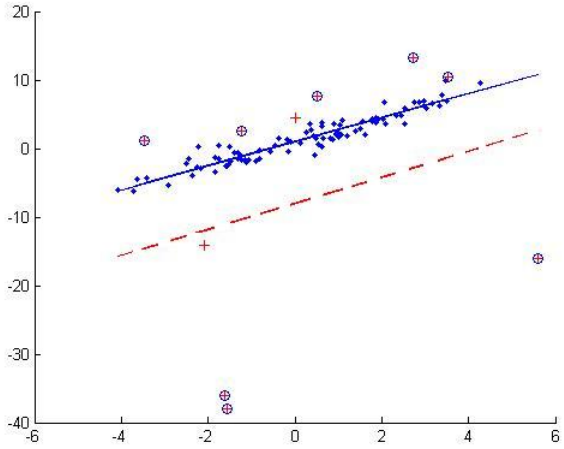


Figure 3-3: An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst eight points. In this figure, 100 points were generated by using the two sided contamination model with 10 outliers. The blue solid line is the LTS regression line that corresponds to the 90 good observations. The circled points are those detected when running maximum trimmed squares with $q = 8$ worst points. The red dashed line corresponds to the regression line of these q worst points.

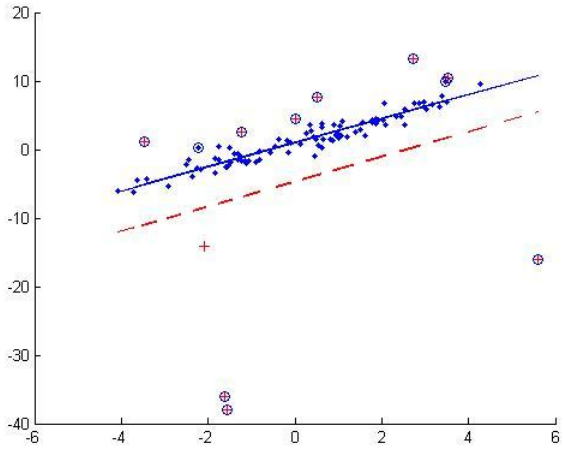


Figure 3-4: An example demonstrating the Maximum Trimmed Squares idea and performance when we want to identify the worst 11 points. In this figure, 100 points were generated by using the two sided contamination model with 10 outliers. The blue solid line is the LTS regression line that corresponds to the 90 good observations. The circled points are those detected when running maximum trimmed squares with $q = 11$ worst points. The red dashed line corresponds to the regression line of these q worst points.

3.3.1 MTS Reformulation as a Semi-definite Programming Problem

In this Subsection, we will show that the MTS problem can be reformulated as a semi-definite programming problem and hence it can be solved efficiently using interior point methods (see [42]). The MTS problem is:

$$\begin{aligned} \max_{z} \quad & \underbrace{y^t D(z) y - y^t D(z) X \left(X^t D(z) X\right)^{-1} X^t D(z) y}_{s} \\ s.t. \quad & z^t \boldsymbol{e} = q \\ & 0 \leq z_i \leq 1 \end{aligned}$$

This optimization problem is equivalent to:

$$\begin{aligned} \max_{z,\ s} \quad & s \\ s.t. \quad & s \leq y^t D(z) y - y^t D(z) X \left(X^t D(z) X\right)^{-1} X^t D(z) y \\ & z^t \boldsymbol{e} = q \\ & 0 \leq z_i \leq 1. \end{aligned}$$

The constraint $s \leq y^t D(z) y - y^t D(z) X \left(X^t D(z) X\right)^{-1} X^t D(z) y$ looks highly non-linear in variable z. However, we can use the Schur complements (see [42]) to rewrite it in the form of a semi-definite programming constraint,

$$\begin{aligned} & s \leq y^t D(z) y - y^t D(z) X \left(X^t D(z) X\right)^{-1} X^t D(z) y \\ \Leftrightarrow \quad & \begin{bmatrix} y^t D(z) y - s & y^t D(z) X \\ X^t D(z) y & X^t D(z) X \end{bmatrix} \succeq 0 \\ \Leftrightarrow \quad & -sE + \sum_{i=1}^{n} z_i \begin{bmatrix} y_i^2 & y_i x_i \\ y_i x_i^t & x_i^t x_i \end{bmatrix} \succeq 0 \\ \Leftrightarrow \quad & -sE + \sum_{i=1}^{n} z_i V_i \succeq 0 \end{aligned}$$

where E is a matrix of size $(p+1) \times (p+1)$ with all elements equal to zero except the first row first column element to be equal to one and $V_i = \begin{bmatrix} y_i^2 & y_i x_i \\ y_i x_i^t & x_i^t x_i \end{bmatrix} = \begin{bmatrix} y_i \\ x_i^t \end{bmatrix} \begin{bmatrix} y_i & x_i \end{bmatrix}$ are given. We have transformed the nonlinear constraint into a semi-definite type constraint. The optimization problem thus becomes:

Model 6 : Maximum Trimmed Squares Optimization Model for Robust Regression

$$\begin{aligned} \max_{z,s} \quad & s \\ s.t. \quad & -sE + \sum_{i=1}^{n} z_i V_i \succeq 0 \\ & z^t \boldsymbol{e} = q \\ & 0 \leq z_i \leq 1 \end{aligned}$$

3.3.2 Performance Guarantee

The "maximum trimmed squares" problem possesses desirable properties. First, it is equivalent to a semi-definite programming problem and can be solved efficiently using interior point methods. Second, the way the MTS identifies outliers is quite intuitive, as shown pictorially.

However, there are still some theoretical issues we need to address. The first issue is whether the MTS problem will produce integer solutions z_j and what we should do if it doesn't. The second issue is that the MTS problem is not totally complementary to the LTS, i.e. the worst set found by the MTS is not a complement set of the good set that the LTS is supposed to find. This non-complementary property should be the case since a hard problem like the LTS cannot be solved by simply complementing the result from an easy problem like the MTS.

Theorem 5 will show that there will be at most $(p+1)$ non-integer z_j and hence the first issue can be resolved if we allow the inclusion of these non-integer decision variables to be in the worst list. Theorem 6 will show that in the worst case, when

outliers are distributed in some specific directions or in some adverse locations, the MTS problem can identify at least one outlier at a time. Hence, no mater how adversely the outliers are located, we can perform an iterative procedure to remove outliers one at a time.

Theorem 5. *At most* $(p+1)$ *decision variables* z_i *are non-integer in the MTS problem.*

Proof sketch. The detail proof is in the appendix. We only show the main idea here. We apply the Karush-Kuhn-Tucker [23] conditions and complementary slackness to the MTS problem and show that those non-integer solutions z_j correspond to a system of equations with $(p+1)$ degrees of freedom. Hence the number of equations must not exceed $(p+1)$, in other words, the number of non-integer decision variables is at most $(p+1)$. □

Definition 1. *For any* k*, we define* $B(k) = LTS(X, y, k)$ *to be the least trimmed squares solution. For any set of observations* S*, we define* $w(S)$ *to be the smallest width between two parallel planes that contain all the observations* S*.*

Assumption 1. *The set of good observations* G *and the set of outliers* X *are distinguishable from each other in the following "strong robust sense":*
a. $G \equiv LTS(X, y, k)$ *where* $k = |G|$ *is the cardinality of the set* G*.*
b. For any set of observations S *with cardinality* $(k+1)$*, then* $w(S) \geq w(G)\left(1 + \sqrt{\frac{p+1}{2}}\right)$.

Theorem 6. *The MTS problem would identify at least 1 outlier under Assumption 1.*

The detail proof for this theorem is in the Appendix. The intuition here is that, if this result does not hold, then good observations need to be sufficiently close to each other such that the sum of squared residuals of any subset of these good observations together with an outlier would be small enough, and hence no outlier would be identified in the optimal MTS solution. The assumption on the separability between the good observations is intuitive. The scale factor of $\sqrt{\frac{p+1}{2}}$ separability seems to be high for large p. However, this factor is only for a guaranteed theoretical result. In most cases in practices, these assumptions don't need to hold in order for the theorem to be valid.

3.3.3 Robust Regression Algorithm using MTS

So far, we have noticed that we can remove at least one outlier by solving the MTS problem, which can be done very efficiently through interior point methods. However, since the MTS is not complementary to the LTS, the MTS with a budget of $(q = n-k)$ might not be able to remove all $(n-k)$ outliers at a time. Instead, it should remove at least one outlier and other good observations at the same time. The percentage of outliers and good observations removed by the MTS depends on how the outliers are located. In the case when outliers are distributed randomly around the good observations, the MTS is expected to find the majority of outliers. In the worst case when outliers are distributed adversely in one particular direction or location, it is possible that only a small fraction of outliers are removed at a time. Hence, we might have to run the MTS iteratively several times before we are able to remove all outliers. The choice of the number of times to run the MTS and the budgets for each run should depend on how adversely the outliers are placed.

Input:	Data $X \in \mathcal{R}^{n\times p}$, $y \in \mathcal{R}^n$.
Optional Input:	The number of outliers k and the number of times M to run the MTS. If k is not given, we set $k = 0.1 * n$ which means we expect about 10% of the observations are outliers. If M is not given, we set $M = 4$ by default. If we know the outliers are randomly distributed, we should set M small. If we know outliers placed in a nearly worst case manner, we should set $M = k$.
Output:	The robust regressor β_{rob}.

Algorithm:	**1**. Set $q = \lceil \frac{k}{M} \rceil$.
	2. For $j = 1$ to M {
	Let: $W = \text{MTS}(X, y, q)$ be the outliers detected and
	$(X, y) = (X, y) \backslash W$ be the remaining set.
	}
	3. Find the least squares coefficients β_0 for the remaining set.
	4. Perform the iterative algorithm shown in Table 3.1 until β_0 converges to β_{rob}.

Table 3.2: Robust regression algorithm via Maximum Trimmed Squares

3.4 Numerical Results

3.4.1 Classical Examples

Figure 3-5 shows the results for three classical examples presented in the Rousseeuw and Leroy book [35] pages 24, 25 and 28 respectively. The dashed red lines are the least squares fits using ordinary least squares and the blue solid lines are our robust regression fits. Our robust regression fits is exactly the same with the LTS fits in these cases. We can see that our robust regression lines fit the majority of the data much better than the least squares lines. In the first example, the contaminated observation is created on purpose to demonstrate the difference between least squares and our technique. The second example is a real example where the number of international phone calls (in 10 millions) were collected for consecutive years. The figure suggests an increasing trend. However, the observations for the years 1964-1970 seem to be off the trend of the majority. It turns out that these observations were recorded in minutes rather than the number of phone calls. This example shows that our robust technique can help detecting such human errors. The third example is a study of the relationship between the stars' temperatures and their light intensities. The robust estimator on the majority helps figuring out an underlying phenomenon in astronomy where a new model is needed to distinguish the four giant stars with the highest temperatures from the rest.

