Modeling High Dimensional Data: Prediction, Sparsity, and Robustness

by

Chinghway Lim

A dissertation submitted in partial satisfaction of the requirements for the degree of
Doctor of Philosophy

in
Statistics

in the
Graduate Division

of the
University of California, Berkeley

Committee in charge:
Professor Bin Yu, Chair
Professor Peter Bickel
Professor Laurent El Ghaoui

Fall 2011
Modeling High Dimensional Data: Prediction, Sparsity, and Robustness

Copyright 2011
by
Chinghway Lim
Abstract

Modeling High Dimensional Data: Prediction, Sparsity, and Robustness

by

Chinghway Lim

Doctor of Philosophy in Statistics

University of California, Berkeley

Professor Bin Yu, Chair

This dissertation is on high dimensional data and their associated regularization through dimension reduction and penalization.

We start with two real world problems to illustrate the practical difficulties and remedies in analyzing high dimensional data. In Chapter 1, we are tasked with modeling and predicting the U.S. stock market, where the number of stocks far exceeds the number of days relevant to the current market. Through an existing statistical arbitrage framework, we reduce the dimension of our problem with the use of correspondence analysis. We develop a data driven regression model and highlight some common statistical methods that improve our predictions. In Chapter 2, we attempt to detect and predict system anomalies in large enterprise telephony systems. We do this by processing large amounts of unstructured log files, again with dimension reduction methods, allowing effective visualization and automatic filtering of results.

We then move on to more general methodology and analysis in high dimensions. In Chapter 3, we consider regularization methods, often used in dealing with high dimensional data, and tackle the problem of selecting the associated regularization parameter. We introduce SSCV, a selection criterion based on statistical stability, but also incorporating model fit, and show that it can often outperform the popular cross validation. Finally, we explore robust methods in the high dimensional setting in Chapter 4. We focus on the relative performance and distributional robustness of the estimators optimizing $L_1$ and $L_2$ loss functions respectively. We verify some expected results and also highlight cases where results from classical asymptotics fail, setting the stage for future theoretical work.
To Jiaen,

for her love, patience and support.
Contents

List of Figures v

List of Tables vii

1 Predicting the U.S. Stock Market 1
   1.1 Background 1
   1.1.1 Performance Measures 1
   1.2 Data 2
       1.2.1 Pre-processing 2
       1.2.2 Setup 3
   1.3 Residual Framework 3
       1.3.1 Peer Groups and Residuals 3
       1.3.2 Correspondence Analysis (CA) 4
       1.3.3 Principal Components Analysis (PCA) 6
   1.4 Methods 8
       1.4.1 Mean Reversion and Sign Switching 10
       1.4.2 CA regression 11
       1.4.3 Building on Regression 13
   1.5 Conclusion 17

2 Analysis of Enterprise Telephony Systems Logs 18
   2.1 Background 18
   2.2 Introduction 18
   2.3 Enterprise Telephony System 20
       2.3.1 Log Collection 21
   2.4 Analysis Goals 21
   2.5 Log Preprocessing 22
       2.5.1 Message Clustering 23
       2.5.2 Message Normalization 24
   2.6 Analysis for Logs with Known Failures 24
       2.6.1 Frequent Itemset Mining 24
2.6.2 Overall Message Frequency Analysis ................. 25
2.6.3 Individual Message Frequency Analysis ............. 26
2.6.4 Application Specific Clustering .................. 27
2.7 Analysis in the absence of failure markers .......... 27
2.8 Conclusions ........................................ 28

3 Statistical Stability ........................................ 33
3.1 Introduction ........................................... 33
3.1.1 Regularization Methods ......................... 33
3.1.2 Selecting the Regularization Parameter $\lambda$ ......... 34
3.1.3 Statistical Stability .............................. 34
3.2 Methodology ........................................... 35
3.2.1 Lasso and Pseudo Solutions ..................... 35
3.2.2 Alignment ...................................... 36
3.2.3 Convergence of Pseudo Solutions ................ 37
3.2.4 Hypothesis Testing and the Statistical Stability Metric .... 38
3.2.5 Incorporating and Exploiting Cross-Validation ....... 40
3.3 Experimental Results ................................... 41
3.3.1 Gaussian Simulation ............................. 41
3.3.2 fMRI Data ...................................... 44
3.3.3 Cytokine Data ................................... 45
3.4 Discussion .............................................. 49
3.4.1 Choice of Stability Metric ......................... 49
3.4.2 Data Perturbation Schemes .................... 49
3.4.3 Computation Cost .................................. 50
3.5 Conclusion ............................................. 50

4 Robustness in High Dimensions .......................... 53
4.1 Introduction ........................................... 53
4.1.1 Model ........................................... 53
4.1.2 Classical Setting ................................ 54
4.1.3 Robustness ...................................... 54
4.1.4 High Dimensions ................................ 54
4.2 $p > n$ case .......................................... 55
4.2.1 Exploring Tail Distributions ..................... 55
4.2.2 Bias-Variance Breakdown for the Double Exponential Case .... 57
4.2.3 Decoupling Selection and Fitting ................ 59
4.3 $p \leq n$ case, $p/n \to \rho$ ............................ 59
4.3.1 Double Exponential Tails ....................... 60
4.3.2 Family of $t$ Tails ............................... 64
4.3.3 Asymptotics: $p/n \to \rho$ ......................... 64
4.4 Conclusion ......................................................... 66

Bibliography ......................................................... 69
List of Figures

1.1 CA scree plot for 2003 1-day returns of our 1317 stocks. We use the first 15 dimensions, demarcated by the vertical line. ......................... 7
1.2 1-day returns and their projections of the stock ’MEDI’ onto 15 dimensional CA and PCA spaces. ............................................. 9
1.3 Mean of lag 1 autocorrelations of residuals past the training period for all 1317 stocks vs. for just the top 100 stocks ranked by negative autocorrelation in the training period. ........................................... 12
1.4 Predictive performance comparison of the three models. Sign-switching simply estimates the next residual as the negative of current residual. Base regression weights the predicted residuals with the training autocorrelation, and full regression includes all four terms. ........................................... 14
1.5 Predictive performance comparison for the various methods. The latter 3 methods are applied to the full regression model. “l1clust”, which is $L_1$ regression coupled with $k$-means clustering achieved the highest mean score. 16

2.1 An Enterprise Telephony System .................................................. 20
2.2 Overall Message Frequency Analysis ........................................... 25
2.3 Use of Slope-based Filters reduces codes ..................................... 29
2.4 Window-Max Filter ................................................................... 30
2.5 Figure 2.3(a) with application-specific clustering of dominating code ........................................................................... 31
2.6 108 days of log without any explicit failure markers ..................... 32

3.1 Approximate bootstrap distribution of maximum $L_1$-norms of Lasso estimates on a typical simulated data set. ................................. 37
3.2 Examples of the sample variance metric. The left panel shows an example where the metric exhibits a “dip”, representing the “convergence” of the pseudo solutions. The right panel shows an example with a much muted “dip”. 39
3.3 Examples of the sample variance metric and the corresponding $SS$ metric. 40
3.4 Scatterplot of predictive correlation scores of $SSCV$ and $CV$ for the top 500 voxels in the fMRI data set. ................................. 46
3.5 Feature selection by SSCV and CV on four randomly selected voxels. The “o”s represent features selected by both methods, while the “+”s represent features selected only by CV. The axes represent the pixel location of the images. The position and size of the points represents the wavelet location and size respectively. Note that most of the extra features CV select are scattered.

3.6 Feature selection results on cytokine data. The columns represent signal pathways predictors and each block of four rows correspond to a cytokine. The four rows within each block represent the selections of the four methods: the final minimal model (MM) and principal component regression (PCR) from the original study, and Lasso with SSCV and CV. The white squares correspond to selected predictors.

4.1 Plot of MSPE in Table 4.4. Circles and solid lines represent OLS values, triangles and dashed lines represent LAD values. The different colors represent the different scale factors for $\epsilon$. Note $n$ is fixed at 100.

4.2 The ratio of MSPE for LAD to MSPE for OLS. The black circles represent the double exponential tails case, and the red pluses represent the Gaussian tails case. Each panel corresponds to the same $p$.

4.3 The ratio of MSPE for LAD to MSPE for OLS for a family of $t$ distributed $\epsilon$. Each line represent a $t$-distributed $\epsilon$ with different degrees of freedom $\nu$. 
List of Tables

3.1 Effect of alignment on cross validation performance. 38
3.2 Performance of SSCV, CV and BIC in picking the regularization parameter for the Lasso for our base case design: constant correlation, $p = 150$. 43
3.3 Standard errors (SE) for performance numbers in Table 3.2. 43
3.4 Performance on fMRI data set. The numbers are averaged across the respective hundred voxels. 45
3.5 Performance of SSCV, CV and BIC in picking the regularization parameter for the Lasso for $p = 50$. 51
3.6 Performance of SSCV, CV and BIC in picking the regularization parameter for the Lasso for $p = 500$. 51
3.7 Performance of SSCV, CV and BIC in picking the regularization parameter for the Lasso for the block correlation design. 52
3.8 Performance of SSCV, CV and BIC in picking the regularization parameter for the Lasso for the Toeplitz correlation design. 52
4.1 Median MSPE of $\hat{\beta}_{L_2L_1}$ and $\hat{\beta}_{L_1L_1}$ for the basic setup: $n = 100$, $p = 200$, $s = 10$ and non-zero $\beta_j = 1$. The four error distributions, Gaussian, Logistic, Double Exponential and Cauchy, are in order of tails with increasing weight. Numbers are based on 100 runs. 57
4.2 Bias-Variance breakdown of the mean MSPE for the double exponential case. 58
4.3 MSPE performance of two-step regression for the double exponential case. Penalized regression is used to select the variables and regular regression is used to fit. Numbers are based on 1000 runs. 60
4.4 MSPE numbers for OLS and LAD for double exponential tails. Numbers are based on 1000 runs. 61
4.5 Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j = 1$. $p = 200$. This is the extended version of Table 4.2. 67
4.6 Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j = 1$. $p = 400$. 67
4.7 Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j \propto 1/(25 + j^2), \sum_j \beta_j^2 = s. \quad p = 200$. This represents the case where $\beta$ has a moderate decay.

4.8 Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j \propto 1/(5 + j^2), \sum_j \beta_j^2 = s. \quad p = 200$. This represents the case where $\beta$ has a steep decay.
Acknowledgments

I like to thank my advisor, Professor Bin Yu, for the years of guidance. It is an understatement that completing this dissertation would be impossible without her. She not only discharged her duty as an academic advisor dutifully, but also motivated me and cultivated my passion for statistics.

I like to thank Ariel Evnine, Lisa Borland, Jeremy Evnine, Roberto Osorio, Venkatesh Nagarajan, and others at Evnine and Associates whom I worked with on the stock market prediction project. Thanks for being so kind and generous with your insights.

I like to thank my mentors and co-authors Navjot Singh and Shalini Yajnik at Avaya Labs for the revealing experience in research outside the academic setting. Thanks for being welcoming and encouraging. It was a very fulfilling summer for me.

I like to thank Derek Bean, Peter Bickel and Noureddine El Karoui whom I worked with on the robust statistics project. Thanks for working so closely with me for the duration of the project.