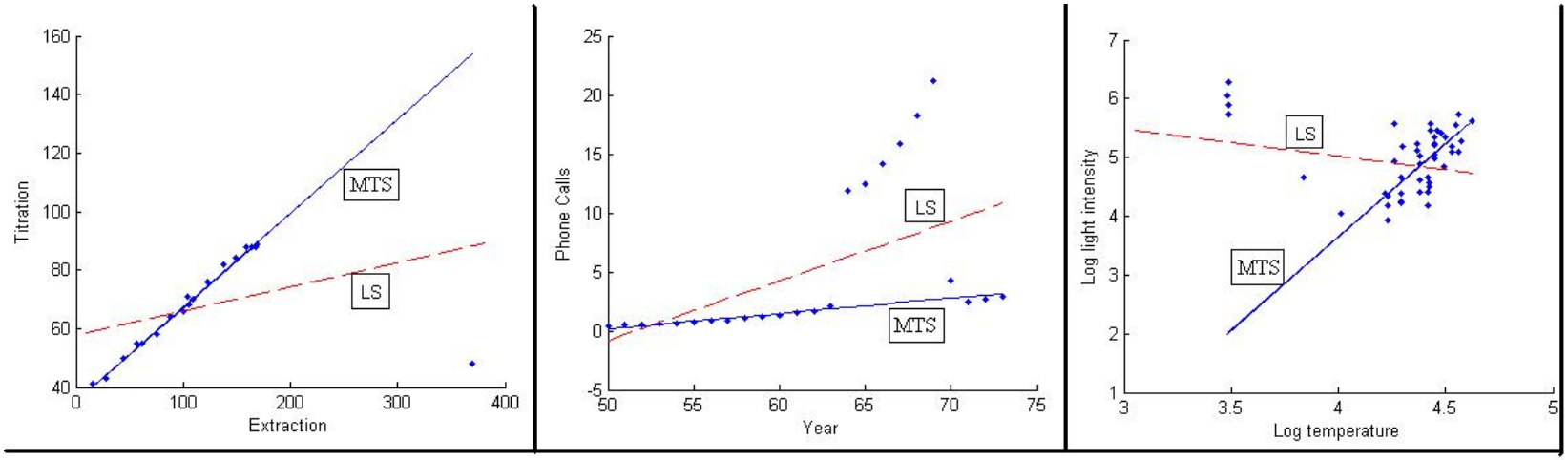


Figure 3-5: Three classical examples demonstrating the performance of our robust regression technique. These examples are taken from Rousseeuw and Leroy's book "Robust Regression and Outlier Detection" pages 24, 25, 28 respectively. The red dashed lines are the least squares lines which are pulled toward the outliers. The blue solid lines are our robust regression lines which coincide with the LTS lines for these examples.

3.4.2 Examples from Different Contamination Models

We use the following two contamination models. In the **One Sided Contamination Model** (OSCM), we generate $X \in \mathbb{R}^{n \times p}$ randomly and $\beta \in R^p$ randomly. We set the first k error terms $\epsilon_j,\ j \in [1, .., k]$ as independent standard normal random variables. We set the last $n - k$ error terms as independent chi squared random variables with 5 degrees of freedom. We then set $y = X\beta + \epsilon$.

In the **Two Sided Contamination Model** (TSCM), we generate $X \in \mathbb{R}^{n \times p}$ randomly and $\beta \in R^p$ randomly. We set the first k error terms $\epsilon_j,\ j \in [1, .., k]$ as independent standard normal random variables. We set the last $n - k$ error terms as independent chi squared random variables with 5 degree of freedom. We then set the sign of these $n - k$ variables randomly such that the outliers lie on both sides of the true regression line. Finally, we set $y = X\beta + \epsilon$.

In Figure 3-6, we show our robust fits in blue color and the corresponding least squares fits in red color. The red points are contaminated observations while the blue points are good observations. The circled points are those that our method identifies as outliers. It is clear that our robust estimators fit the majority of the data much better than the least squares estimators do.

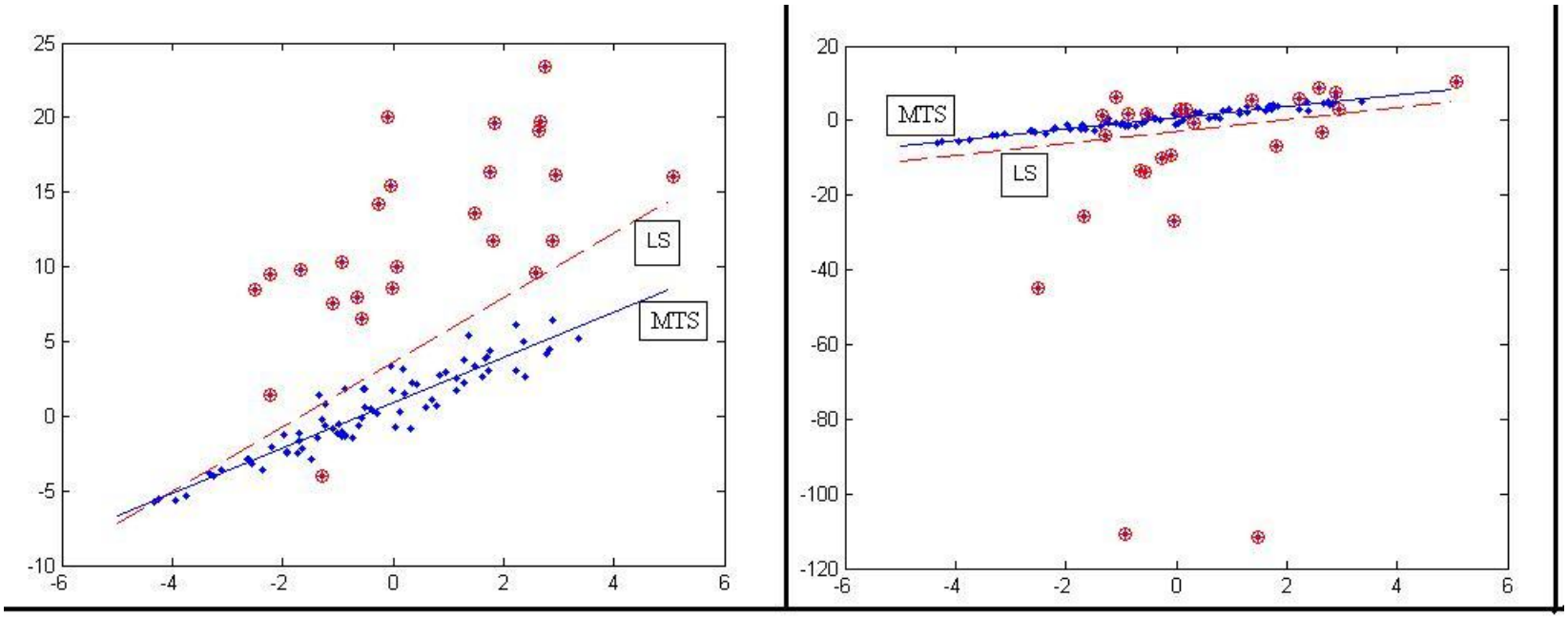


Figure 3-6: Two examples demonstrating the performance of our robust regression technique in which data are generated from the One Sided Contamination Model (shown on the left) and the Two Sided Contamination Model (shown on the right). The good observations are the blue dotted points. The contaminated observations are the red crossed points. The red dashed lines are the least squares lines which are pulled toward the outliers. The blue solid lines are our robust regression lines which coincide with the LTS lines for these examples. The circled points are those outliers detected by our algorithm.

3.4.3 Extreme Case Example

In Figure 3-7, we show an extreme example where 10 outliers are located in extreme locations such that the least squares estimator is totally distorted. The red points (10 of them) are contaminated observations while the blue points (200 of them) are good observations. Our robust fit is the blue solid line and the corresponding least squares fit is the red dashed line. The circled points are those outliers identified by our method. We can see that our robust regression line fits the majority of the data very well.

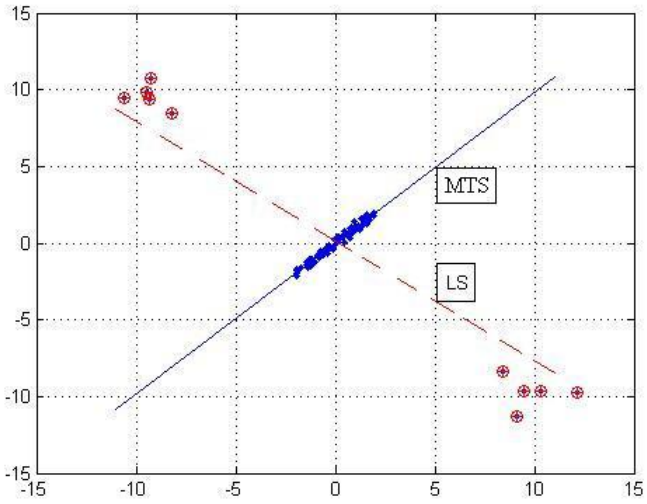


Figure 3-7: An example to demonstrate our robust regression technique can identify outliers no mater how extremely they are located. This example shows an extreme case with 10 outliers located in extreme locations that pull the least squares regression line close to themselves. The good observations are the blue dotted points. The contaminated observations are the red crossed points. The blue solid line is our robust fit which coincide with the LTS fit. The circled points are those outliers detected by our algorithm.

3.4.4 Robust Regression Accuracy

We check the accuracy of our robust estimators with the number of observations n ranging from 100 to 5000 and the problem dimension p ranging from 2 to 50. For each problem size, we create 10 sample paths. Each sample path corresponds to a

data matrix of size $n \times p$ generated from the TSCM model. For each sample path, we run our robust estimation algorithm to find out those k observations with smallest trimmed distances. The robust fit is the corresponding least squares fit applied to those k observations. Once we have found an estimator $\hat{\beta}$, we find the corresponding distances $d_j = y_j - x_j\hat{\beta}$. The objective value corresponding to $\hat{\beta}$ is the norm of k smallest squared distances.

$$\bar{d} \;=\; \sqrt{\frac{\sum_{j=1}^{n} z_j d_j^2}{k}}$$

where z_j equal to 1 if j is among those k observations with smallest d_j^2.

Figure 3.3 shows the average trimmed distances using our robust fit, compared to the true fit(those that are found by least squares regressors on the non-contaminated subset) and the least squares fit. We found that the average trimmed distances from our regressors is around that of the true regressors and much smaller than that of the least squares regressors. It is also interesting to see some instances where our robust fit produces smaller trimmed distances than the true fit. This occurs because our contamination model can produce observations with smaller distances than some of the non-contaminated observations (i.e. when the chi squared random variable with 5 degree of freedom is smaller than a standard normal variable).

3.4.5 Computational Time

Figure 3.4 shows the average times taken by our algorithm. The algorithm took less than 1 hour for all problems with dimension $p \leq 20$ or $n \leq 2000$. The longest instance was for $n = 5000$ and $p = 50$ which took around 2.5 hours. Although 2.5 hours is quite long for that case, the computational time of our algorithm scales well when we increase n and p and our robust regression method is a polynomial time algorithm since the MTS problem is equivalent to a semi-definite programming problem and can be solve efficiently using interior point methods (see [42]). This polynomial complexity property is a significant improvement compared to existing methods like the Fast-

	100	**200**	**500**	**1000**	**2000**	**5000**
2	0.76 [0.77, 4.71]	0.70 [0.71, 3.07]	0.73 [0.73, 3.30]	0.74 [0.74, 3.36]	0.74 [0.74, 3.62]	0.73 [0.73, 3.74]
5	0.73 [0.75, 4.30]	0.7164 [0.73, 3.50]	0.7173 [0.72, 3.78]	0.72 [0.73, 3.67]	0.72 [0.73, 3.75]	0.74 [0.74, 3.41]
10	0.67 [0.72, 3.34]	0.72 [0.74, 3.55]	0.71 [0.73, 3.54]	0.72 [0.73, 3.55]	0.74 [0.75, 3.36]	0.74 [0.74, 3.42]
20	0.57 [0.65, 4.13]	0.66 [0.70, 3.93]	0.70 [0.72, 4.33]	0.73 [0.74, 3.52]	0.72 [0.73, 3.30]	0.73 [0.73, 3.54]
30			0.68 [0.71, 4.14]	0.71 [0.73, 3.77]	0.71 [0.73, 3.36]	0.72 [0.73, 3.46]
50				0.70 [0.73, 3.95]	0.70 [0.72, 3.51]	0.73 [0.74, 3.45]

Table 3.3: Average trimmed absolute residuals for different problem sizes under the Two Sided Contaminated Model. The top row is for the number of observations n while the first column is for the problem dimensions p. The numbers in the table show the average trimmed distances $\bar{d} = \sqrt{\frac{\sum_{j=1}^{n} z_j d_j^2}{k}}$ where $d_j = y_j - x_j\beta$ and z_j are the indicator variables for whether observation j is among those k observations with smallest squared distances d_j^2. The ranges below these numbers correspond to the best (i.e. if we know all the contaminated observations) and the worst distances (i.e. if we use least squares regression).