I like to thank Vince Vu, David Purdy, Guilherme Rocha, Pradeep Ravikumar, Ethan Anderes and other senior members of my research group for showing me the ropes when I was starting out. The tools and skills I learnt early on were invaluable.

I like to thank Garvesh Raskutti, Kyle Jia, Nancy Wang, Karl Rohe, Yuval Benjamini, Harry Kim, Luke Miratrix, Hongwei Li and other newer members of my research group for all the helpful discussions and comments along the way.

Last but certainly not least, I like to thank Yu-Jay Huoh, Jing Lei, Brad Luen, Mike Higgins, Moorea Brega, Dave Shilane, Allan Sly and everyone I shared good times with in my days at Berkeley Statistics. I will always treasure those memories.
Chapter 1

Predicting the U.S. Stock Market

1.1 Background

In this chapter, we study modeling and predictions of the U.S. stock market based on statistical arbitrage. Statistical arbitrage in general is a trading strategy that exploits statistical mispricing of assets based on their expected value. When predictions of the expected values are accurate, this results in a guaranteed profit in the long run. More recently, statistical arbitrage refers to a more specific category of strategies used commonly by hedge funds. Common features include maintaining market neutral portfolios, using data dependent trading signals (as opposed to fundamentals), and high portfolio turnover rates.

In our case, we seek to predict the stock movements based only on its history. That is, we do not take in auxiliary information in pricing the stocks’ expected value. For example, we will not use sector or industry labels of our stocks but instead define groupings based purely on historical stock prices. We will work within an existing “residual framework”, described in Section 1.3, focusing on implementing statistical ideas for high dimensional data. Our resulting predictions are not meant to be the end-all answer, but instead should be viewed as an attempt at finding independent predictions to complement the other signals and heuristics in use. Nevertheless, we seek to get as much out as we can from our data, described in Section 1.2, utilizing a series of statistical tools to improve our predictive performance.

1.1.1 Performance Measures

The ultimate goal in predicting the U.S. stock market is of course to generate profits through trading stocks. We need to translate our predictions to portfolio construction: maintain an active stock portfolio and generate trading rules dictating when to execute trades. While predictions are the main driver of any portfolio construction algorithm, there are a variety

\footnote{This work was done in part as a consulting project with Evnine & Associates, an investment firm which managed equity portfolios.}
of other considerations. Firstly, there is a need to manage risk. This depends on your risk
appetite and willingness to take on different types of risk e.g. market risk, exposure to
certain sectors etc. Secondly, the very act of trading impacts the market. This is somewhat
mitigated by trading only stocks with high liquidity and large market capitalization, but the
impact is nevertheless present. Neither factor is trivial to model, and we will not attempt
to do so here.

We will instead use correlation between predicted and realized returns as a benchmark.
We use correlation instead of absolute losses since it is the relative positions of the predicted
returns that matter in portfolio construction. To be clear, the correlation score is for each
day across stocks, though we usually aggregate it across days afterwards. We also observe
Spearman’s rank correlation. Although the magnitudes can be quite different, it largely
reflects the same conclusions when using it to compare predictions between different models
and methods.

1.2 Data

We have price data from a universe of U.S. stocks over the period January 3, 2000 to
September 10, 2007. These are 1317 out of over 5000 stocks listed in New York Stock
Exchange and NASDAQ chosen by proprietary criteria. They satisfy certain criteria such
as minimum stock price, trade volume, minimum market capitalization among others. In
general, these are stocks with relatively high liquidity and hence relatively low trading costs.
This is to minimize the impact of the trading strategy on the market itself. There is survival
bias, as only stocks which are still trading at the end of the period are included.

1.2.1 Pre-processing

For each stock, we take its daily closing price to represent its price for the day. We use
the closing price and not the opening price as prices are more stable at the end than at the
start of the trading day. More importantly, the closing price represents a more plausible
trading scenario in simulations. The working assumption here is that we are able to observe
the closing price, say five minutes before closing, and execute our desired trade in the five
minute window.

The closing prices quoted in the data are already back-adjusted to account for dividends
and splits. This is important for backtesting simulations as well as training the models.
For example, if a stock trades at $10 and gets a 2:1 split, it would suddenly appear as $5
after the split. If the previous prices were not adjusted, we would see a huge drop in stock
price signaling a non-existent catastrophic event. In contrast the back-adjusted data would
have prices prior to the split multiplied by 0.5. Similarly, dividends are accounted for by
multiplying previous prices by the suitable multiplier.

There are some missing data. This corresponds to stocks which were not traded on a
trading day. This can be due to the lack of liquidity (which we have already tried to take care of), but more often due to trading halts. To handle this, we simply impute the missing data with the previous known value. If there is no previous known value (e.g. the stock only started trading in the period), we impute the first known value.

1.2.2 Setup

Let $A$ be the $n \times q$ data matrix where element $a_{ij}$ represents the log price of stock $i$ on day $j$ for $i = 1, \ldots, n$ and $j = 1, \ldots, q$. Of course, the price of the stock is just an arbitrary unit. We are more interested in the returns of the stock: its gain or loss relative to its price. Let $B^{(k)}$ be the $n \times (q - k)$ matrix with element

$$b_{ij}^{(k)} := a_{i,j+k} - a_{ij}$$

representing the $k$-day (log) returns of stock $i$ on day $j$. For our data, $n = 1317$, $q = 1932$ and we looked at $k = 1, \ldots, 8$.

Let $X$ be some contiguous block of $p$ columns of the matrix $B^{(k)}$ for analysis. We think of $p$ as the number of days in the market's history that is relevant to the current market. These $p$ days are also known as the look back period, from which we fit our model. In backtesting simulations, where we use past data to evaluate our models and methods, we think of this as the training data, with the subsequent columns in $B^{(k)}$ used to evaluate predictions based on $X$. In practice, $X$ represents the rightmost block of $B^{(k)}$, the most recent $p$ days, used to predict the immediate future.

We will fit a model for each $k$. We can try to combine the predictions from all the models to make a “best” aggregated prediction. Recall, however, that the results from the modeling here will only be one component of the overall trading strategy. With that in mind, we instead focus on doing the best we can for each $k$, and leave the problem of finding the optimal aggregation alone for now.

To sum up, $X$ is the $n \times p$ matrix of returns where each row corresponds to a stock and each column corresponds to a day.

1.3 Residual Framework

The residual framework is the backbone of our model. It formalizes the notion of mean-reversion that we want to exploit.

1.3.1 Peer Groups and Residuals

Let $\hat{X}$ and $R$ be $n \times p$ matrices. $\hat{X}$ is thought of as the “expected” returns corresponding to $X$. In practice, $\hat{X}$ is not the expectation of $X$ in the statistical sense, but instead contains fitted values based on $X$ itself. Typically, the fitted values are based on industry groups
and sectors: the “expected” return of a stock would be a weighted average of the returns of stocks in its group or sector. For example, say stock \( i \) is Southwest Airlines Inc. (LUV), the \( i \)th row of \( X \) would be its returns while the \( i \)th row of \( \hat{X} \) could be the returns of the Dow Jones Transportation Average (DJTA), an index of the transportation sector. We say that the stocks that make up the DJTA form LUV’s peer group. In this context, we can also interpret \( \hat{X} \) as peer group returns.

Now, define

\[ R := X - \hat{X} \]

as our matrix of residuals. As before, each row represents a stock, and each column represents a day in our training period. By its very definition, its element \( r_{ij} \) quantifies how well stock \( i \) performed relative to its peer group on day \( j \). The working assumption here though, is that the peer group’s performance is indicative of the stock’s performance. This implies that often times, the difference is a temporary mispricing of the stock. It is intuitively clear then why mean-reversion occurs. For example, a large residual suggests that the corresponding stock is overvalued and that the market will correct this inefficiency exerting downward pressure on the stock’s price. This is the effect we will try to capture in Section 1.4

The effectiveness of the residual framework hinges on the underlying peer group definitions. Traditional peer group definitions are based on economic fundamentals such as industry sectors, market capitalization size, and growth vs value characteristics. There is no doubt that they capture valuable information useful for prediction. Nevertheless, we focus on finding a data driven mechanism for peer group definitions. We do this for two main reasons. Firstly, a data driven mechanism is more dynamic. It allows peer group definitions react to changes in the market as new data becomes available. Secondly, it may uncover unexpected relationships that are not explained or well understood by economic theory. We note again we are not claiming our approach will give us the best predictions. Our goal is more modest: to seek unexplored information to complement existing predictions.

1.3.2 Correspondence Analysis (CA)

Correspondence analysis (CA) is a multivariate descriptive and exploratory technique. It is traditionally used to analyze contingency tables, giving a measure of correspondence between rows and columns of the table. Its use has since expanded, including unexpected applications like image denoising [7]. See [16] for a comprehensive review. CA is well suited for our stock returns data, where stocks (rows) and days (columns) have an intertwining relationship. For example, a stock can be defined by its returns over a few specific days, e.g. reaction to natural disasters, and at the same time, a day can be defined by its returns over several stocks, e.g. big rise in technology stocks. It has been shown that CA is able to represent both stock and day clusterings meaningfully [14]. We will build upon that basic mapping, and take the next step of using the resulting CA space to set up our data driven peer groups.
Computing the CA Space for Returns

Recall $X$ is our $n \times p$ matrix of returns. To compute the CA space, we first need to shift $X$ so that all values of $X$ are non-negative. We do this by updating

$$X \leftarrow X - \min(X).$$

The choice of $\min(X)$ as the adjustment factor was shown to have insignificant effect on the mapping [14]. Let $M (p \times p)$ and $N (n \times n)$ be diagonal matrices with entries the inverse of the column and row sums of $X$ respectively and let $q := \min(n, p)$ be the effective dimension of $X$. We want to find $p \times q$ eigenvector matrix $U$ and $q \times q$ diagonal eigenvalue matrix $\Lambda$ such that

$$X'NXMU = U\Lambda$$

and

$$U'MU = I.$$

This can be achieved, for example, by the usual eigen-decomposition operations on the positive semidefinite matrix $M^{1/2}X'NXM^{1/2}$. Similarly, we want to find a corresponding $n \times q$ matrix $V$ such that

$$XMX'NV = V\Lambda$$

and

$$V'NV = I.$$

It is easy to check that solving one problem will lead to the other, using the following equations:

$$V = XMU\Lambda^{-1/2}$$

and

$$U = X'NV\Lambda^{-1/2}.$$

Finally, we arrive at our CA coordinates:

$$C_{stock} = NXMU$$

and

$$C_{day} = MX'NV,$$

$n \times q$ and $p \times q$ matrices representing the $n$ stocks and $p$ days respectively. Each row of $C_{stock}$ is a $q$-vector representing a stock’s coordinates in the CA space. Similarly, each row of $C_{day}$ is a $q$-vector representing a day’s coordinates in the same space.
CA Peer Groups

In the CA space, stocks with similar history are close to one another. We quantify this similarity by looking at the Euclidean distances between the stocks. Based on these distances, we could cluster the stocks into groups and use them as “data-driven” industry sectors and proceed with peer group residuals as before. However, there is no need to restrict ourselves to discrete groupings. Each stock can include every other stock in its peer group and simply weighting them by their proximity. This gives the peer groups more flexibility to reflect the uniqueness of each stock, instead of having just one aggregate representing every stock in a peer group.

Mathematically, for each stock, we construct a peer group of stocks which is a weighted average of all other stocks. The weights used are the normalized inverse squared Euclidean distance. That is, we have an \( n \times n \) weight matrix \( W \) with 0 down the diagonal and entries

\[
w_{ij} = \frac{1/||C_{stock}^{(i)} - C_{stock}^{(j)}||^2}{\sum_j 1/||C_{stock}^{(i)} - C_{stock}^{(j)}||^2},
\]

for \( i = 1, \ldots, n, j = 1, \ldots, n, i \neq j \). The peer group residuals matrix is then simply

\[ R = X - WX. \]

We do not use all \( q \) dimensions of the CA space, but only a subspace. Firstly, this gives us a more compact representation of the stocks. Secondly, it is a natural way to regularize our data to reduce the impact of noise. We ignore the first dimension since it is just a shifting constant. We then take the next 15 dimensions. This choice is to reflect the eigenvalue distribution of the CA mapping, the diagonal entries of \( \Lambda \). Figure 1.1 shows the CA scree plot for 2003 returns. Note that the scree plot will vary with different dates and different returns window \( k \), but the plot tapers off before 15 almost always. This also coincides with the industry-standard 10 to 15 components in modeling stock returns. Also, a property of the CA space is that the squared \( L_2 \) norm of the coordinates scale with the eigenvalues. Hence, the exact number of dimensions we use is not a huge issue since we are truncating the axes that will have little impact on euclidean distance.

Even though the work presented here will only use the stock coordinates, the dual-coordinates allow us the flexibility to incorporate information from the days in future work, e.g., clustering days and using different day-state models.

1.3.3 Principal Components Analysis (PCA)

Given that we are using only the stock coordinates of the CA map, it is natural to ask why we do not simply do PCA. PCA has a more natural interpretation of looking at maximal variance directions. It is clear that CA is somewhat just a stretched PCA, and we would
Figure 1.1: CA scree plot for 2003 1-day returns of our 1317 stocks. We use the first 15 dimensions, demarcated by the vertical line.
CHAPTER 1. PREDICTING THE U.S. STOCK MARKET

like to understand how close the two methods are with respect to our data. To be clear, the corresponding PCA map is finding $U$ and $\Lambda$ such that

$$X'XU = UA$$

subject to

$$U'U = I$$

and

$$C_{stock} = XU.$$  

Note that this is equivalent to our CA mapping if the diagonal entries of $N$ and $M$ are constant. We can check this by looking at the coefficient of variation, that is the ratio of the standard deviation and mean. This again depends on the training period and returns windows, but in checking the various setups we use across different sections of our data set, all the coefficients of variation are much smaller than 1. For example, for the 2003 1-day returns we saw above,

$$\frac{sd(N_{ii})}{mean(N_{ii})} \approx 0.002$$

and

$$\frac{sd(M_{jj})}{mean(M_{jj})} \approx 0.02,$$

where the mean and standard deviations are over $i = 1, \ldots, n$ and $j = 1, \ldots, p$ respectively.

There is another subtle point. In the above equations, $X$ has been ‘treated’ in both PCA and CA before the equations are applied. For PCA, we center the columns to get the maximum variance interpretation. For CA, we added the minimum to make the whole $X$ matrix non-negative.

Thus, it is unclear whether the centering and shifting have a tangible effect on the mapping, even when $N$ and $M$ are constant. To investigate this, we look at the projections of some stocks on our CA and PCA space and compare them. As before, we use the 15 dimensions corresponding to the 15 largest (non-trivial) eigenvalues in both spaces. Figure 1.2 shows a stock in the first 100 days of the training period: the CA and PCA projections are remarkably close, relative to the original stock returns. This is typical for most stocks.

For our data, the CA and PCA mappings are very close. This gives us the best of both worlds. PCA is a well understood method for regularizing noisy data. It retains the maximal variance that can be captured with 15 dimensions. At the same time, the CA interpretation leaves the door open for future work incorporating the day coordinates in CA space.

1.4 Methods

In this section, we will explore various techniques that we use to predict residuals. As indicated above, we use predictive correlation as our benchmark. In each instance, we will make residual predictions for 20 days following the training period. This mimics the practical
Figure 1.2: 1-day returns and their projections of the stock 'MEDI' onto 15 dimensional CA and PCA spaces.
implementation of making daily predictions but updating the trading model monthly. We compute the correlation between the residual predictions and the realized residuals and take the average across the 20 days as our performance measure.

Let $X_{\text{test}}$ be the 20 columns of stock returns following $X$. We train our residual model with $R = X - WX$ where $W$ is derived from the CA mapping of $X$ as described earlier in Section 1.3.2. We then test our predictions against

$$R_{\text{test}} = X_{\text{test}} - WX_{\text{test}}.$$ 

We do this in a rolling manner: the columns of $R_{\text{test}}$ are made available as input to the model as we predict the next column (residuals for the next day). Note that $W$ is based only on the original training data. We are trying to predict future residuals based on the peer groups defined in the training period; it would make little sense to predict future residuals based on as yet undefined peer groups.

1.4.1 Mean Reversion and Sign Switching

Central to the residual framework is the assumption that mean reversion occurs. When a stock has a large positive residual (outperforms its peers significantly), we expect the market to correct itself, driving the stock to have a negative residual the following day. Similarly, a stock that underperforms significantly relative to its peers, will be expected to “catch up” in the near future. This leads to a simple prediction strategy of ‘switching signs’: predicted residuals are simply the prior day residuals with the signs switched. While stocks can certainly outperform its peers ‘legitimately’ with no market correction expected to follow, the hope is that in our large universe of 1317 stocks, the loss in these cases can be overcome by gains from the other cases.

There are still issues with this approach. We do not expect every stock to exhibit mean reversion entirely on the following day. Different stocks may exhibit different lengths of mean reversion and thus the success of the strategy depends on analyzing the residuals at the right returns window $k$. Related to this, we also do not expect every stock to exhibit mean reversion as frequently. For example, a more volatile stock will present more opportunities for mean reversion to occur. There is another factor at play: the suitability of the peer group as a proxy for a stock’s expected value. For any model, there will be peer groups who do better in accurately representing their stocks than others.

All this boils down to selection of stocks for sign switching. We want to select stocks which under our CA peer group construction, has residuals that exhibit strong mean reversion. To do this, we look at the autocorrelation of each stock’s residuals, noting that negative autocorrelation reflects mean reversion. For residuals based on $k$ day returns, the most negative autocorrelation typically occurs at lag $k$. This is not surprising: we expect positive autocorrelation for lags less than $k$ due to the overlapping returns windows, and any autocorrelation to dilute as we increase the lag. Hence, we will use the lag $k$ autocorrelation
as an indicator for the strength of mean reversion. In particular, we will rank a stock’s suitability for sign switching by how negative its lag $k$ autocorrelation is.

For the sign switching strategy to work, the negative autocorrelation must persist past the training period. Empirically, this is more pronounced for the higher ranked stocks: those with larger negative autocorrelations within the training period continue to exhibit large negative autocorrelation. To illustrate this, we look at 1-day returns across our data set. For each 100 day period, we compute the CA residuals, and look at the lag 1 autocorrelations of the residuals for the next 20 days (outside the training period). We compare the mean of these autocorrelations for all 1317 stocks to that for just the 100 stocks with the most negative autocorrelations in the training period. Figure 1.3 shows the mean autocorrelations across the 100 day periods in our data set. It is clear (with just one outlier) that our ranking captures the strength of mean reversion.

However, this poses a new problem: with no benefit of hindsight, how far down the ranks do we include in our sign-switching strategy? Our answer here is that instead of making this 0-1 judgement, we weight our predictions by their autocorrelations. This forms the base case of our regression methods.

### 1.4.2 CA regression

To simplify notation, we will focus on the $k = 1$ case, but the obvious extensions to other values of $k$ hold. The selection of stocks for sign switching is akin to estimating a future residual $r_{i,j+1}$ by

$$\hat{r}_{i,j+1} = -I(\text{corr}_i < c) \cdot r_{i,j},$$

where $I$ denotes the indicator function, $c$ is some fixed threshold, and $\text{corr}_i$ is the autocorrelation of stock $i$’s residuals in the training period, $j$ is the index for days spanning the concatenated matrix of $R$ and $R_{\text{test}}$. Note that making a zero residual prediction equates to having no information on the residual’s future movement.

As alluded earlier, instead of making a choice for $c$, we could instead simply use the estimator

$$\hat{r}_{i,j+1} = \text{corr}_i \cdot r_{i,j}.$$

This of course is simply the least squares estimator from regressing each stock’s residuals on the 1-lagged version of itself, a natural mathematical interpretation of the mean reversion effect.

With the regression framework in place, we explored terms on suggestions based on prior experience and arrived at the estimator

$$\hat{r}_{i,t+1} = \beta_1 (\text{corr}_{i,t} \cdot r_{i,t}) + \beta_2 r_{i,t} + \beta_3 \left( \frac{\sigma_{st,i,t}}{\sigma_{lt,i,t}} \cdot r_{i,t} \right) + \beta_4 r_{i,t-1},$$

where $\sigma_{st,i,t}$ and $\sigma_{lt,i,t}$ are the short and long term volatility of residual $i$ as estimated by its standard deviations on the most recent 20 days and length of training period respectively.
Figure 1.3: Mean of lag 1 autocorrelations of residuals past the training period for all 1317 stocks vs. for just the top 100 stocks ranked by negative autocorrelation in the training period.
$\beta_1, \ldots, \beta_4$ are the least squares coefficients estimated after plugging in all the other empirical values. The second term tweaks our model towards a pooled general mean reversion effect. The third term adds flexibility to that general effect by taking into account recent changes in volatility. The fourth term adds an extra lag term. Other terms featuring combinations of the above and further lag terms were considered but dropped as they were not consistently statistically significant across the different time points in our data set.

The first term remains the dominant effect, with $\beta_1$ often close to 1. Nevertheless, using the expanded four term model improves the predictive performance on average, as measured by our correlation score as described in Section 1.1.1. Figure 1.4 shows the average predictive correlation for consecutive 20-day testing periods through our data set. That is, at each point, we use the prior 100 days to train the model, and use that model to predict for the following 20 days before updating the model. We see that the regression models outperforms the sign switching strategy considerably, while the full regression improves upon the base regression more often than not. The average predictive score across the entire data set are 0.0218, 0.0313, and 0.0325 respectively.

As can be seen from Figure 1.4, the predictive performance varies a great deal across different time points. Naive computations of standard errors of the performance scores based on assumptions of independence in the above case works out to be around 0.002, larger than the improvement attained from adding the extra three terms to the regression model. Of course, this is erroneous since the models operate under the common mean reversion principles, leading to strong positive associations between the predictive scores. Looking instead at the paired differences at each time point of the scores between the base and full regression models, the mean difference is actually around twice its standard error: evidence that the full regression model is superior to the base regression model.

### 1.4.3 Building on Regression

The above regression models were fitted with the most vanilla of methods: the classical ordinary least squares (OLS). However, many of the classical assumptions are clearly violated: non-independence of data, heteroscedasticity, presence of outliers. Here, we discuss a few of the different statistical methods we applied (under the same model) in hope of improving our predictions.