LTS or mixed integer programming formulations for large scaled problems. It is a shortcoming for our research not to include a comparison between our method with the Fast-LTS. The reason is that Matlab can only handle cases where the number of observations n is less than $20,000$ due to Matlab's limitation in memory allocation.

3.5 Future Research

For future research, we plan to implement our method in other languages such as R or SAS to avoid the memory limitation of Mallab so that large scale simulations can be done. In addition, currently we use SDPT3 to solve the semi-definite programming problem. We plan to implement our own interior point methods to exploit the special structure of the problem.

	100	**200**	**500**	**1000**	**2000**	**5000**
2	1.87	2.50	10.34	35.1	245.1	1645.5
5	1.77	3.25	13.64	42.1	231.9	2459.8
10	1.86	4.74	20.28	66.0	428.4	2910.8
20	4.22	9.30	39.78	118.1	600.6	3530.1
30			64.52	214.2	959.0	4960.2
50				494.9	1943.6	9211.2

Table 3.4: Average computational time for different problem sizes under the fully contaminated model. The top row is for the number of observations n while the first column is for the problem dimensions p. The numbers in the table show the computational time in seconds.

3.6 Conclusion

We have presented a method for robust regression using the idea of "maximum trimmed squares" where the worst subset of observations with the largest sum of squared residuals are found by solving a semi-definite programming problem. The maximum trimmed squares problem is formulated to resolve the difficulty in the concave objective function of the least trimmed squares regression problem. The maximum trimmed squares problem can identify many outliers in a single call if these outliers are distributed randomly around the good observations. In the worst case when outliers are located adversely in one particular direction or location, we can iteratively solve the maximum trimmed squares problem where, at each step, at least one outlier is removed. We have tested our method with different data sets and found our method produced fast and highly accurate results.

Chapter 4

Robust Ranking and Portfolio Optimization

4.1 Introduction

Ranking problems are common in real life, where there is a comparison and an evaluation process involved, for example, in sports and in asset selection. A ranking between two objects simply provides information about which object is preferable among the two. In portfolio construction, managers often have preferences on the assets they are investing. They might prefer one stock (or one sector of stocks) more than other stocks (or other sectors). The ranking on the assets could arrive from individual preferences or from empirical evidence. There is an extensive literature on accounting anomalies such as the post earning announcement drift [3] and Fama-French risk factors [17] that can be used to select stocks and to obtain their ranking. The main advantage of using ranking over the traditional mean-variance portfolio construction is in avoiding the estimation of the expected returns, which is a very challenging task. Although the idea of using ranking is quite intuitive and it is easy to build trading strategies from a given ranking, the ranking itself needs to be estimated and there is uncertainty associated with the estimation. Hence, we often see the ranking expressed in the form of confidence intervals. For example, it is more natural for a manager to believe an asset to be in the top five (ten) or in the bottom five (ten) rather than

to claim it to be the best (or the worst) one. In addition, ranking could arrive from many different sources (e.g. each analyst could comes up with a ranking) and there is always uncertainty associated with ranking. This motivates us to develop a robust ranking model where we want to find the optimal policy (e.g. the portfolio weights) to maximize some objective function even for the worst realization of the ranking within the uncertainty set. This robust ranking model is a minimax mixed integer programming problem and is often very hard to solve. We apply the column generation method where constraints are efficiently generated using a network flow model. We show that the robust ranking model can be solved efficiently through numerical results. We also apply the robust ranking model to portfolio optimization where the ranking uncertainty set is obtained from post announcement earning drifts. In the next Subsection, we will review the existing literature related to our robust ranking model and portfolio optimization.

4.1.1 Literature Review

The traditional Markowitz mean-variance portfolio optimization framework [27] aims to find the optimal portfolio to maximize some objective functions whose input parameters including the mean and the covariance matrix. Many researchers include Michaud [30] and Chorpa and Ziemba [15] have shown the poor out-of-sample performance of the mean-variance optimal portfolio due to estimation errors on the expected return. Due to this drawback, many practitioners have used nonparametric ranking methods for portfolio optimization. The main advantage of using ranking is to avoid the difficult task of estimating the expected return. Almgren and Chriss [1] use assets' ranking instead of assets' returns to model the portfolio optimization problem. Although the ranking that Almgen and Chriss used was easy to obtain, it is still prone to estimation error. We present a robust ranking model in which the ranking is allowed to be in an uncertainty set. This research is along the line of the robust optimization approaches which are very prevalent. The earliest work on robust optimization was Soyster [38] in 1973. The key for an attractive robust model is to ensure the construction of the uncertainty set to be intuitive and capture well

the uncertainty nature of the parameters while the problem is still computationally tractable. Ben-tal and Nemirovski [4] and Bertsimas and Sim [6] presented a series of papers on robust optimization with different frameworks for controlling the conservative levels for different problem structures. Since then, the robust optimization frameworks have been applied to many different applications such as in inventory control and in dynamic pricing (see [7], [32] and [33] for examples). For the most updated theory and development of the field, see Beyer and Sendhoff comprehensive review in [9].

In the field of robust portfolio optimization, Goldfarb and Iyengar [18] and Bienstock [10] develop different frameworks for the uncertainty sets. Goldfarb and Iyengar use a factor model for the assets' returns and then the uncertainty arrives naturally from the estimation of the parameters of the factor model. This construction leads to a first norm uncertainty set for the returns and a second norm uncertainty set for the covariance matrix. Finally, they transform the robust model to a SOCP problem which is computationally tractable.

Bienstock, on the other hand, constructs the uncertainty set around the level of deviations of the assets' returns from their expected values. These deviations are both in terms of the number of "misbehavior" assets as well as their total returns. This model becomes a mixed integer optimization problem. Bienstock then uses a decoupling technique to solve it efficiently.

Our robust model is different from all the aforementioned methods due to the difference in the parameter space of the problem. Since the parameter in the ranking problem is the ranking itself, which is discrete, the uncertainty set in our case is discrete. This combinatorial characteristic makes the problem very difficult to solve in general. We exploit the nice property of the ranking and apply the column generation method to solve the robust ranking problem. The constraint generation process is done through solving a transportation problem. Our method of controlling the conservative level is also different from the earlier works. We incorporate the deviation of the rankings from a nominal ranking into the objective function itself. This method in effect creates a non-uniform uncertainty set where the rankings that are further

from the nominal ranking are penalized more than those that are closer.

4.1.2 The Robust Ranking Problem

Consider n objects with a ranking $\mathbf{R} = (R_1, R_2, \ldots, R_n)$, which is a permutation of the set $\{1, 2, \ldots, n\}$. For example, in the case $R_i = i,\ \forall\ i \in \{1, 2, \ldots, n\}$, object i is ranked i^{th} in the set. We want to find a weight vector $\boldsymbol{\omega}$ that belongs to some generic constraint set $\mathcal{P}(\boldsymbol{\omega})$ to maximize some generic objective function $f(\boldsymbol{\omega}, \mathbf{R})$. An example of the constraint is $\mathcal{P}(\boldsymbol{\omega}) = \{\boldsymbol{\omega} \mid \boldsymbol{\omega}^t \boldsymbol{e} = 1\}$ to make sure the weights are normalized. If the ranking $\mathbf{R}$ is known, the problem becomes:

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & f(\boldsymbol{\omega}, \mathbf{R}) \\ s.t. \quad & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

In this case, the optimization problem can be solved easily for many classes of objective function f and constraint set $\mathcal{P}$. However, in many situations, the ranking is unknown. Instead, we assume that we only know the ranking belongs to some known polytope of the form $\boldsymbol{\mathcal{U}}(\mathbf{R}) = \left\{\mathbf{R} \in \Pi(1, 2, \ldots, n) \mid R_i \in S_i,\ \forall\ i \in \{1, 2, \ldots, n\}\right\}$, where $\Pi(1, 2, \ldots, n)$ is the set of all the permutations of the set $(1, 2, \ldots, n)$ and S_i are subsets of the set $\{1, 2, \ldots, n\}$. For example, S_i could be confidence intervals $[l_i, u_i]$ such that $l_i \leq R_i \leq u_i$. The robust ranking problem aims to find the weight vector $\boldsymbol{\omega}$ to maximize some objective function $f(\boldsymbol{\omega}, \mathbf{R})$ for the worst realization of the ranking $\mathbf{R}$:

Model 7 : General Robust Ranking Model

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R})} \quad f(\boldsymbol{\omega}, \mathbf{R})\right] \\ s.t. \quad & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

This problem is very challenging to solve in general because the set $\boldsymbol{\mathcal{U}}(\mathbf{R})$ is typically a very large discrete set. Even if we know the weight vector, solving for the worst realization of the ranking would be a combinatorial problem and could be

very difficult. We will show that, for a large class of the objective function f and the constraint set $\mathcal{P}$, we can solve this robust ranking problem efficiently by using the column generation algorithm.

4.1.3 Chapter Structure

This Chapter aims to solve the robust ranking problem in general. We will provide the general conditions of the objective function and the constraint set for the algorithm to work. However, since the computation performance depends on the specific objective function $f(\boldsymbol{\omega}, \mathbf{R})$, we will provide the algorithm for the general objective function but demonstrate its usage, its intuitions, as well as its performance through the specific objective functions in the robust portfolio optimization problem. We will show how to apply the column generation method to solve the robust ranking problem in Subsection 4.2.1. This method iteratively solves the relaxed robust ranking problem with a smaller ranking uncertainty set and then adds the worst possible ranking corresponding to the relaxed optimal weights to the relaxed uncertainty set. The algorithm stops when the relaxed uncertainty set is stable.

For this algorithm to work, we need to find efficient methods for generating additional constraints and to solve the relaxed problem. The constraint generation problem is analyzed in Subsection 4.2.2 and the relaxed problem is analyzed in Subsection 4.2.3. We extend our robust ranking model to control the degree of conservatism in Subsection 4.2.4 and to allow group ranking in Subsection 4.2.5. We show how to apply our robust ranking model to portfolio optimization in Section 3.

We present numerical results in Section 4.4 with the overall computation time and the number of iterations in Subsection 4.4.1. In Subsection 4.4.2, we present an empirical study where we apply our robust ranking model to the universe of stocks in the Dow Jones Industrial Average Index. The ranking uncertainty set is obtained from the post earning announcement drift as shown in Subsection 4.4.2. We show the procedure for running an out-of-sample test in Subsection 4.4.2 and the empirical results in Subsection 4.4.2. Finally we conclude our Chapter in Section 4.5.