$L_1$ Regression

99% of the returns in our dataset has absolute value less than 0.1. However, there are returns that are as large as 0.9. These outliers have strong influence in our model fitting using the usual $L_2$ loss. Yet, they are unlikely to appear again in the near future, hence negatively affecting our prediction performance. We use $L_1$ regression (median regression) which is known to be more robust to outliers. Indeed the results were encouraging. There
Figure 1.4: Predictive performance comparison of the three models. Sign-switching simply estimates the next residual as the negative of current residual. Base regression weights the predicted residuals with the training autocorrelation, and full regression includes all four terms.
was marked improvements in our prediction scores across our different models, pushing our best scores from 0.0325 to 0.0347.

**Clustering Stocks**

The regression model gives very little flexibility for different stocks to behave differently. To allow for more flexibility but at the same time not overparameterize, we applied the regression model separately to clusters of stocks. Here, we turn to our CA coordinates for the clustering. We use k-means with 10 clusters in the CA space. If the clusters were too small ($\leq 5$), we reduce the number of clusters iteratively. We also combine this clustering method with $L_1$ regression. Figure 1.5 shows the boxplots for the predictive correlations of the various methods.

The scores for adding just clustering did not improve. However, there is a gain when combining clustering with $L_1$ regression. By applying $L_1$ regression on each of the clusters of stocks separately, we further improve the average score from 0.0347 to 0.0351. The difference may seem small, especially compared to the spread of scores, but it is statistically significant comparing it to the spread for the difference of scores.

**Regularization**

We tried both Lasso and Ridge regression (adding $L_1$ and $L_2$ penalty on the coefficients respectively). It turns out though that in almost all cases, the best prediction scores in the family corresponded to the unpenalized (OLS) estimates.

**Expert Knowledge**

Along the way, there were many choices to be made on parameters that were not central to our ideas and methods, e.g., training length, model update interval, number of CA dimensions to be used and number of clusters. While some experimenting did provide some guidance for each of these parameters, it was computationally prohibitive to attempt to optimize them simultaneously. We instead relied on expert knowledge to make these choices for us. These choices often turn out to be local optima. (It is equally prohibitive to verify global optimality.) For example, we experimented with different number of CA dimensions from 5 to 50, and found that 15, the suggested choice, gave the best performance scores.

There were also cases where adjustments were made based on experience. For example, we discarded extreme daily returns of 30% or more. This was done as 30% daily returns were very unlikely and probably due to recording error. We also thresholded other values over 12% down to 12% as a compromise between potentially discarding significant data and allowing an error to exert too much influence on our predictions. These adjustments increased the prediction scores across all the methods. (They are only applied on the training data and not the testing data.)
Figure 1.5: Predictive performance comparison for the various methods. The latter 3 methods are applied to the full regression model. “1clust”, which is $L_1$ regression coupled with $k$-means clustering achieved the highest mean score.
The above remarks serve as reminders that expert knowledge, while not a statistical method, can and should be used in conjunction with other methods to improve predictive performance.

1.5 Conclusion

In this chapter, we made an attempt at predicting the U.S. stock market. In particular, we investigated statistical arbitrage strategies and more specifically prediction within the context of the residual framework. We utilized correspondence analysis to show that data driven methods can extract meaningful results even when we ignore meta information like industry sectors. We built a mean reversal regression model and showed that it was able to score well in forward prediction across the duration spanned by our data set. Further, we highlighted how statistical methods like the more robust $L_1$ regression and the data-driven $k$-means clustering can help improve these predictions. One area of interest for future work is to explore ways to take advantage of the duality between days and stocks set up by CA.
Chapter 2

Analysis of Enterprise Telephony Systems Logs

2.1 Background

In this chapter, we analyze large amounts of data generated by large enterprise telephony systems. We aim to detect, and in some cases, predict system anomalies. To make sense of these large amounts of unstructured log files, we employ similar techniques from dealing with high dimensional data like clustering and normalization. We also see, just as in Chapter 1, expert knowledge is complementary to these high dimensional tools.

2.2 Introduction

Large complex telephony applications, like enterprise Voice-over-IP (VoIP) systems, call-centers, and contact centers, require high reliability and availability in order to prevent any downtime that would lead to loss of business, loss of revenue and in some cases, loss of life. Most of these systems are already designed with very high availability in mind. However, as with any complex software system, bugs, misconfiguration, and other operator problems, do cause occasional failures and bring the system down. When a downtime occurs, as part of recovery procedure the system administrators and the service support staff may look at the system trace and debug logs to identify the cause of the failure. The current process of exploring the log file is fairly manual in nature. A large software system may consist of a number of modules operating in a multi-threaded environment, with each module writing its debug and trace messages to a log file. We believe that the system logs (both debug and trace logs) depict the state of the system fairly accurately. They may allow us to detect and

---

1This chapter is extracted from a published article [28]. It is joint work with Navjot Singh and Shalini Yajnik from Avaya Labs, Basking Ridge, New Jersey.
sometimes predict, the occurrence of anomalous behavior and take timely action to prevent future downtime.

Study of system logs to characterize system and user behavior has been a focus of research for some time. Web logs containing user search patterns and navigation behavior [15] have been studied to improve web site usage and also to provide users with targeted product advertising. Syslogs, transaction logs and error logs have also been mined for detecting failures and/or anomalous behavior in a system [41, 25, 29]. Log visualization [13] has been another area of fertile research, where abstract visualization of log file has been used as a tool for determining system state.

Early work on the study of logs focused on statistical modeling of errors and failure prediction based on the models. Iyer et al [43] noted that the alerts in the DEC VAX-cluster systems were correlated and based on this observation developed models for behavior of these systems. Work by Nassar and Andrews [36] showed that there was an increase in the rate of non-fatal errors before the system went into a failure mode. Data mining approaches like frequent pattern mining and sequential pattern mining have also been used to detect commonly occurring event patterns before a failure [31, 30, 46]. The paper by Vaarandi [46] clusters event logs based on the message content and then detects anomalies by finding deviations from these clusters. The work by Oliner and Stearley [37] describes the problems encountered in studying large logs generated by supercomputer systems.

Despite the fact that the system trace and debug logs contain a wealth of state information, they have been an under-utilized resource. This may be due to the highly unstructured nature of the contents of such logs. Most of the previous work on log analysis is based on searching for and mapping a set of predefined textual patterns in the log. In contrast, trace/debug logs are usually generated by system developers for testing and in-field debugging and hence may not have any pre-defined textual patterns that can be monitored. Indeed, the problem there is really to detect patterns which depict aspects of system behavior, irrespective of the textual content of the individual log messages.

The main focus of this paper is to use a combination of data mining and statistical analysis techniques on the logs obtained from large enterprise telephony systems and show how such techniques are very useful in detecting and in some cases predicting failures and anomalies in the system. Currently, system administrators and service support personnel rely on experience and expert information to extract useful information out of the log files. As an example, a service support personnel at a customer site may manually browse through the log file to detect the cause of a system crash. Our eventual goal is to quantify and automate most of these skills to make evaluation and predictions based on the log files more efficient. The techniques that we present in this paper form the basis for an automated tool that we are currently in the process of building.

The rest of the chapter is organized as follows. Section 2.3 gives an overview of the enterprise telephony communication application that we studied. Section 2.4 explains the goals of the analysis. Section 2.5 discusses the techniques that were used to reduce the data through preprocessing. Sections 2.6 and 2.7, respectively, apply data mining and statistical
2.3 Enterprise Telephony System

Avaya is a vendor of enterprise voice communication equipment and applications. An enterprise telephony system (Figure 2.1) consists of several components which communicate with each other to provide the telephony service to end-users. These include the Communication Manager (CM), the gateways, and a number of IP-capable end-points. The CM implements the bulk of the telephony features. It performs various tasks such as phone authentication and registration, call routing, call signaling, and, call initiation/termination. In a typical large deployment, a single CM is capable of supporting hundreds of thousands of IP phones with loads of similar scale in terms of peak busy hour calls.

![Figure 2.1: An Enterprise Telephony System](image)

From a software point of view, the CM is an extremely large and complex piece of software with millions of lines of code that have evolved over a decade of development. During operation, the CM comprises of more than 50 processes that handle the computing and communication tasks described above. Each process writes trace and debug messages into a shared CM log file. Below is an example of the log generated by two processes - proc1 and proc2. The log follows the pattern of $\text{Date}\!:\text{Time}\!:\text{Sequence}\!:\text{ProcessName (PID)}\!:\text{Priority} \!:\! \text{[Payload]}$. The \textit{Payload} part of the log consists of trace and debug messages generated by each process.
As one would expect from a debug/trace log, the payload part of each line does not follow any strict pattern and can be classified as unstructured.

2.3.1 Log Collection

As mentioned in Section 2.2, the motivation for this chapter was to understand the failure behavior of CM deployments in the field. So, the first order of business was to collect the debug/trace logs from live systems. Due to time and bandwidth constraints imposed by the customer environment, the log size was restricted to about four hours during failure of which pre-failure log was approximately three hours. We retrieved 714 log files from 460 live CMs that spanned 23 different releases. Each of these log files had a failure that caused a system restart. We applied our techniques to all 714 logs to study their behavior. The corresponding analysis and its results are described in Section 2.6. Additionally, we extracted log data for 108 days from a single system where we had no prior knowledge of failures. The analysis of this data is presented in Section 2.7.

2.4 Analysis Goals

The initial focus of our study was to understand whether any useful information can be gleaned from the debug/trace logs of CM. We had three main goals in mind while doing the analysis.

Signature of a Log: The first question to ask was whether we could determine the signature of a log file under normal operating conditions. Systems vary widely in terms of size and functionality. In turn, the log files from different systems can differ greatly. We define the signature of a log file by (a) the type of messages that occur in the log, (b) the frequency of each message type, and (c) the distribution of the message types over time. As an example, a particular CM deployment may have periodic audit turned on and the signature of the resulting log should contain periodic occurrences of the audit messages. Note that identifying the signature of a log file under normal operation is also very important in identifying and predicting failures, as any deviation from the normal signature would be an indication of an anomaly or a failure.

Categorizing Failures: Often, the system logs contain descriptive information about a failure. This could be in the form of events leading up to the failure, and/or footprints from the recovery process. Each failure may have its own characteristic footprint. The ability
to automatically derive this footprint from the logs, will help the system administrator in troubleshooting failure conditions. 

**Predicting Failures:** The ultimate goal of the analysis is to be able to use these logs to predict and preempt failures. Previous studies have indicated that a system may go through a series of non-fatal errors before it really crashes [36]. There are many instances where a sequence of events directly leads to a failure in the system. By identifying the accompanying symptoms present in the log files, it is possible to predict upcoming failures reliably.

Analyzing the log data to achieve the above goals is a complex task. Some of the complexities of the analysis come from the following:

1. Log messages are often corrupted and/or lost, especially when the system is under failure conditions or under heavy load. Any log mining technique will suffer from the incorrectness that arises out of this.

2. The volume of data generated in the logs can be really large. Log messages are textual and any data mining tool will need to reduce the volume of data stored and processed, before the data can be handled in an efficient way.

3. The content of the log messages is very domain/application specific and detailed knowledge of the application and domain is needed to determine if a particular behavior is really anomalous. Any general purpose log analysis technique cannot deal with the application-specific nature of the logs.

Unfortunately, it is difficult to avoid the first problem of lost messages. Logging is usually the lowest priority task and an overloaded system may delay and/or lose log messages. However, our study indicates that there are enough indicators in the log file and losing a few has minimal impact on our ability to detect problems. In the next section, we discuss log preprocessing techniques that aim to purge the corrupted messages that arise out of the first problem and tackle the second problem through data reduction. Solution to the third problem is proposed in Section 2.6.4.

### 2.5 Log Preprocessing

In the CM application, several processes and threads write to the same log file. Even with locking, there are cases where the log messages are either partially written and/or corrupted. The first preprocessing step was to cleanup the log file by removing all corrupted messages and unprintable characters.

As shown earlier, the format of entries in the log file was $Date:Time:Sequence:ProcessName (PID):Priority:[Payload]$. By manually looking through the log file, we realized that most of the semantic information in the message was stored in the process name and the payload
field. We define the message formed by combining these two fields \textit{ProcessName:Payload} as the \textit{Effective Message}. Henceforth, when we refer to \textit{message}, we mean the effective message.

In order to determine the signature of a log file with respect to various message types, the first order of business was to identify all unique messages in the log files. We extracted out the effective message part from all entries of each log file and built a set of unique messages across all 714 logs. From the set of all 11.5 million messages in the logs, there were about 2.5 million such unique messages. In statistical terms, defining the signature of a log in terms of 2.5 million messages is like studying a 2.5 million variable space. This is not only very computationally expensive but may also lead to detection of no event patterns in the logs, since a large fraction (\sim 22\%) of the messages are unique. Therefore, before any analysis technique could be applied, we needed to reduce the unique message set through some form of clustering.

\subsection{2.5.1 Message Clustering}

Note that the payload part of the message and hence the effective message comes from debug/trace messages written by the system developers. Such messages are usually composed of a text string with some parameters. The parameters could be IP addresses, memory locations in the code, or any other generic alphanumeric strings.

Each invocation of a given message in the code may print out in the log file as a different message, depending on the values of the parameters in the message. One approach that could be taken to reduce the number of unique messages in the data set is to de-parameterize the messages and cluster similar messages together. The simplest and most coarse-grained de-parameterization technique that we used in the first round of analysis was to replace the following with generic tokens: (a) IP/Ethernet addresses, (b) memory locations, (c) all hexadecimal digits. Once this was done, we then clustered the messages based on Levenshtein distance [27]. This simple combination of de-parametrization/clustering reduced the number of unique messages to nearly 13,000 which was about 0.5\% of the original message set. This is a more manageable set of information and the ratio of the unique messages to the total number of messages is small enough to detect possible patterns in the logs.

Note that there is a trade-off between the granularity of de-parameterization/clustering and the amount of information retained. A general purpose clustering technique like the one we used leads to a higher loss of information. However, the size of the unique message set is reduced significantly and becomes easier to analyze. After the first round of analysis, custom clustering/de-parameterization techniques (determined through domain-specific knowledge) can be applied to important subsets of messages to retain more information during the analysis.
2.5.2 Message Normalization

Storing textual messages in memory during analysis is very resource intensive and can slow down the analysis significantly. To avoid this problem, we normalized our message set by assigning unique numerical identifiers to each message in the set of all 13,000 unique messages. The techniques for assigning identifiers to messages has been discussed in our earlier work [47]. Henceforth, we will refer to these numerical identifiers as message codes or just codes. We pre-processed each of our 714 log files, such that each line of log was converted to a timestamp and a numeric code.

Next, we discuss the analysis of these pre-processed logs files.

2.6 Analysis for Logs with Known Failures

Our study was targeted towards two types of data sets. In the first data set of 714 logs, each log was retrieved from a system after a crash failure and had a distinct failure marker. These logs contained three hours of data before failure and one hour after failure. There were different types of crash failures in the logs and each was identified with the presence of its corresponding message code in the log. Our analysis for these logs was aimed towards finding a common signature across all failures of the same type and determining if the failure can be predicted through a build-up of messages. Note that even though we applied our techniques to all 714 log files, results from a sample set of log files presented here are enough to showcase the validity of our techniques.

2.6.1 Frequent Itemset Mining

The first effort at analyzing the pre-processed logs was to detect common pattern across log files with similar types of failures. We wanted to find item sets that were commonly associated with each failure code. The hope was that the presence of these item sets would predict or at least characterize the failure. We grouped the logs based on their corresponding failure codes and mined for frequent item sets within the group. The high support frequent item sets turned out to be codes associated with the failure and recovery process. Our results show that frequent itemset mining was a useful approach for categorizing failures as it indicated the set of common messages that occurred during the failure of a certain type. However, it did not help in predicting failures.

As one may expect, the failure and recovery messages mostly occur just before and immediately after the failure. To eliminate them from our prediction analysis, we cut the logs just before failure, and perform frequent item set mining on the shortened log. We found that codes 4514 (related to process errors) and code 4597 (generated from the maintenance system) were the most common codes before failure.
Unfortunately, their presence alone did not make good indicators of upcoming failure as they were fairly prevalent codes across all log files. However, the results here did prompt us to look deeper into these codes in subsequent analysis. This is discussed further in Section 2.6.4.

![Graph showing overall message frequency analysis](image)

Figure 2.2: Overall Message Frequency Analysis

### 2.6.2 Overall Message Frequency Analysis

In our earlier work [47], total message frequency (i.e. total number of message across all codes per unit time) was analyzed, showing some promise of information. This technique relied on the fact that processes become more chatty as the failure approaches, especially in the cases where there is a slow build up of failure in the system. As an example, Figure 2.2 shows the overall message frequency plotted against time. Message frequency was obtained for each 1 minute interval. The graph shows a steady increase in the message frequency before the failure at the 180 minute mark. In many cases simple visualization of the overall message frequency graphs helped in identifying common failures across log files.
2.6.3 Individual Message Frequency Analysis

Taking the next step, we looked at the message frequencies of individual codes in a log file. We wanted to find codes that showed peculiar trends before failure. As explained before, we focus our attention on the part of the log just before failure. We computed message frequencies for 1 minute intervals for each code and plotted these for visualization. A sample graph is shown in Figure 2.3(a). This is a graph of individual message frequencies for the same log file shown in Figure 2.2. There are 29 different message codes and except for a few dominant codes, it is difficult to visually determine any pattern in the other codes. We encountered similar difficulties across other log files where the number of codes was too large to explore. Therefore, we needed a mechanism to filter out only the “interesting” codes.

We implemented 3 types of automated filters to pick out the codes to plot: slope, window-max, and window-rate filters. These filters were targeted to bring out the codes that show some form of trend or anomaly before the failure.

**Slope Filter:** The slope-filter was the most natural. We are interested in identifying codes that steadily build up towards the failure. We simply ran a linear regression on the individual message code frequencies and plotted only the interesting codes with slope higher than a specified threshold.

The slope filter captured codes 4514 and 4597. Figure 2.3(b) shows the slope-filtered graph for the same log shown in Figure 2.3(a). As can be seen from the graph, the number of “interesting” codes go from 29 down to 2. The graph shows that these two codes were largely responsible for the increased message frequencies detected in Figure 2.2. This behavior of codes 4514 and 4597 was observed in log files spanning across a number of systems. Hence, for such systems, to do failure prediction based on a steady build-up of log messages, we only had to look at codes 4514 and 4597.

**Window-Max and Window-Rate Filters:** The other two filters are based on the idea that there is a period of time just before failure during which the log behaves anomalously. Here we designate a specified window before failure as the window of interest and call it the pre-failure window. We would like to detect codes that had increased activity in this window. In our analysis, we used a pre-failure window of one hour. The window-max filter plots only the codes whose maximum frequency in the pre-failure window is a threshold-fold higher than the maximum frequency before the window. The rate filter plots only the codes whose average frequency is a threshold-fold higher than the average frequency before the pre-failure window.

The window-max and window-rate filters found more codes of interest than the slope based filters. It is worth noting that the two window based filters had similar results. This is due to the fact that a large number of codes only appear in the pre-failure window and not before. This is evidence that there is information, other than the failure code itself, in the log files describing the failure. By using the filters, we can identify these codes to help categorize the failure. For example, Figures 2.4(a),(b) show a set of codes obtained after applying the window-max filter on a log file from a system which had periodic failures over
a course of time. Each failure of the system was found to have a signature similar to Figure 2.4(a). Figure 2.4(b) shows only the pre-failure window from Figure 2.4(a). Looking at the corresponding messages in the log file revealed that the set of messages were caused by a number of phones repeatedly trying to register themselves with the CM (an anomaly condition). In another case, there was a set of 48 logs that displayed highly similar codes related to scheduled maintenance just before failure. The failure turned out to be maintenance triggering a crash. The two examples given above show how filters can be used to extract failure signatures from the log files.

### 2.6.4 Application Specific Clustering

The results in Section 2.6.3 prompted us to look deeper into codes 4514 and 4597 that show an increasing trend towards failure. Recall that during the log preprocessing stage, we had de-parameterized and clustered the original log messages. We inevitably lost some information from the logs but this was a necessary step to reduce the total number of messages.

Now, we introduce the concept of application-specific clustering, where we allow the de-parameterization and clustering to be tuned to the specific application logs. As an example, for code 4514 below, we tune the clustering to not de-parameterize the first and second NUM fields, i.e. we reinstate some of the parameters in the 4514 messages to recover some of this information.

**Original Clustered Message for code 4514:**
\[
\text{hmm:CM} \quad \text{proc} \quad \text{err: pro=(NUM),err=(NUM),seq=(NUM),da=...}
\]

**Application Specific Clustering:**
\[
\begin{align*}
\text{hmm:CM} & \quad \text{proc} \quad \text{err: pro=7171,err=200,seq=(NUM),da=...} \\
\text{hmm:CM} & \quad \text{proc} \quad \text{err: pro=7172,err=603,seq=(NUM),da=...}
\end{align*}
\]

This led to a breakdown of about 400 different sub-codes of the original code 4514. Extracting just these messages from the logs, we employed the techniques shown in Section 2.6.3 on this message set. The results showed that, while the codes may differ between releases, only a few codes in each release were responsible for the increasing trend found earlier. \textit{This allowed us to isolate the exact error numbers and processes that were associated with the pre-failure increasing trend.} Figure 2.5 shows the same log file as Figure 2.3 with slope-based filter applied on the sub-codes of code 4514. The graph shows only 2 of the 400 sub-codes were responsible for the increasing trend.

### 2.7 Analysis in the absence of failure markers

In the above analysis, we have assumed that we know when the failures occur and are able to mark out the failure times. Note that without this information, none of the automated filters described in Section 2.6.3 make much sense. However, even in the absence of these
failure markers, our techniques can still yield useful results. As an example, our second data set was a series of log files spanning a 108 days period, obtained from one CM deployment. Within this period, there were intervals when the network was very slow and impacted the quality of service provided by the CM. However, these times were not exactly marked in the logs through some known failure codes.