4.1.4 Contributions

- We develop a generic robust ranking model and use the column generation method to solve it. This process involves the transformation of the constraint generation problem, which is a combinatorial problem, into a network flow problem such that constraints can be generated efficiently.
- We combine results from post announcement earning drifts and our ranking model to portfolio optimization and perform empirical tests. We also provide computational results to demonstrate the algorithm performance.

4.2 Solving the Robust Ranking Problem

4.2.1 The Column Generation Algorithm

Let $\delta(\boldsymbol{\omega}) = \min_{\mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R})} f(\boldsymbol{\omega}, \mathbf{R})$. The robust ranking model 7 can be rewritten as follows:

$$\begin{aligned} \max_{\boldsymbol{\omega}, \delta} \quad & \delta \\ s.t. \quad & \delta \leq f(\boldsymbol{\omega}, \mathbf{R}) \quad \forall\, \mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R}) \\ & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

This new model has a simpler objective function compared to the original robust ranking formulation. However, the set of constraints is much larger. This reformulated model motivates us to apply the constraint generation method. In this method, we relax the robust ranking problem with a smaller set of constraints and solve for the relaxed optimal solution. We then check if all the constraints are satisfied. This is done by solving a subproblem of finding the worst realization of the ranking that corresponds to the relaxed optimal solution and check if the worst ranking is already in the relaxed ranking list. If not, we add the violating constraints to the relaxed constraint set. This process is done until no more constraints are violated. The formal procedure of the constraint generation method is shown in Algorithm 1:

Algorithm 1 : Column Generation Method for The Robust Ranking Problem

- **Initialization step**:
  Find initial weight vector $\boldsymbol{\omega}^{(0)}$, set the relaxed uncertainty set $\boldsymbol{\mathcal{U}r} \equiv \emptyset$ and set $k = 0$.

- **Iterative steps**:
  a. Solve the **constraint generation problem**:

$$\mathbf{R}^{worst} = \underset{\mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R})}{\operatorname{argmin}} \left[f(\boldsymbol{\omega}^{(k)}, \mathbf{R}) \right].$$

  b. If $\mathbf{R}^{worst} \in \boldsymbol{\mathcal{U}r}$, terminate the algorithm. Otherwise, add $\mathbf{R}^{worst}$ to the relaxed uncertainty set $\boldsymbol{\mathcal{U}r}$.
  c. Set $k = k + 1$ and solve the **relaxed problem**:

$$\boldsymbol{\omega}^{k} = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \left[\min_{\mathbf{R} \in \boldsymbol{\mathcal{U}r}} f(\boldsymbol{\omega}, \mathbf{R}) \right]$$

  d. Go back to step **a**.

In order to apply the constraint generation method, we need to make sure that we can solve the constraint generation problem and the relaxed problem efficiently. In the next two Subsections, we will show the general conditions for these two problems to be computationally tractable.

4.2.2 Solving the Constraint Generation Problem

The constraint generation problem is often difficult to solve because of its combinatorial structure. Nevertheless, it is very interesting to notice that for a large class of objective function $f(\boldsymbol{\omega}, \mathbf{R})$, which is separable in $\mathbf{R}$, we can transform the constraint generation problem into a transportation problem. Let $f(\boldsymbol{\omega}, \mathbf{R}) = \sum_{j=1}^{n} g_j(\boldsymbol{\omega}, R_j)$, then the cost between source i and sink j in the transportation problem is $g_j(\boldsymbol{\omega}, j)$ if $j \in S_i$, where S_i is the list of possible ranks that object i can have, and is infinity otherwise.

The transportation network has n source and n sink nodes with a flow of $+1$ in each source node and -1 in each sink node. Figure 4-1 shows an example of 3 assets

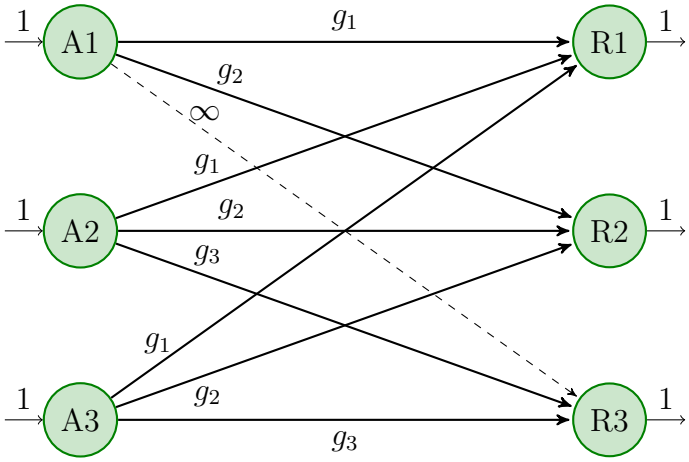


Figure 4-1: The constraint generation problem is reformulated as a transportation problem. The path connecting source node i to sink node j implies that asset i is assigned to rank j. Suppose the uncertainty set is: $\boldsymbol{\mathcal{U}}(\mathbf{R}) = \{\mathbf{R} \in \Pi(1,2,3) \mid R_1 \in \{1,2\},\ R_2 \in \{1,2,3\},\ R_3 \in \{1,2,3\}\}$. This means asset 1 has the first or second ranking while assets 2 and 3 have all the possible ranking. Since asset 1 cannot be in the third rank, the cost of the flow from asset 1 to rank 3 is set to infinite. The total cost of the network flow problem is equal to the objective function $f(\boldsymbol{\omega}, \mathbf{R})$.

with the ranking uncertainty of:

$$\boldsymbol{\mathcal{U}}(\mathbf{R}) = \{\mathbf{R} \in \Pi(1,2,3) \mid R_1 \in \{1,2\},\ R_2 \in \{1,2,3\},\ R_3 \in \{1,2,3\}\}\,.$$

A path from source i to sink j implies that object i has the actual rank j. The cost between source i and sink j is $g_j(\omega, r_j)$ for $j \in S_i$ and is infinite otherwise. We want to find the minimum cost of this transportation network. The optimal solution of this problem is always a binary solution. We can use the auction algorithm or the network simplex algorithm to solve this problem efficiently (see [8], for example, for detail).

Remark: In order to transform the constraint generation problem to a network assignment model, the uncertainty set must have the permutation property. That is an object is assigned to one and only one rank. We extend this restriction for the case of group ranking where many objects can obtain the same rank in Subsection 4.2.5.

4.2.3 Solving the Relaxed Problem

The relaxed problem is equivalent to:

$$\begin{aligned} \max_{\boldsymbol{\omega},\delta} \quad & \delta \\ s.t. \quad & \delta \leq f(\boldsymbol{\omega}, \mathbf{R}), \quad \forall\, \mathbf{R} \in \boldsymbol{\mathcal{U}r} \\ & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

For a small set $\boldsymbol{\mathcal{U}r}$, this problem is computationally tractable for many classes of objective function $f(\boldsymbol{\omega}, \mathbf{R})$. For example, if $f(\boldsymbol{\omega}, \mathbf{R})$ is a concave function and $\mathcal{P}(\boldsymbol{\omega})$ is a convex set in $\boldsymbol{\omega}$, the relaxed problem is a convex optimization problem. Notice that even when the objective function is non-convex, it might be still possible to transform the relaxed problem into a convex optimization problem. For example, we will show in Subsection 4.3.2 that, although the Sharpe ratio objective function in the robust portfolio optimization problem is non-convex, we can exploit its homogeneous property and transform the relaxed problem into a quadratic programming problem which can be solved efficiently.

4.2.4 Extension to Control Conservative Levels

One of the main criticisms for robust optimization is its over-conservatism at its optimal solution. The robust optimal solution usually corresponds to the cases where the parameters take extreme values in their boundary instead of corresponding to some nominal ranking $\bar{\mathbf{R}}$ that we expect most. We propose a model where we incorporate the deviations of the parameters from their nominal values into the objective function itself. This protects the regions of parameters that are closer to the nominal value more than those that are further.

Here, $\gamma \geq 0$ and $\gamma|\mathbf{R} - \bar{\mathbf{R}}|$ represents the belief about how far the actual ranking could go from the nominal ranking. If γ is large, we have a strong belief that the actual ranking will be around the nominal ranking and we want the robust model to protect that region more. If γ is small, then the actual ranking could be anywhere in

Model 8 : Robust Max-Sharpe Portfolio Optimization Model with Non-Uniform Uncertainty Set

$$\begin{array}{ll} \max\limits_{\boldsymbol{\omega}} & \left[\min\limits_{\mathbf{R}\in\boldsymbol{\mathcal{U}}(\mathbf{R})} \quad \left[f(\boldsymbol{\omega},\mathbf{R})+\gamma|\mathbf{R}-\bar{\mathbf{R}}|_{L1}\right]\right] \\ s.t. & \boldsymbol{\omega}\in\mathcal{P}(\boldsymbol{\omega}) \end{array}$$

the uncertainty set. In other words, γ can be viewed as the 'concentration' parameter to represent how 'dense' the uncertainty set $\boldsymbol{\mathcal{U}}(\mathbf{R})$ is surrounding the nominal ranking $\bar{\mathbf{R}}$. The new objective function has been changed to:

$$f'(\boldsymbol{\omega},\mathbf{R}) = f(\boldsymbol{\omega},\mathbf{R})+\gamma|\mathbf{R}-\bar{\mathbf{R}}|_{L1} = \sum_{j=1}^{n} g'_j(\boldsymbol{\omega},R_j)$$

where $g'_j(\boldsymbol{\omega},R_j) = g(\boldsymbol{\omega},R_j)+\gamma|R_j-\bar{R}_j|$. Hence, the new objective function is still separable and the constraint generation problem can be transformed into a network flow model. The column generation algorithm is still the same. However, we need to modify the transportation model to adjust this change in the objective function. Figure 4-2 shows the model in detail. We penalize the objective function if the realized ranking R_i is different from its nominal ranking $\bar{R}_i$. For example, if the actual rank of object 3 is 1 (or 2) instead of its nominal value 3, we would include a penalty of 2γ (or γ) into the objective function. Hence, the cost of assigning asset 3 to rank 1 (or 2) has been changed from g_1 to $g_1+2\gamma$ (or $g_1+\gamma$).

The choice of γ can be done through cross-validation. The set of possible candidates for γ can be found by comparing the relative values of the optimal value $f(\boldsymbol{\omega}^*,\mathbf{R}^{worst})$ of the robust model when $\gamma = 0$ and the corresponding deviation $|\mathbf{R}^{worst}-\bar{\mathbf{R}}|_{L1}$. Then we can set:

$$\gamma = \alpha * \frac{f(\boldsymbol{\omega}^*,\mathbf{R}^{worst})}{|\mathbf{R}^{worst}-\bar{\mathbf{R}}|_{L1}}$$

for some choice of α. With $\alpha = 1$, we treat the risk of being at the boundary of the uncertainty set and the risk of changing from the nominal value equally. Letting

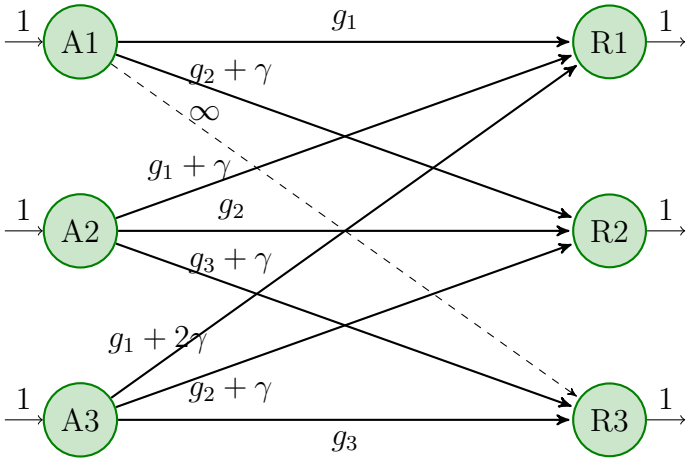


Figure 4-2: The network flow problem when the conservative level is incorporated

$\alpha < 1$ would mean we are more conservative. Letting $\alpha > 1$ would mean we believe more in the nominal value.