In the absence of failure markers, we first plotted the overall message frequency and then used our knowledge gained from the previous analysis to track the usual suspect codes (e.g. 4514 and 4597). Both of these plots indicated anomalous regions. The overall message frequency plot showed intervals of time (about 1 hour) where the message frequency increased tenfold from normal, indicating possible problems. We zoomed into those areas and identifying them as “failure” areas, used techniques discussed earlier to get to the interesting codes. By plotting these interesting codes across the whole 108 days period, we see that they indeed corresponded to the dates when the network experienced difficulties. Figure 2.6 shows the plot of the message frequency per hour for the anomalous codes. Large sections of log file (e.g. first 450 hours ~ 18 days) show normal behavior with sudden spikes in the codes where the network anomalies occur. Most of the anomalous codes were associated with large number of phones re-registering and slowing down the CM.

2.8 Conclusions

Despite the many innate difficulties in dealing with the CM log files, we were able to extract meaningful results. The key to our approach is to transform the chaotic log files into a standard form for visualization. Together with user feedback and simple analysis tools, visualization allowed expert knowledge to be efficiently applied to anomaly prediction, detection and categorization. Our analysis techniques led to the detection and categorization of several failure types and in some cases predicted trends which lead towards failures.

The techniques and analysis presented in the paper need not be restricted to CM log files or even general log files. Any indexed (time or otherwise) set of text files can be visualized the same way.
Figure 2.3: Use of Slope-based Filters reduces codes
CHAPTER 2. ANALYSIS OF ENTERPRISE TELEPHONY SYSTEMS LOGS

Figure 2.4: Window-Max Filter
Figure 2.5: Figure 2.3(a) with application-specific clustering of dominating code
Figure 2.6: 108 days of log without any explicit failure markers
Chapter 3

Statistical Stability

3.1 Introduction

3.1.1 Regularization Methods

There is an ever increasing amount of data in all fields of science and engineering. Often, this data comes in high dimensions relative to the sample size, posing a new challenge to scientists, engineers, and decision makers. These problems, plagued by the curse of dimensionality, suffer from overfitting when classical methods are applied. Regularization methods are used to tackle this problem of overfitting head on, usually by imposing a penalty on the complexity of the solution or through early stopping. For example, in fitting the usual linear regression model, the Lasso [44] and ridge regression [18] adds a $L_1$ and $L_2$ penalty on the coefficient estimates respectively to the usual least squares fit objective function. Regularization methods can also take the form of early stopping iterative algorithms like classical forward selection or $L_2$-Boosting [11].

Common to these methods is that they provide a family of possible estimators instead of just one estimator, with the unregularized solution at one end of the spectrum. This family is indexed by a regularization parameter and is commonly referred to as the solution path. For the Lasso and ridge regression, the parameter determines the extent of the respective penalties. For the iterative algorithms, the parameter represents the number of steps they take. Despite the difference in nature, numerous works have shown these regularization methods, at least in the context of the linear model, are intrinsically related [35, 12, 52]. In that light, we will not focus on the distinction between the different types of regularization parameters but instead simply use $\lambda$ as a catch-all representation for them. In the same vein, we focus on the Lasso in this chapter even though we believe the method we present will work in the general framework.
3.1.2 Selecting the Regularization Parameter $\lambda$

Much work has been done to show that regularization methods yield desirable solutions. For example, the popular Lasso has been shown to be $L_2$-consistent [11] and model selection consistent [51, 33, 45, 48] in the high dimensional setting when respective conditions are met. These results guarantee the existence of the $\lambda$ needed, but offer little guidance on how to find the desired $\lambda$ in practice. Indeed, selecting the regularization parameter $\lambda$ with guaranteed theoretical performance turns out to be a particularly difficult problem.

One can rely on traditional model selection criteria like Akaike’s information criterion (AIC) [1] and Bayesian information criterion (BIC) [42]. They are easy to compute but their validity rely on model assumptions. Furthermore, they are derived from asymptotic results, so even when model assumptions are satisfied, they may not work well in the finite sample case.

More commonly used today are model-free approaches like cross-validation (CV) and bootstrap methods. They have become computationally feasible for increasingly large data sets with the rapid advancements in computing power. These methods rely on data resampling to assess prediction error of candidate solutions and can be found in various statistics and machine learning literature [17, 8, 9, 10]. In particular, it can be used within regularization methods to select $\lambda$ (and the corresponding estimate). Doing so often leads to estimators with good predictive performance. However, there are other performance metrics that are also of interest in statistics, among them parameter estimation losses and variable selection consistency, with important practical connections. Unsurprisingly, optimizing predictive performance does not necessarily translate to having success with respect to these other performance metrics.

3.1.3 Statistical Stability

Statistical estimation is often tied to the optimization of a random function based on data. Take for example, when fitting a linear model for random variables $X \in \mathbb{R}^p, Y \in \mathbb{R}$, one might want to minimize the predictive $L_2$ loss,$$
 f(\beta) = E_{X,Y}(Y - X'\beta)^2.
$$

However, since the underlying joint distribution of $(X, Y)$ is unknown, we minimize the empirical loss$$
\hat{f}(\beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x'_i\beta)^2
$$

instead, where $(x_i, y_i)$ for $i = 1, \ldots, n$, are the observed samples of $(X, Y)$. By minimizing $\hat{f}$ instead of $f$, we incur a random estimation error dependent on the sample we observed. In the ideal scenario, e.g. when the sample contain independent and identically distributed observations and the sample size $n$ is large, this estimation error incurred is small. If we draw
multiple samples from \((X, Y)\), each resulting estimate will be close to that of minimizing \(f\), and consequently close to each other. This closeness across different samples can be seen as a form of stability in the estimation procedure, and we call it statistical stability.

It is clear that statistical stability is a necessary property for a reasonable estimation procedure: the solution is not meaningful if it varies considerably from sample to sample. This begs the question of whether the converse is true. Are estimates that vary least most meaningful? Are statistically stable solutions automatically good? This certainly cannot be true in general: an arbitrary constant estimate will not vary but is certainly meaningless. However, when we add a faithfulness or model fit requirement, we are able to devise a model-free criterion based on statistical stability for the selection of the regularization parameter \(\lambda\).

We show that our criterion \(SSCV\) provides a viable alternative to \(CV\) and \(BIC\). In particular, we compare the three approaches with respect to various performance metrics when applied to the Lasso on both simulated and real data sets. We find that our criterion compares favorably with \(CV\) and \(BIC\) where they are known to excel, and outperforms them in other scenarios over different performance criteria. In particular, we get excellent model selection results, both in simulations and our real data sets, where the results are validated by subject knowledge.

We note that previous works based on stability of solutions have shown positive results in terms of model selection \([9, 4, 34]\). The work here differs from them in two substantial ways. Firstly, we restrict our attention to selecting the regularization parameter. Even though we evaluate our choice by the performance of the corresponding solution, our focus remains on determining the right amount of regularization. Secondly, we do not introduce any further tuning parameters. Our aim is to prescribe a method for tuning regularization methods, adding more parameters, for us, is counter productive.

### 3.2 Methodology

#### 3.2.1 Lasso and Pseudo Solutions

Let \(X \in \mathbb{R}^{n \times p}, Y \in \mathbb{R}^{n}\) be our data set. The Lasso generates a family of solutions,

\[
\hat{\beta}[\lambda] = \arg \min_{\beta} \{ ||Y - X\beta||_2^2 + \lambda||\beta||_1 \}.
\]

\(\hat{\beta}[\lambda]\), as a function of \(\lambda\) is also known as the Lasso solution path for \(\beta_j (j = 1, \ldots, p)\). We want to select a solution from this solution path; that is, choose a \(\lambda\) and take its corresponding solution in the solution path. As alluded to earlier, we would like to make this choice based on statistical stability and fit.

Since the notion of statistical stability is tied to the sampling distribution of the data, it is unavoidable that we need multiple solution paths to make such an assessment. Of course,
it is often costly and infeasible to obtain extra data in practice. Thankfully, this problem is not new, and there are well-established ways to get around it. The key is to exploit the existing data by employing data perturbation schemes, parlaying it into multiple data sets. Let \((X^*[k], Y^*[k])\) represents our \(k\)th pseudo data set, derived from \((X, Y)\). In our case, these are the cross-validation folds: we randomly partition the data into \(V\) groups and form \(V\) pseudo data sets by leaving out one group at a time. (See Section 3.4.2 for other data perturbation schemes.) We then get pseudo solutions,

\[
\hat{\beta}[k; \lambda] = \arg \min_{\beta} \left\{ \| Y^*[k] - X^*[k] \beta \|_2^2 + \lambda \| \beta \|_1 \right\}
\]

for \(k = 1 \ldots V\).

### 3.2.2 Alignment

For many regularization methods, there are multiple representations for the regularization parameter \(\lambda\). In the case of the Lasso above, \(\lambda\) refers to the \(L_1\) penalty parameter. Other popular choices to index the solution path are the \(L_1\)-norm of the coefficient estimate, and the \(L_1\)-norm expressed as a fraction of the \(L_1\)-norm of the unregularized solution. Each of these representations for the solution path has its own merits, and is equivalent to the others (when non-trivial) for any single solution path.

However, care must be taken on how to most meaningfully align our solution paths, when we reference the same \(\lambda\) across different (pseudo) solution paths. In particular, when \(n < p\), the \(L_1\)-norm of the unregularized solution corresponds to the saturated fit and can vary a lot depending on which data points were sampled. This makes \(L_1\)-fraction a poor choice, as the same index may correspond to very different amounts of regularization. The effect is more pronounced when the features are more correlated. Figure 3.1 shows a histogram of the maximum \(L_1\)-norms for 10,000 bootstrap Lasso estimates of the base case Gaussian simulation in Section 3.3.1. There is considerable spread: in this case, the upper decile is more than 20% more than the lower decile.

To highlight the effect of alignment on estimation performance, we compared the performance of cross-validation with the three alignments for the low noise scenarios detailed in Section 3.3.1. As shown in Table 3.1, aligning the solution paths with \(L_1\)-fraction does comparatively worse than aligning with \(L_1\)-norm or the penalty parameter. Note that in the popular R package “lars” used in solving the Lasso efficiently, the included cross-validation code aligns with \(L_1\)-fraction.

For our method, we find that there is little difference in performance when aligning with either the penalty parameter or the \(L_1\)-norm. In this work, to allow a fair comparison, we opt to use the \(L_1\)-norm alignment, which worked best for cross-validation. That is, \(\lambda\) denotes the \(L_1\)-norm of \(\hat{\beta}[\lambda]\), the solution path.
Figure 3.1: Approximate bootstrap distribution of maximum $L_1$-norms of Lasso estimates on a typical simulated data set.

### 3.2.3 Convergence of Pseudo Solutions

Given the pseudo solutions $\hat{\beta}[k; \lambda]$ for $k = 1, \ldots, V$, we can compute the estimates

$$\hat{Y}[k; \lambda] = X\hat{\beta}[k; \lambda].$$

To evaluate stability, we need a measure for how far apart the estimates are at each $\lambda$: stable pseudo solutions should give similar estimates. One possibility is to look at the average pairwise squared Euclidean distance between the $V$ estimates:

$$A(\lambda) := \frac{1}{\binom{V}{2}} \sum_{k \neq j} ||\hat{Y}[k; \lambda] - \hat{Y}[j; \lambda]||_2^2.$$
CHAPTER 3. STATISTICAL STABILITY

Cross-Validation Estimation Error

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>Penalty</th>
<th>( L_1 )-norm</th>
<th>( L_1 )-fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.794</td>
<td><strong>0.787</strong></td>
<td>0.817</td>
</tr>
<tr>
<td>0.2</td>
<td>0.781</td>
<td><strong>0.773</strong></td>
<td>0.821</td>
</tr>
<tr>
<td>0.5</td>
<td>0.969</td>
<td><strong>0.956</strong></td>
<td>1.04</td>
</tr>
<tr>
<td>0.9</td>
<td>1.83</td>
<td><strong>1.80</strong></td>
<td>1.95</td>
</tr>
</tbody>
</table>

Table 3.1: Effect of alignment on cross validation performance.