4.2.5 Extension to Allow Group Ranking

So far, we have assumed there is a complete ranking for all the objects. However, in many situations, objects are ranked into groups or tiers where there is no ranking information between those objects within the same group. For example, instead of having n different ranks for n assets, we can use only 10 ranks and divide assets into deciles. We can extend the robust ranking problem to allow robust group ranking easily. The uncertainty in this case comes from the fact that some assets might not be in their predicted tiers. Let K be the number of tiers. The algorithm for solving this problem is still unchanged. However, we need to modify the transportation model to be a network flow model with n source and K sink nodes. Let us take an example with $n = 4$ assets and $K = 2$ tiers. Asset 1 is predicted to be in the first tiers. Assets 2 and 3 are either in the first or the second tier. Asset 4 is in the second tier. The corresponding network flow model is modified according to Figure 4-3.

We have presented the general robust ranking model, its computational method using the column generation algorithm and its extensions. In the next section, we will show how to apply our robust ranking model to portfolio optimization.

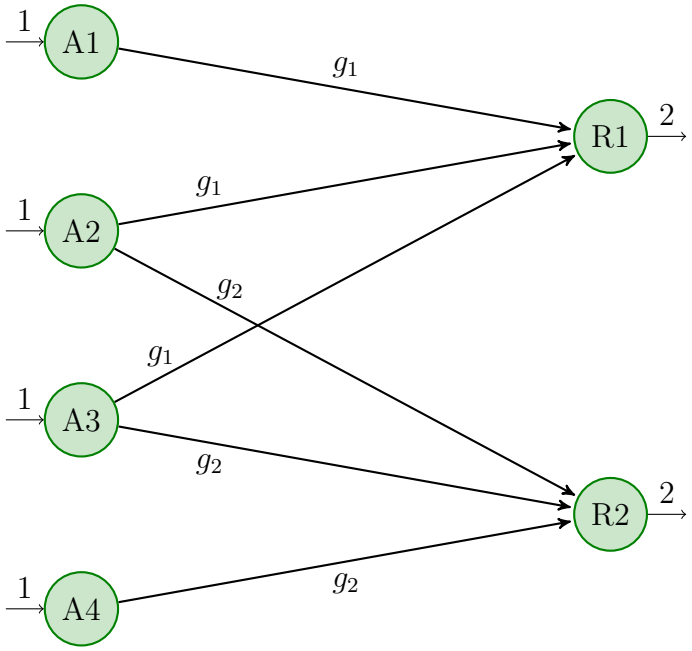


Figure 4-3: The network flow problem when allowing group ranking in the uncertainty set.

4.3 Robust Ranking Models for Portfolio Optimization

In this Section, we will apply our generic robust ranking model to portfolio optimization. We will show different choices of the objective functions and the constraint sets. We will present their computational performance in Subsection 4.4.1 and their empirical results in Subsection 4.4.2.

Consider a portfolio optimization problem where an investor forms a portfolio out of n assets to maximize some objective function. In the traditional Markowitz mean-variance portfolio optimization framework, we want to find the optimal portfolio to maximize some objective function whose input parameters include the mean and the covariance matrix. Many researchers include Michaud [30] and Chorpa and Ziemba [15] have shown the poor out-of-sample performance of the mean-variance optimal portfolio due to estimation errors on the expected return. The main advantage of using ranking is to avoid the difficult task of estimating the expected return. Most

trading strategies that are based on ranking are formed by putting the top ranked assets into the long side and the bottom ranked assets into the short side. This strategy would perform very well if the ranking information is correct and would perform poorly if the reverse is true. Hence, the risk of these portfolios is usually very high. Our robust ranking models allow the ranking to be in some uncertainty set and then find the best portfolio that performs well even for the worst realization of the ranking within the uncertainty set. In order to obtain the ranking uncertainty set, investors often use their intuition as well as quantitative methods to make inferences about the assets' ranking. For example, some investors might believe that assets belonging to one sector will outperform those in other sectors. Investors might also use well-documented accounting anomalies such as the post earning announcement drift (see Ball and Brown in [3]) or the Fama and French factors in [17] as the means to rank assets. Once we have obtained the ranking uncertainty set, we could apply our general robust ranking model. The choice of the objective function f and the constraint set $\mathcal{P}(\boldsymbol{\omega})$ could vary from investor to investor. We present two possible models in the next Subsections.

4.3.1 Model I: Robust Max Weighted Ranking with Nonnegative Weight Constraint

$$\begin{array}{ll} \max\limits_{\boldsymbol{\omega}} & \left[\min\limits_{\mathbf{R}\in\boldsymbol{\mathcal{U}}(\mathbf{R})} \quad \boldsymbol{\omega}^t\mathbf{R}\right] \\ s.t. & \boldsymbol{\omega}^t\boldsymbol{e} = 1 \\ & \boldsymbol{\omega}^t \geq 0 \end{array}$$

In this model, we want to find the optimal portfolio weight $\boldsymbol{\omega}$ to maximize the average ranking $\boldsymbol{\omega}^t\mathbf{R}$ for the worst realization of the ranking $\mathbf{R}$ that lies in the uncertainty set $\boldsymbol{\mathcal{U}}(\mathbf{R})$. The constraints $\boldsymbol{\omega}^t\boldsymbol{e} = 1$ and $\boldsymbol{\omega}^t \geq 0$ are for the normalization and short-selling restriction respectively. In this case, the objective function f is separable

in ranking and hence we can apply the constraint generation method.

$$f(\boldsymbol{\omega}, \mathbf{R}) = \boldsymbol{\omega}^t \mathbf{R} = \sum_{j=1}^{n} g_j(\boldsymbol{\omega}, R_j),$$

where $g_j(\boldsymbol{\omega}, R_j) = \omega_j R_j$.

4.3.2 Model II: Robust Max Weighted Ranking with Risk Constraint

$$\begin{array}{ll} \max\limits_{\boldsymbol{\omega}} & \left[\min\limits_{\mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R})} \quad \boldsymbol{\omega}^t \mathbf{R}\right] \\ s.t. & \boldsymbol{\omega}^t \boldsymbol{\Sigma} \boldsymbol{\omega} \leq 1 \end{array}$$

In this model, we want to find the optimal portfolio weight $\boldsymbol{\omega}$ to maximize the average ranking for the worst realization of the ranking $\mathbf{R}$ that lies in the uncertainty set $\boldsymbol{\mathcal{U}}(\mathbf{R})$. The constraint set $\boldsymbol{\omega}^t \boldsymbol{\Sigma} \boldsymbol{\omega} \leq 1$ allows the total portfolio risk to be bounded above. This model is very similar to a mean-variance portfolio optimization model except that we use the ranking $\mathbf{R}$ in the place of the expected return $\boldsymbol{\mu}$. This robust model is a hybrid between a nonparametric model using ranking and a parametric model using the covariance matrix. The main criticism for using the mean-variance portfolio optimization is in the estimation error from the expected return but not from the covariance matrix. Hence, we could retain the good features from the mean-covariance model by adding the risk constraint to the robust ranking model.

We can show that Model II is equivalent to:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\omega}} & \left[\min\limits_{\mathbf{R} \in \boldsymbol{\mathcal{U}}(\mathbf{R})} \quad \dfrac{\boldsymbol{\omega}^t \mathbf{R}}{\sqrt{\boldsymbol{\omega}^t \boldsymbol{\Sigma} \boldsymbol{\omega}}}\right] \\ s.t. & \boldsymbol{\omega}^t \boldsymbol{e} = 1 \end{array}$$

This model looks like a max Sharpe version. In fact, if we could transform a ranking into the equivalent returns, we could have a mapping from the ranking uncertainty set into a discrete expected return uncertainty set and then Model II would be exactly

the robust max Sharpe model. For example, suppose we decide to use the Almgren and Chriss method [1] to transform the complete ranking $R_1 \geq R_2 \geq ... \geq R_n$ into the equivalent returns c, where $c_1 \geq c_2 \geq ... \geq c_n$. Then, for any asset i with ranking information $l_i \leq R_i \leq u_i$, its return r_i belongs to the set $\{c_{l_i}, c_{l_i+1}, ..., c_{u_i}\}$. Thus, the uncertainty set for the returns is constructed as follows:

$$\boldsymbol{\mathcal{U}}(\mathbf{r}) = \left\{\mathbf{r} \mid r_i \in \{c_{l_i}, c_{l_i+1}, ..., c_{u_i}\},\ \forall\ i \in [1, 2, \ldots, n]\right\}$$

The robust max Sharpe portfolio optimization problem can be formulated as:

$$\begin{array}{lll} \max\limits_{\boldsymbol{\omega}} & \min\limits_{\mathbf{r}\in\boldsymbol{\mathcal{U}}(\mathbf{r})} & \dfrac{\sum_{i=1}^{n}\omega_i r_i}{\sqrt{\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega}}} \\ s.t. & \boldsymbol{\omega}^t\boldsymbol{e} = 1 & \end{array}$$

In addition to the max Sharpe objective function, it is also possible to do robust ranking for many other types of objective functions in the portfolio optimization problem such as a quadratic utility function.

4.4 Numerical Results

4.4.1 Computational Time

Table 1 shows the total solving time (in seconds), the average solving time for the relaxed problem and for the constraint generation problem and the number of iterations that the algorithm takes to converge to the optimal solution. We can see that the algorithm finds the optimal solutions very fast. For example, with $N = 100$ and $k = 20$, the algorithm converges in only 136 iterations even though the size of the uncertainty set is very large.

We also record the time for solving the relaxed problem and the constraint generation problem through iterations in Figure 4-4. We can see that the constraint generation problem takes about the same time in every iteration. This is because the transportation model does not change through iterations. The relaxed problem

Number of assets	10	20	20	50	75	100	100
$u_i - l_i$	4	4	10	10	10	10	20
Number of iterations	8	13	38	44	61	68	136
Average time solving the constraint generation problem	0.14	0.19	0.19	2.26	11.97	23.85	28.82
Average time solving the relaxed problem	0.41	0.46	0.5	1.26	4.17	5.79	7.36
Total time	4.59	9.11	27.9	156.5	987	2017	4921

Table 4.1: Computational time for running the constraint generation algorithm with different problem sizes. We used Matlab to run simulations. We used Sedumi for solving the relaxed problem and use Matlog for solving the constraint generation problem. All the computation results were produced by an Intel Pentium 4, 2GHZ and 256 MB of RAM computer. Notice that the solutions found were optimal. If we allow ϵ-violation on the constraint generation problem, the algorithm is expected to run much faster.

takes longer time through iterations because the number of constraints increases by 1 through each iteration. However, we can see that the increase is not significant.

4.4.2 Empirical Study

We apply robust ranking into portfolio optimization where the ranking is obtained from the standardized earning surprise. We first test the common belief about post announcement earning drift. We then compare the performance of the robust models with their non-robust counterparts. We test these models with stocks in the Dow Jones Industrial Average Index during the period from 2000 to 2007.