It is not hard to see that this is equivalent to the more familiar “sample variance” formulation,

\[
\hat{\text{Var}}(\bar{\hat{Y}}[\lambda]) = \frac{1}{V} \sum_{k=1}^{V} ||\hat{Y}[k; \lambda] - \bar{\hat{Y}}[\lambda]||^2,
\]

where \( \bar{\hat{Y}}[\lambda] = \frac{1}{V} \sum_{i=1}^{V} \hat{Y}[i; \lambda] \).

Figure 3.2 shows two examples of this sample variance metric. The left panel is particularly illuminating: the pseudo solutions diverge as they grow at first but converge somewhat before diverging again. Here, convergence and divergence simply refers to the sample variance metric (which is really just the average pairwise distance) decreasing and increasing respectively. Heuristically, this behavior is exactly what one would expect if there is a “correct” amount of regularization. Different samples would take different paths towards the “correct” solution before moving away from one another due to overfitting. Hence, we propose to select the \( \lambda \) corresponding to the minimum point after the first negative slope. That is, we want to choose \( \lambda \) corresponding to the “dip”.

By doing this, we incorporate fit into our selection even though our criterion is based on stability. The convergence of the solution paths is key: not only does it suggest we are close to the truth, we are also gifted with statistical stability. Also note that this automatically excludes \( \lambda \)’s where the solution paths trivially agree.

However, this convergence effect is not always clear. The “dip” is not always present as shown in the example on the right panel. There you can still see the drop in gradient, but it is not clear which \( \lambda \) we should pick. Notice, however, that in a solution path, the norm of the solution varies with the amount of regularization (by definition in our case). Since larger solutions naturally varies more, using the sample variance metric skews the choice towards solutions with small norms. We need to do some form of normalization to account for this effect.

3.2.4 Hypothesis Testing and the Statistical Stability Metric

In hypothesis testing, a test statistic based on data is computed and its corresponding \( p \)-value is calculated by matching the test statistic with its model-specific theoretical distribution.
CHAPTER 3. STATISTICAL STABILITY

Figure 3.2: Examples of the sample variance metric. The left panel shows an example where the metric exhibits a “dip”, representing the “convergence” of the pseudo solutions. The right panel shows an example with a much muted “dip”.

This test statistic often takes the form of a mean value over its estimated standard deviation, e.g. the student’s $t$-test. The desired outcome for the $t$-test, as is often the case regardless of the assumed model and $p$-value computation, is to have thle test-statistic away from 0. The heuristic there is clear: if the hypothesized effect is real, the size of the mean value should be large compared to its estimated standard deviation.

In the same vein, our sample variance metric should be relative to the mean size of the corresponding solution squared. We define the statistical stability metric,

$$SS(\lambda) := \frac{\text{Var}(\hat{Y}[\lambda])}{||\hat{Y}[\lambda]||_2^2},$$

the normalized version of the sample variance metric. Figure 3.3 shows the corresponding $SS$ metrics in dashed lines superimposed on the old sample variance metric. On the left, the “dip” from the sample variance metric is preserved by the $SS$ metric. On the right, there is now a pronounced minimum we can select.

The $SS$ metric’s reciprocal has exactly the form of a test-statistic. We can view the $SS$ selection of $\lambda$ as a set of hypothesis tests. For each $\lambda$, we are testing if the fit ($\hat{Y}[\lambda]$) is statistically different from fitting the null model ($\hat{Y} = 0$), albeit without a specified theoretical distribution. Our $SS$ criterion of choosing the $\lambda$ corresponding to the convergence of pseudo solutions, is exactly choosing $\hat{Y}[\lambda]$ with locally minimal normalized variance. This
in turn, is exactly choosing the solution whose \( SS \) metric has the largest reciprocal, or in our analogy, the most statistically significant solution along the path.

### 3.2.5 Incorporating and Exploiting Cross-Validation

There is no guarantee that our \( SS \) metric would have only one local minimum. Note that unless the multiple solution paths match up perfectly, there will be a local minimum or multiple local minima. That is, we will still observe the convergence effect \( SS \) is looking for even if \( Y \) bears no relation to \( X \) at all. To prevent scenarios like this where \( SS \) fails, we incorporate cross-validation into our selection. We have already limited our choice of minimum \( SS \) to local minima. Here we further limit it to local minima before the cross-validation choice. This call this improved criterion \textit{statistical stability with cross validation} (\( SSCV \)).

We are exploiting the fact that cross-validation overselects [26, 49]. When there is a meaningful local minimum, cross-validation will likely overselect, and \( SSCV \) behaves just like \( SS \) above. However, when \( Y \) bears no relation to \( X \), or when the noise overwhelms the signal, cross-validation will likely choose the trivial solution correctly. In this case, \( SSCV \) will follow suit and pick up the trivial solution.

Note that this has negligible additional computation cost, as we are essentially getting the cross-validation choice for free. The bulk of the computation lies in computing the multiple solution paths we already have.
3.3 Experimental Results

In this section, we evaluate \( SSCV \)'s performance relative to the cross validation (\( CV \)) across a variety of problems. In each problem, we fit a linear model using the Lasso. We focus our attention on the comparison with \( CV \) as it is the most popular criterion in practice.

We start with simple sparse gaussian linear model simulations with our focus on the high dimensional data set up. We will tweak the simulation parameters such as correlation strength within features and signal strength, as well as explore popular correlation structures of the design matrix, to cover a wide variety of problem scenarios. We compare the solutions picked by \( SSCV \) and \( CV \) with regard to parameter estimation, prediction, and model selection performance measures. We also include the \( BIC \) choice, but note that it performs poorly as expected in our high dimensional setting.

We also explore the performance of our method on two real data sets from neuroscience and bioinformatics. We use a combination of objective predictive performance and subject knowledge on plausible models to illustrate the efficacy of \( SSCV \) over \( CV \). In all cases, note that we are comparing different choices of \( \lambda \) on the same solution path (from the original data). Furthermore, we use the same data splits to reduce the effect of sampling variation between our \( CV \) and \( SSCV \) choices. We use \( V = 8 \) cross validation folds to obtain the pseudo solutions.

3.3.1 Gaussian Simulation

Let \( X_i \in \mathbb{R}^p \) for \( i = 1, \ldots, n \) be independent identically distributed Gaussian variables with mean 0 and covariance \( \Sigma \). We have the usual linear model \( Y_i = X_i'\beta + \epsilon_i \), where \( \beta \in \mathbb{R}^p \) is the unknown parameter, and \( \epsilon_i \in \mathbb{R} \) is independent Gaussian noise with standard deviation \( \sigma \). \( \beta_j \) are drawn from \( U[\frac{1}{3}, 1] \) for \( j = 1, \ldots, 10 \) and 0 otherwise. The separation from zero is for model selection to make sense. This is a common assumption in theoretical work.

The reported estimation and prediction errors are defined as

\[
||\hat{\beta} - \beta||_2 \quad \text{and} \quad \sqrt{E_X(||X\hat{\beta} - X\beta||_2^2)} = \sqrt{(\hat{\beta} - \beta)'\Sigma(\hat{\beta} - \beta)}
\]

respectively. For model selection, we use the \( F \)-measure which balances false positive and false negative rates of identifying non-zero coefficients of \( \beta \). The higher the \( F \)-measure the better. Each simulation is repeated 1000 times and the performance measures are aggregated across them.

A Base Case

Within the Gaussian linear model setup, there are many problem scenarios that favor one method over others. In particular, the following problem settings are known to affect the performance of the Lasso: correlation strength between features, strength of signal (size of
coefficients) relative to the noise levels, dimension of the problem (p), and the correlation structure of the features. This is of course not an exhaustive list but is sufficient to cover a wide range of problems. As the strength of the correlation and signal are key to the behavior of the Lasso solution, we will include a full complement of these problem settings to illustrate when and why $SSCV$ works well.

We start with a base case scenario. Here, $\Sigma$ has entries 1 down the diagonal and constant $\rho$ on the off-diagonal. We vary $\rho = 0, 0.2, 0.5, 0.9$ and $\sigma = 0.5, 1, 2$. We set $n = 100$ and $p = 150$ to emulate the high dimensional data setting. Note that this implies that the columns of $X$ are empirically correlated even when the features they represent are independent.

As expected, $CV$ does well in terms of prediction error (see Table 3.2). However, observe that this does not necessarily translate to success in terms of other performance measures. With estimation error, we find that once we leave the orthogonal case $\rho = 0$ where estimation and prediction error are equivalent, $SSCV$ has lower estimation error than $CV$ despite having comparable prediction error.

For model selection, we use the $F$-measure, the harmonic mean of the precision and recall rates, which are inversely proportional to false positive rate and false negative rate respectively. A high $F$-measure is achieved when both false positive and false negative rates are low. Recall that we are selecting solutions from the same solution path. The Lasso solution path corresponds roughly to a nested family of models in terms of features picked since features seldom gets dropped as the we relax the penalty term. Hence, having a low false negative rate (high recall) typically comes at the cost of a high false positive rate (low precision). The $F$-measure balances these two objectives.

By this measure, $SSCV$ often outscores $CV$ by a considerable margin. $CV$ picks more true variables, but in the process picks up a disproportionately large number of noise variables. This is in line with theory that $CV$ often overselects [49]. $SSCV$ cuts down the false positive rate, but not too much at the expense of the false negative rate.

The results are summarized in Table 3.2 and the standard errors (SE) are given in Table 3.3. Note that the performance measures are highly correlated since for each simulation run, the selections by $SSCV$, $CV$ and $BIC$ are from the same solution path. Hence, the SEs for paired differences in performance measures are actually lower than the SEs for each of the values as reported in Table 3.3.

**Effect Of Ambient Dimension**

We repeat the simulations but this time for different $p$. Note that only the number of non-relevant features is changing; the number of non-zero coefficients remain at 10, the sample size $n$ remains at 100. The comparison of $SSCV$ and $CV$ from the base case extends here: $CV$ does well in prediction error, especially in the independent predictors case, but loses out to $SSCV$ in most other scenarios. The results are summarized in Table 3.5 and 3.6.