Post Earning Announcement Drift

Ball and Brown [3] were the first to document the post earning announcement drift effect. The authors define the standardized earning surprise as follows:

$$SUE_t = \frac{E_{a,t} - E_{e,t}}{\sigma(E_a - E_e)},$$

where $E_{a,t}$ and $E_{e,t}$ are the actual earning and the average analysts' estimates on earning for period t. The authors show that a simple portfolio with a long side of

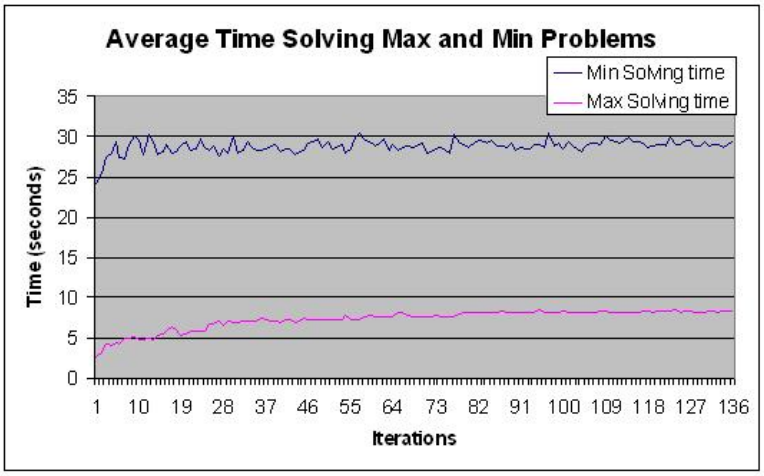


Figure 4-4: Computational time for solving the relaxed problem and the constraint generation problem.

the highest SUE assets and a short side of the smallest SUE assets would earn significant returns with high Sharpe ratio. This results provide us a method to rank assets according to their SUE_t as we will show in Subsection 4.4.2.

Data Collection

- **Stock Universe**: We construct our portfolio from stocks in the Dow Jones Industrial Average Index. We avoid possible errors with the database as much as possible by a) keep only those stocks with consistent earning records (every March 31st, June 30th, Sep. 30th and Dec. 31st) and b) removing those stocks whose issuing companies has merged or split or changed their names. This results in a set of 14 stocks in the period from 2000 to 2007. [1]

- **Earning Information**: The estimate earnings and actual earning are retrieved from the IBES database from the Wharton School of Business (WRDS).

- **Stocks' Returns**: The stocks's returns are retrieved from the CRSP database

[1] It is arguable that our choice of replicating the DJIA stock universe would contain survivorship bias. However, it is our main interest to study the relative performance of our robust models compared to their non robust counterparts

from the Wharton School of Business (WRDS).

- **Time Frame**: Our data starts from 2000. Since the CRSP and IBES only provide data up to 2007, our test results end in 2007.

Trading Strategy Implementation

Our trading time periods start right after the quarter earning announcement and we hold our trading position until the end of that quarter before a new portfolio is formed. Suppose we are at time t, that is the beginning of a quarter, we look back for 1 year and find the Dow Jones Industrial Average constituents that are in the index for the entire year. (this should provide a subset of the all the constituents at time t). We find the earning estimates by analysts for the last quarter and average them to obtain $E_{e,t}$. We also find the actual earning announced by the stock issuing companies for the last quarter to obtain $E_{a,t}$. We then find the standardized earning surprise as $SUE_t = \frac{E_{a,t}-E_{e,t}}{\sigma(E_a-E_e)}$, where $\sigma(E_a - E_e)$ is the historical standard deviation of the discrepancy between the estimated and actual earnings. We rank SUE_t of all the stocks selected and obtain a ranking $\bar{R} = \{\bar{R}_1, \bar{R}_2, ..., \bar{R}_n\}$. The uncertainty set is set equal to: $U(R) = R \mid (\bar{R}_i - l_i) \leq R_i \leq (\bar{R}_i + u_i)$ for different choices of widths l and u. We run the robust models to obtain the robust portfolio weights w_t. Since we hold this portfolio for the next quarter, the portfolio return for quarter $t+1$ is $w_t^t r_{t+1}$ where r_{t+1} is the return of the assets in quarter $(t+1)$

Empirical Results

First, we test whether the post announcement earning has any drift to stocks' returns. This is done by simply comparing between a portfolio of the highest ranked SUE stocks, another portfolio of the lowest ranked SUE stocks and the equal weighted portfolio. This comparison is shown in Table 4.2 and Figure 4-5. We can see the highest ranked portfolio outperforms the equal weighted and the lowest ranked portfolios (in terms of the expected returns and the Sharpe ratios). This implies that the post announcement earning does have drift on the stock returns in this case.

Model	Mean Returns	Standard Deviation	Sharpe Ratio
Equal Weighted portfolio	0.0795	0.1797	0.4424
Lowest SUE portfolio	0.0413	0.2033	0.2030
Highest SUE portfolio	0.1177	0.1829	0.6439

Table 4.2: Risk and return characteristics for different strategies. Rows 2, 3 and 4 show that the high SUE portfolio indeed outperforms the equal weighted portfolio and the low SUE portfolio. This implies that the post earning announcement does contain drift to the stocks' returns.

Next, we compare the performances between non-robust portfolios and robust portfolios in Table 4.3. The third and the fourth rows show the comparison between a non-robust portfolio of simply using the nominal ranking and a robust portfolio using Model I where the uncertainty set is set as $U(R) = R \mid (\bar{R}_i - 2) \leq R_i \leq (\bar{R}_i + 2)$. Rows 5,6,7,8,9 show the comparison between a non-robust portfolio and robust portfolios using Model II. The differences between the robust portfolios in rows 6,7,8,9 are in the way we construct the uncertainty sets. These sets are all constructed with the forms $U(R) = R \mid (\bar{R}_i - w_i) \leq R_i \leq (\bar{R}_i + w_i)$ where the widths w range from 1 to 4. In all cases, we can see clearly the robust portfolios have smaller risks (standard deviations) compared to their non-robust counterparts. The expected returns for these portfolios fluctuate around their non-robust counterparts. This is because our way of constructing the uncertainty sets was very simple and we did not use any additional information. Introducing the uncertainty sets would either a) reduce the value of the information we have in the ranking or b) make the ranking less sensitive to estimation errors or both. In fact, we can see both of these effects in this empirical example.

Finally, we test the performance of our robust models on varying the correctness of the uncertainty sets. We construct two portfolios: the first one with an additional 'good information' and the second one with an additional 'bad information' reflected in the uncertainty sets. These uncertainty sets are constructed as follows: $U(R) = R \mid (\bar{R}_i - 1 + w_i) \leq R_i \leq (\bar{R}_i + 1 + w_i)$, where $w_i = \pm 1$. Let RT be the truth ranking of the stocks in the next period. The first portfolio is set closer to the truth ranking RT while the second portfolio is set further from RT compared to the nominal ranking $\bar{R}$. In the first portfolio, we set:

Figure 4-5: This figure shows that post earning announcement does indeed have valuable information. The blue curve (the middle one) is for the equal weighted portfolio of 14 assets that are extracted from the Dow Jones Industrial Average Index. The black curve (the lowest one) is for the equal weighted portfolio of the lowest SUE assets. The red curve (the top curve) is for the equal weighted portfolio of the highest SUE assets.

$$w_i^{(1)} = \begin{cases} 1, & \text{if } \bar{R}_i < RT_i \\ -1, & \text{if } \bar{R}_i > RT_i \\ 0, & \text{otherwise} \end{cases}$$

In the second portfolio, we set $w_i^{(2)} = -w_i^{(1)}$. Table 4.4 shows the comparison between a robust ranking portfolio without any additional information and the two portfolios we have just constructed. We can see clearly that having additional good information (shown in the fourth row) produces a better portfolio than the one without it while having additional bad information (shown in the fifth row) could downgrade the portfolio. Hence, in practice, the uncertainty set should be set carefully. The confidence interval for each assets could be different from others. This is the point when beliefs from both quantitative analysts, fundamental analysts and managers should be put together to construct a good uncertainty set.

Model	Mean Returns	Standard Deviation	Sharpe Ratio
Equal Weighted portfolio	0.0795	0.1797	0.4424
Non-robust (I)	-0.0475	0.2859	-0.1661
Robust (I)	0.0906	0.1744	0.5194
Non-robust (II)	0.2173	0.2740	0.7931
Robust (II), ± 1	0.2387	0.2694	0.8861
Robust (II), ± 2	0.1962	0.2392	0.8200
Robust (II), ± 3	0.2471	0.2166	1.1408
Robust (II), ± 4	0.1528	0.2189	0.6984

Table 4.3: Risk and return characteristics for different strategies. Rows 3 and 4 show that a non-robust and a robust portfolio using Model I. Rows 5,6,7,8,9 show a non-robust portfolio and 4 robust portfolios using Model II. The uncertainty sets for the robust portfolios in rows 6,7,8,9 were constructed as $U(R) = R \mid (\bar{R}_i - w_i) \leq R_i \leq (\bar{R}_i + w_i)$ where the widths w range from 1 to 4. We can see that robust portfolios have smaller risk compared to their non-robust counterparts.

Model	Mean Returns	Standard Deviation	Sharpe Ratio
Equal Weighted portfolio	0.0795	0.1797	0.4424
Robust (II), ± 1	0.1977	0.2405	0.8220
Robust (II), ± 1 + good information	0.3377	0.3131	1.0787
Robust (II), ± 1 + bad information	0.1308	0.2528	0.5176

Table 4.4: Risk and return characteristics for different strategies. Rows 4 is for a robust portfolio whose uncertainty set is around the nominal ranking $\bar{R}$ but skew toward the true ranking. Row 5 is for another robust portfolio whose uncertainty set is skewed in the opposite direction compared to the nominal ranking. We can see that the 'closer' the uncertainty sets to the actual ranking, the better the corresponding robust portfolio performs.

4.5 Conclusion

In conclusion, we present a robust ranking model and apply the column generation method to solve the problem. The construction of the uncertainty set and the method for controlling the degree of conservatism are intuitive. We exploit the special characteristic of the ranking to ensure that constraints are generated efficiently through solving a network flow model. Our method works for a large class of objective functions. We show computational performance of the algorithm for the specific case of robust portfolio optimization. We found the algorithm performs very fast even for moderately large problem sizes. Our idea of using the column generation method is applicable in other robust models whose uncertainty sets are discrete as long as we can find an efficient method to generate constraints. For empirical tests, we use the post announcement earning drift to construct the ranking uncertainty set. We test

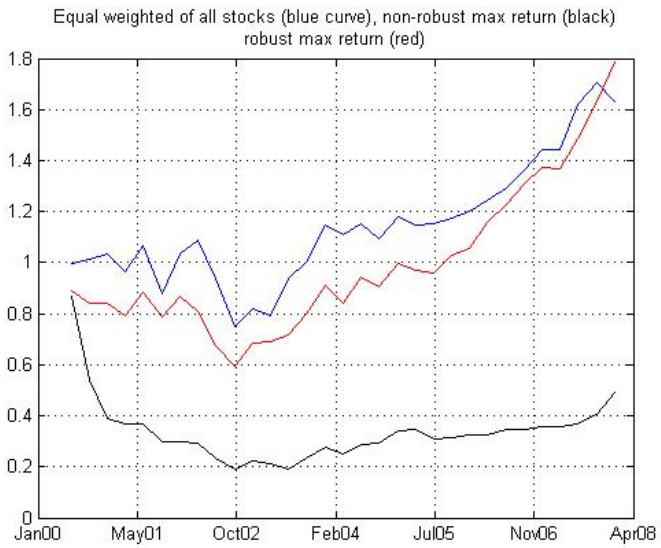


Figure 4-6: Result for model I: This figure compares the performance of the robust ranking model 1 versus its non-robust counterpart. The blue curve shows the cumulative quarterly returns of the equal weighted portfolio. The black curve shows that of the non-robust model where the uncertainty set is equivalent to a nominal ranking. The red curve shows that of the robust model. We can see the robust curve out-performs the non-robust curve.

our robust models for stocks in the DJIA index in the period from 2000-2007 and found the robust portfolios produce smaller risk and higher Sharpe ratios compared to their non-robust counterparts.

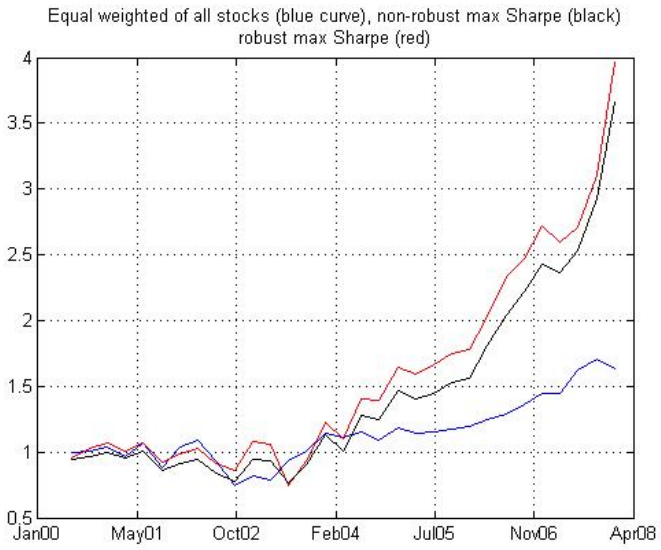


Figure 4-7: Result for model II: This figure compares the performance of the robust ranking model 2 versus its non-robust counterpart. The blue curve shows the cumulative quarterly returns of the equal weighted portfolio. The black curve shows that of the non-robust model where the uncertainty set is equivalent to a nominal ranking. The red curve shows that of the robust model. We can see the robust curve out-performs the non-robust curve.

Chapter 5

Future Directions

Robust optimization or robust statistics (or a combination)?
We have worked with both robust optimization and robust statistics and we found each of them has attractive properties as well as limitations. One of the interesting questions is whether we should use robust optimization or robust statistics in general and in financial applications in particular. We conjecture that the choice of which one to use depends significantly on the assumptions on the data generating processes. We also believe that a combination of robust optimization and robust statistics would lead to strong methods for financial applications. We find this topic very interesting and worth pursuing further.

Robust Estimation and Regime Switching in Portfolio Optimization:
It is clear that stock returns do not follow a normal distribution. It is possible that the returns follow a regime switching model where the majority of the observations follows a normal distribution while the rest follows some other random process. We could use robust estimation and robust regression to estimate these regimes. Once we have found the regimes, we could produce the corresponding optimal portfolios.

Applying OM/OR techniques to robust statistical analysis and other combinatorial problems:
We plan to continue the success of applying mathematical modeling and optimization methods to robust statistics. The most popular tools that the robust statistics community uses are heuristic methods and there is a great opportunity for apply-

ing mathematical programming as an alternative. One example that we found quite intriguing is that the PROGRESS algorithm which had been used for solving the robust regression problem for quite a long period of time is essentially equivalent to the simple gradient method. This can be seen directly when we model the problem as a nonlinear knapsack problem. In addition to robust statistics applications, we are also searching for other areas especially those combinatorial problems where heuristic methods or rules-based methodologies are used currently. The favorite optimization tool that we would rely on is semi-definite programming since it is so powerful in relaxing combinatorial problems as well as in transforming highly non-linear problems into linear matrix inequalities.

Portfolio optimization and other financial applications:
There are many possible research directions related to the portfolio optimization problem. Our work in [12] has shown that none of the current methods perform well on empirical data. We know that financial data doesn't really follow normality assumptions and we want to see how much this affects existing models. We also plan to test how my robust ranking method would work compared to other portfolio optimization methods. We also plan to test how my robust covariance matrix would work for portfolio optimization where we use the robust covariance matrix instead of the maximum likelihood counterpart.

Appendix A

A.1 Proof of Theorem 1

Theorem 1: Let $v_i = \begin{bmatrix} 1 \\ R_i \end{bmatrix}$ and $Z = \begin{bmatrix} 1 & \sum_{i=1}^n x_i R_i^t \\ \sum_{i=1}^n x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} = \sum_{i=1}^n x_i v_i v_i^t$, then the constraint $d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_i - \hat{\boldsymbol{\mu}}_{rob})$ is equivalent to $d_i + 1 \geq v_i^t Z^{-1} v_i$.

Proof. We have:

$$
\begin{aligned}
& d_i \geq (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \hat{\boldsymbol{\Sigma}}_{rob}^{-1} (R_i - \hat{\boldsymbol{\mu}}_{rob}) \\
\Leftrightarrow \quad & \begin{bmatrix} d_i & (R_i - \hat{\boldsymbol{\mu}}_{rob})^t \\ (R_i - \hat{\boldsymbol{\mu}}_{rob}) & \hat{\boldsymbol{\Sigma}}_{rob} \end{bmatrix} \succeq 0 \qquad \text{(A.1)} \\
\Leftrightarrow \quad & \begin{bmatrix} d_i & R_i^t - \hat{\boldsymbol{\mu}}_{rob}^t \\ R_i - \hat{\boldsymbol{\mu}}_{rob} & \sum_{i=1}^n x_i R_i R_i^t - \hat{\boldsymbol{\mu}}_{rob} \hat{\boldsymbol{\mu}}_{rob}^t \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & \begin{bmatrix} d_i + 1 & R_i^t \\ R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq \begin{bmatrix} 1 & \hat{\boldsymbol{\mu}}_{rob}^t \\ \hat{\boldsymbol{\mu}}_{rob} & \hat{\boldsymbol{\mu}}_{rob} \hat{\boldsymbol{\mu}}_{rob}^t \end{bmatrix} \\
\Leftrightarrow \quad & \begin{bmatrix} d_i + 1 & R_i^t \\ R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq \begin{bmatrix} 1 \\ \hat{\boldsymbol{\mu}}_{rob} \end{bmatrix} \begin{bmatrix} 1 & \hat{\boldsymbol{\mu}}_{rob}^t \end{bmatrix} \\
\Leftrightarrow \quad & \begin{bmatrix} 1 & 1 & \hat{\boldsymbol{\mu}}_{rob}^t \\ 1 & d_i + 1 & R_i^t \\ \hat{\boldsymbol{\mu}}_{rob} & R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq 0 \qquad \text{(A.2)}
\end{aligned}
$$

$$
\begin{aligned}
\Leftrightarrow \quad & \begin{bmatrix} d_i+1 & 1 & R_i^t \\ 1 & 1 & \hat{\boldsymbol{\mu}}_{rob}^t \\ R_i & \hat{\boldsymbol{\mu}}_{rob} & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix} \succeq 0 \\
\Leftrightarrow \quad & d_i + 1 \geq \begin{bmatrix} 1 & R_i^t \end{bmatrix} \begin{bmatrix} 1 & \sum_{i=1}^n x_i R_i^t \\ \sum_{i=1}^n x_i R_i & \sum_{i=1}^n x_i R_i R_i^t \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ R_i \end{bmatrix} \qquad \text{(A.3)} \\
\Leftrightarrow \quad & d_i + 1 \geq v_i^t Z^{-1} v_i
\end{aligned}
$$

Notice that we have used the Schur complements (see [42]), in A.1 and A.3 while the general theorem for A.2 is the following: $X - vv^t \succeq 0 \quad \Leftrightarrow \quad \begin{bmatrix} 1 & v^t \\ v & X \end{bmatrix} \succeq 0$. This result can be shown by noticing that $\begin{bmatrix} 1 & 0 \\ -v & I \end{bmatrix} \begin{bmatrix} 1 & v^t \\ v & X \end{bmatrix} \begin{bmatrix} 1 & -v^t \\ 0 & I \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & X - vv^t \end{bmatrix}$ □

A.2 Proof of Theorem 5

Theorem 5: At most (p+1) decision variables z_i are non-integer in the relaxed problem.

Proof. Consider the maximum trimmed squares problem:

$$
\begin{aligned}
\max_z \quad & f(z) \\
s.t. \quad & -z \leq 0 \\
& z \leq 1 \\
& z^t \boldsymbol{e} = q
\end{aligned}
$$

Let $\alpha \in \mathcal{R}^n$, $\beta \in \mathcal{R}^n$ and $\gamma \in \mathcal{R}$ be the Lagrangian multipliers for the constraints $-z \leq 0$, $z \leq 1$ and $z^t \boldsymbol{e} = q$ correspondingly. The KKT and the complementary slackness require the following conditions hold:

$$\begin{cases} \frac{\partial\left[f(z)-\alpha^t(-z)-\beta^t(z-1)-\gamma(z^t\boldsymbol{e}-q)\right]}{\partial z_i} = 0, \ \forall \ i \\ z_i\alpha_i = 0, \ \forall \ i \\ (z_i-1)\beta_i = 0, \ \forall \ i \\ (z^t\boldsymbol{e}-q)\gamma = 0 \\ \alpha \geq 0, \ \beta \geq 0 \end{cases}$$

We have:

$$\begin{aligned} 0 &= \frac{\partial\left(f(z)-\alpha^t(-z)-\beta^t(z-1)-\gamma(z^t\boldsymbol{e}-q)\right)}{\partial z_i} \\ &= (y_i - x_i\beta(z))^2 + \alpha_i - \beta_i - \gamma \end{aligned}$$

Let $\mathbf{S}$ be the set of all non-integer z_i. This means $0 < z_i < 1, \ \forall \ i \in \mathbf{S}$ and hence $\alpha_i = \beta_i = 0, \ \forall \ i \in \mathbf{S}$ by the complementary slackness conditions. Therefore,

$$(y_i - x_i\beta(z))^2 = \gamma, \ \forall \ i \in \mathbf{S}$$

By a small randomization on the original data (X, y), we can make sure that the system of equations $(y_i - x_i\beta(z))^2 = \gamma$ hold for no more than $(p+1)$ observations (y_i, x_i) (since the systems has $(p+1)$ degrees of freedom on $\beta(z)$ and γ). This means the cardinality of the set $\mathbf{S}$ is less than or equal to $(p+1)$. In other words, there are no more than $(p+1)$ non-integer solution z_i.

□

A.3 Proof of Theorem 6

Theorem 6: The MTS problem would identify at least 1 outlier under Assumption 1.

Proof. Suppose that the reverse is true, that is maximum trimmed squares problem

identifies points $P_1, P_2, ...P_{p+1}$ which are all good observations and suppose the optimal value of the MTS is f^*. Let $h = w(G)$ be the width between the parallel planes that contains all the good observations. Then $f^* \leq (p+1)\left(\frac{h}{2}\right)^2$ because the total squared residuals for $P_1, P_2, ...P_{p+1}$ is at most $(p+1)\left(\frac{h}{2}\right)^2$. Let X be an outlier. We choose a set of $(p-1)$ good observations G_1 such that the plane CD passing through these points and X would classify all remaining good observations to be in one side of that plane (see Figure A-1)

Let D be the vertical distance from point X to the robust regression line of the good set G and let $d = D - \frac{h}{2}$. According to our assumption on G to be a good subset in the "strong robust sense", we must have $d \geq \sqrt{\frac{(p+1)}{2}}h$. Let H be the distance between any good observation G_2 to the plane CD, then the sum of squared residuals for the subset of X, G_1 and G_2 is at least $min\left(\frac{H^2}{2}, \frac{d^2}{2}\right)$. We will show the proof for this result later but suppose for now that this result is true. Then, because the subset of X, G_1 and G_2 is not the optimal MTS solution, we have:

$$(p+1)\left(\frac{h}{2}\right)^2 \geq min\left(\frac{H^2}{2}, \frac{d^2}{2}\right)$$

Since $d \geq \sqrt{\frac{(p+1)}{2}}h$, the following inequality must true:

$$\begin{aligned} (p+1)\left(\frac{h}{2}\right)^2 &\geq \frac{H^2}{2} \\ \Rightarrow \quad \sqrt{\frac{(p+1)}{2}}h &\geq H \end{aligned}$$

This means we can find a pair of parallel planes with a width of $\sqrt{\frac{(p+1)}{2}}h$ that contains all the good observations and the outliers X. This contradict to our assumption for G to be the good subset in the "strong robust sense".

Let MN be the regression plane that corresponds to the points G_1 G_2 and X and let r_1, r_2 and r_3 be their corresponding residuals. The only result left for us to prove is $r_1^2 + r_2^2 + r_3^2 \geq min\left(\frac{H^2}{2}, \frac{d^2}{2}\right)$.

There are two cases. The projection of G_2 on the plane CD either lie within or outside the convex hull of the points X and G_1. In the former case, we can prove that $r_1^2 + r_2^2 + r_3^2 \geq \frac{H^2}{2}$ and in the later case, we can prove $r_1^2 + r_2^2 + r_3^2 \geq \frac{d^2}{2}$. We only show the proof for the former case here. We have:

$$(H - r_2)^2 \leq max(r_1^2, r_3^2) \leq r_1^2 + r_3^2$$
$$\Rightarrow r_1^2 + r_2^2 + r_3^2 \geq (H - r_2)^2 + r_2^2 \geq \frac{H^2}{2}$$

□

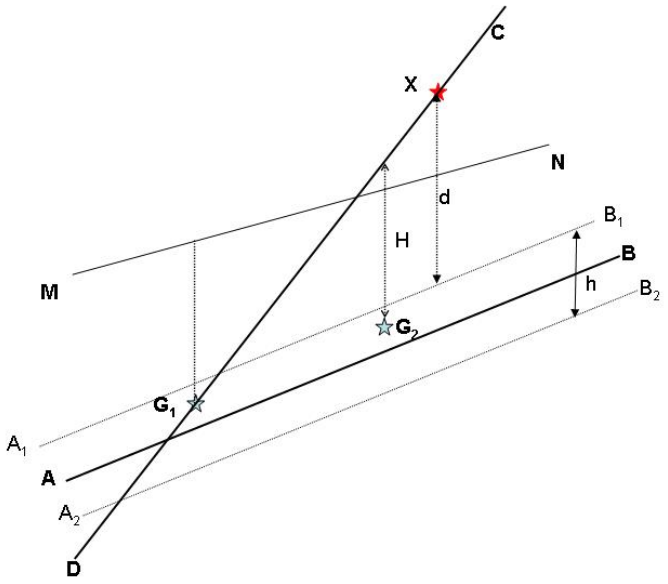


Figure A-1: Figure demonstrating a geometrical intuition for the proof of Theorem 2

A.4 Stock Universe for Robust Ranking Empirical Tests

The following table shows the list of stocks we used in our empirical test:

Symbols	Company Names
'AA'	Alcoa Incorporated
'BA'	Boeing Corporation
'C'	Citigroup Incorporated
'CAT'	Caterpillar Incorporated
'GE'	General Electric Company
'GM'	General Motors Corporation
'IBM'	International Business Machines
'INTC'	Intel Corporation
'JNJ'	Johnson & Johnson
'KO'	Coca-Cola Company
'MCD'	McDonalds Corporation
'MRK'	Merck & Company
'MSFT'	Microsoft Corporation
'UTX'	United Technologies

Table A.1: List of assets in our sample.

The remaining 16 assets in the DJIA index were not included for various reasons. The companies with symbols 'AXP', 'DD', 'DIS', 'MMM', 'PG', 'XOM' were not included because they missed the earning estimates for September 2000. The companies 'HD', 'HPQ', 'WMT' were not included because their earning announcements were on a different time scale than the rest, i.e. their announcement were at the end of April, July, Oct. and Jan. while the rest are at the end of March, Jun., Sep. and Dec. (at least this is what we saw on the Wharton database). The companies: 'AXP', 'JPM', 'BAC', 'CVX', 'KFT', 'PFE', 'VZ' were not included because either they have merged/split or changed their name in the last 8 years which might cause the Wharton database not to record their earning and returns properly.

Bibliography

[1] R. Almgren and N. Chriss, *Optimal portfolios from ordering information*, Journal of Risk (2006).

[2] F. Alqallaf, S.V. Aelst, V.J. Yohai, and R.H. Zamar, *A model for contamination in multivariate data*, (2005).

[3] R. Ball and P. Brown, *An empirical evaluation of accounting income numbers*, Journal of Accounting Research (Autumn 1968), 159–178.

[4] A. Ben-Tal and A. Nemirovski, *Robust solutions of linear programming problems contaminated with uncertain data*, Mathematical Programming **88** (2000).

[5] D. Bertsimas and R. Shioda, *Classification and regression via integer optimization*, Operations Research **55** (2007), no. 2, 252271.

[6] D. Bertsimas and M. Sim, *The price of robustness*, Operations Research **88** (2002).

[7] D. Bertsimas and A. Thiele, *A robust optimization approach to inventory theory*, Operations Research **54**, no. 1.

[8] D. Bertsimas and J.N. Tsitsiklis, *Introduction to linear optimization*, Athena Scientific, Belmont, MA, 1997.

[9] H.G. Beyer and B. Sendhoff, *Robust optimization - a comprehensive survey*, to be appeared (2007).

[10] D. Bienstock, *Experiments with robust optimization*, ISMP 2006, Rio (Aug. 2006).

[11] F. Black and R. Litterman, *Global portfolio optimization*, Financial Analysts Journal (Sep.-Oct. 1992).

[12] T. Brennan, A. Lo, and T.D Nguyen, *Portfolio theory*, To be appeared (2007).

[13] Gorlich A. Jarosz A. Burda, Z. and J. Jurkiewicz, *Signal and noise in correlation matrix*, Physica A (2004).

[14] V. Chandola, A. Banerjee, and V. Kumar, *Outlier detection: A review*, Technical Report (2007).

[15] V.K. Chopra and W.T. Ziemba, *The effect of errors in means, variances and covariances on optimal portfolio choice*, Journal of Portfolio Management (Winter 1993).

[16] V. DeMiguel and Nogales F., *Portfolio selection with robust estimates of risk*, LBS working paper, submitted for publication (2006).

[17] E.F. Fama and K.R. French, *Common risk factors in the returns on stocks and bonds*, Journal of Financial Economics **33** (1993), no. 1, 3–56.

[18] D. Goldfarb and G. Iyengar, *Robust portfolio selection problems*, Math. Oper. Res. **28** (2002), no. 1.

[19] P.W. Holland and R.E. Welsch, *Robust regression using iteratively reweighted least-squares*, Communications in Statistics: Theory and Methods **6** (1977), 813–827.

[20] P.J. Huber, *Robust statistics*, John Wiley and Sons, New York, 2004.

[21] P. Jorion, *Bayes-Stein estimation for portfolio analysis*, Journal of Financial and Quantitative Analysis **21** (1986), 279–292.

[22] J. Khan, V.S. Aelst, and R. Zamar, *Robust linear model selection based on least angle regression*, Technical Report, Department of Statistics, University of British Columbia (2005).

[23] H.W. Kuhn and A.W. Tucker, *Nonlinear programming*, Proceedings of 2nd Berkeley Symposium (1951), 481–492.

[24] Cizeau P. Bouchaud J.-P. Laloux, L. and M. Potters, *Noise dressing of financial correlation matrices*, Physical Review Letters.

[25] Cizeau P. Potters M. Laloux, L. and J.-P. Bouchaud, *Random matrix theory and financial correlations*, International Journal of Theoretical and Applied Finance **3** (2000), no. 3, 391–397.

[26] O. Ledoit and M. Wolf, *A well-conditioned estimator for large-dimensional covariance matrices*, Journal of Multivariate Analysis **88** (Feb. 2004), no. 2, 365–411.

[27] H. Markowitz, *Portfolio selection*, Journal of Finance **7** (Mar. 1952), no. 1, 77–91.

[28] R.A. Maronna, R.D. Martin, and V.J. Yohai, *Robust statistics: Theory and methods*, John Wiley and Sons, New York, 2006.

[29] R.O. Michaud, *Efficient asset management*, Harvard Business School Press, Boston, Massachusetts, 1998.

[30] ———, *The Markowitz optimization enigma: is 'optimized' optimal?*, Financial Analysts Journal **45** (Jan.-Feb. 1989), no. 1, 31.

[31] S. Pafka and I. Kondor, *Noisy covariance matrices and portfolio optimization ii*, Physica A **319** (Mar. 2003), no. 1.

[32] G. Perakis and E. Adida, *A robust optimization approach to dynamic pricing and inventory control with no backorders*, Mathematical Programming: Series A and B **107** (2006), no. 1, 97 – 129.

[33] G. Perakis and A. Sood, *Competitive multi-period pricing for perishable products*, Mathematical Programming: Series A and B **107** (2005), no. 1, 295 – 335.

[34] P.J. Rousseeuw, *Least median of squares regression*, Journal of the American Statistical Association **79** (1984), 871–880.

[35] P.J. Rousseeuw and A.M Leroy, *Robust regression and outlier detection*, John Wiley and Sons, New York, 1987.

[36] P.J. Rousseeuw and K. van Driessen, *A fast algorithm for the minimum covariance determinant estimator*, Technometrics **41** (1999), 212 – 223.

[37] ———, *Computing lts regression for large data sets*, Data Mining and Knowledge Discovery **12** (2006), 2945.

[38] A.L. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Oper. Res. **21** (2007).

[39] K.C. Toh, M.J. Todd, and R.H. Tutuncu, *Sdpt3 version 4.0 (beta) – a matlab software for semidefinite-quadratic-linear programming*, (2006).

[40] R.H. Ttnc and M. Koenig, *Robust asset allocation*, Annals of Operations Research **132** (2004).

[41] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), no. 1, 49–95.

[42] ———, *Applications of semidefinite programming*, Applied Numerical Mathematics: Transactions of IMACS **29** (1999), no. 3, 283–299.

[43] S. Verboven and M. Huberty, *Libra: a matlab library for robust analysis*, (2004).

[44] R. Welsch and X. Zhou, *Application of robust statistics to asset allocation models*, Forthcoming in REVSTAT (2007).

[45] G. Zioutas and A. Avramidis, *Deleting outliers in robust regression with mixed integer programming*, Acta Mathematicae Applicatae Sinica **21** (2005), no. 2, 323–334.