For the low dimensional case $p = 50$, we see that $BIC$ performance is much improved compared to the $p = 150$ case. This is to be expected since $BIC$ was developed to tackle
### Table 3.2: Performance of \( \text{SSCV} \), \( \text{CV} \) and \( \text{BIC} \) in picking the regularization parameter for the Lasso for our base case design: constant correlation, \( p = 150 \).

| \( \rho \) | \( \sigma \) | \( \text{SSCV} \) | \( \text{CV} \) | \( \text{BIC} \) | \( \text{SSCV} \) | \( \text{CV} \) | \( \text{BIC} \) | \( \text{SSCV} \) | \( \text{CV} \) | \( \text{BIC} \) | \( \text{SSCV} \) | \( \text{CV} \) | \( \text{BIC} \) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.5 | 0.470 | **0.399** | 0.632 | 0.470 | **0.399** | 0.632 | **0.673** | 0.402 | 0.225 | 19.7 | 39.7 | 78.7 |
| 0 | 1 | 0.885 | **0.787** | 1.27 | 0.885 | **0.787** | 1.27 | **0.594** | 0.413 | 0.225 | 21.7 | 37.8 | 77.4 |
| 0 | 2 | 1.53 | **1.47** | 2.29 | 1.53 | **1.47** | 2.29 | **0.434** | 0.403 | 0.207 | 22.8 | 29.8 | 58.3 |
| 0.2 | 0.5 | **0.386** | 0.391 | 0.687 | 0.373 | **0.360** | 0.623 | **0.535** | 0.440 | 0.223 | 27.4 | 35.4 | 79.6 |
| 0.2 | 1 | 0.757 | 0.773 | 1.36 | 0.719 | **0.711** | 1.23 | **0.522** | 0.445 | 0.223 | 27.8 | 34.5 | 78.3 |
| 0.2 | 2 | **1.37** | 1.43 | 2.56 | **1.31** | 1.33 | 2.34 | **0.475** | 0.414 | 0.206 | 25.6 | 31.6 | 73.4 |
| 0.5 | 0.5 | **0.465** | 0.479 | 0.871 | **0.341** | 0.348 | 0.624 | **0.493** | 0.456 | 0.222 | 30.6 | 33.8 | 80.0 |
| 0.5 | 1 | **0.927** | 0.956 | 1.72 | **0.682** | 0.695 | 1.23 | **0.485** | 0.445 | 0.214 | 29.9 | 33.6 | 80.2 |
| 0.5 | 2 | **1.62** | 1.67 | 3.18 | **1.21** | 1.23 | 2.30 | **0.427** | 0.396 | 0.19 | 26.2 | 29.5 | 74.6 |
| 0.9 | 0.5 | **1.02** | 1.04 | 1.92 | **0.330** | 0.339 | 0.615 | **0.466** | 0.444 | 0.211 | 31.1 | 33.2 | 80.4 |
| 0.9 | 1 | **1.75** | 1.80 | 3.55 | **0.572** | 0.587 | 1.14 | **0.396** | 0.377 | 0.183 | 26.9 | 28.9 | 75.1 |
| 0.9 | 2 | **2.46** | 2.58 | 5.50 | **0.858** | 0.861 | 1.81 | **0.274** | 0.262 | 0.158 | 19.0 | 21.3 | 58.3 |

Table 3.3: Standard errors (SE) for performance numbers in Table 3.2.
problems in the classical regime of $n > p$. It is interesting to note that the performance of SSCV and BIC are very comparable in this case. Indeed, in this scenario, they are very close to (empirically) optimal with respect to the $F$-measure. Of course, once we go to $p = 500$, BIC’s performance falls off the cliff. SSCV continues to beat out CV by a wide margin.

Other Correlation Structures

The constant correlation structure can be seen as a simple one latent variable model. Here we introduce other correlation structures corresponding to more complex models and run the same simulations. First, block correlation: all $p$ features are randomly grouped into 10 blocks, and within each block, the features have correlation $\rho$ while features from separate blocks are independent. Here, we let $\rho = 0.3, 0.5, 0.9$. Second, Toeplitz design: $\Sigma_{ij} = \rho^{|i-j|}$, with $\rho = 0.5, 0.9, 0.99$. In this case, the true variables indices are randomly distributed among the $p$ variables so that they are not all strongly correlated with each other. The results for the two designs are summarized in Tables 3.7 and 3.8 respectively.

Despite the different correlation structures, the qualitative results from the prior section holds again in both variations. For prediction error, CV almost always outperforms SSCV, but SSCV’s performance can be quite close to CV’s when $\rho \neq 0$. For estimation error, SSCV gains on and eventually outperforms CV with increasing correlation levels. And for model selection, SSCV almost always has a higher $F$-measure than CV. Also unsurprising, BIC continues to do poorly in these $p > n$ regimes.

3.3.2 fMRI Data

This data is from the Gallant Neuroscience Lab at University of California, Berkeley. In this experiment, a subject is shown a series of randomly selected natural images and the fMRI response from his primary visual cortex is recorded. The fMRI response is recorded at the voxel level, where each voxel corresponds to a tiny volume of the visual cortex. The task is to model each voxel’s response to the $n = 1500$ images. The image features are approximately 10000 transformed Gabor wavelet coefficients. We evaluate the prediction performance by looking at correlation scores against an untouched test set of 120 images with 10-13 replicates. There are 1250 voxels in all. We ranked them according to their predictive performance under a different procedure from a previous study [23]. Not all of them are informative, so we only look at the top 500.

We find that while the prediction performance are nearly identical for SSCV and CV, SSCV selects much fewer features. The results are in Table 3.4. For the sake of brevity, they are averaged across groups of 100 voxels. For example, for the top 100 voxels, on average, the correlation scores are similar, but SSCV selects 30 features compared to CV’s 70 features. That is, SSCV selects a simpler model that predicts just as well as CV. Figure 3.4 shows how close the correlation scores are.
Table 3.4: Performance on fMRI data set. The numbers are averaged across the respective hundred voxels.

We note again that SSCV picks fewer features than CV by design (Section 3.2.5). That being said, the reduction is huge here: SSCV picks less than half the number of features as CV across the different voxels. Furthermore, this was with little or no loss in predictive performance. To understand the results better, we look at the individual voxels and examine the features selected. In almost all the cases, SSCV selects a subset of the features selected by CV. This is because they both select from the same Lasso solution path and features are rarely dropped after being added to the solution as we relax the regularization.

Now, each feature corresponds to a Gabor wavelet characterized by its location, size and orientation. We plot the features selected by both CV and SSCV as well as the extra features selected by CV. The points in the plot represent the location and size of the Gabor wavelet selected. Figure 3.5 shows four randomly selected voxels.

We can see quite clearly that the features selected by SSCV are clustered in one area whereas the features selected by CV but not SSCV are scattered across the image. Biologically, we expect each voxel to respond only to a particular area of the visual receptive field. This confirms that the extra features selected by CV are not meaningful. Note that the location information of the Gabor wavelets were not used in fitting the model.

### 3.3.3 Cytokine Data

This data is from experiments performed by the Alliance for Cellular Signaling (AfCS), archived and made available at the Signaling Gateway, a comprehensive and free resource supported by the University of California, San Diego (UCSD). Pradervand et al. [39] from the Bioinformatics and Data Coordination Laboratory at UCSD processed and analyzed this data in an attempt to identify signal pathways responsible for regulating cytokine release. There are 7 cytokines, 22 signal pathway predictors. The signal pathways cannot be directly manipulated. Instead, ligands are stimulated to elicit responses from the signal pathway predictors and cytokines. For each cytokine, we have about 100 samples, each corresponding to average measured responses of the cytokine and signal pathways when a specific ligand pair is stimulated.
In the original study [39], principal component regression (PCR) is used to fit the data to a linear model and select the significant signal pathways. The selection is done by thresholding the estimated coefficients via a pseudo-bootstrap method. They do this for each of the seven cytokines. That is, they solve seven linear regression problems, each with $n \approx 100$ and $p = 22$, and apply thresholding to select the relevant signal pathways. These PCR results are then merged with other data and analysis to derive a final minimal model (MM).

We run Lasso with $SSCV$ and $CV$ on the seven linear regression problems and compare our results with the results from PCR and MM. Fig 3.6 shows the feature selection results for the four methods. We regard MM, which encompasses extra data and is not directly restricted by the linear model, as the benchmark for feature selection performance.

We can see from Fig 3.6 that Lasso with CV does poorly. It selects the most features
Figure 3.5: Feature selection by SSCV and CV on four randomly selected voxels. The “o”s represent features selected by both methods, while the “+”s represent features selected only by CV. The axes represent the pixel location of the images. The position and size of the points represents the wavelet location and size respectively. Note that most of the extra features CV select are scattered.
Figure 3.6: Feature selection results on cytokine data. The columns represent signal pathways predictors and each block of four rows correspond to a cytokine. The four rows within each block represent the selections of the four methods: the final minimal model (MM) and principal component regression (PCR) from the original study, and Lasso with SSCV and CV. The white squares corresponds to selected predictors.
for every cytokine, often by a large margin. Lasso with SSCV on the other hand, selects the same or slightly larger number of features than MM. Moreover, with the exception of cytokine \( TNFa \), SSCV always includes the features PCR selected which survived to the minimal model. In the case of \( TNFa \), PCR barely selects (close to threshold) the one feature that SSCV missed. SSCV in general selects only about half the number of features PCR selects. There are far fewer false positives with respect to MM. At the same time, it rarely misses out any of the important features that PCR picked up.

### 3.4 Discussion

#### 3.4.1 Choice of Stability Metric

Our SS metric is based on assessing the stability of the fitted values \( \hat{Y}[\lambda] = X\hat{\beta}[\lambda] \) instead of the estimates \( \hat{\beta}[\lambda] \). This seems counter-intuitive since we are interested in a variety of performance measures, most of which are based on the quality of \( \hat{\beta}[\lambda] \) itself. However, we note that these performance measures only make sense if the underlying \( \beta \) is identifiable. To that end, there is a large volume of work showing the Lasso is model selection consistent under fairly general conditions [51, 33, 45, 48]. In particular, it assures us the asymptotic recovery of the underlying true \( \beta \).

However, in the finite sample case, and especially when the features are highly correlated, different linear combinations of features (of a given sparsity) may give approximately equivalent fits. Under data perturbation, it is not surprising that the different solution paths chooses different features. This makes any metric based on \( \hat{\beta}[\lambda] \) statistically unstable since \( V \) is small. Note that this does not contradict the assessment of the eventual \( \hat{\beta}[\lambda] \) picked since SSCV and CV, picking from the same solution path, would both suffer from any failure of the original Lasso.

#### 3.4.2 Data Perturbation Schemes

In this work, we have used cross-validation folds to compute our pseudo-solutions. There are of course many other ways to generate pseudo datasets. One related approach would be to apply bootstrap sampling [4]. Here, simply sample with replacement from the original data set to generate multiple data sets. These two approaches are obvious choices, and can be applied to any estimation procedure (even those without an optimization formulation). A third choice, which applies only to penalized \( M \)-estimators such as the Lasso, is based on perturbations of the penalty [34]. Note that such perturbations of the penalty amount to perturbing (indirectly) the samples, but in a different way than bootstrapping. Finally, we can simply perturb the data directly by adding noise to \( X \) and/or \( Y \). For example, we can add random Gaussian noise to the response [9].

We find that the choice of data perturbation scheme does not affect the results much.
3.4.3 Computation Cost

With high dimensional data, computation can be costly. In the case of the Lasso, each implementation requires $O(\min(np^2, n^3))$ operations [12], which quickly gets expensive with larger data sets. Using the statistical stability metric to select the regularization parameter incurs only as much computation as using cross validation. This is because the bulk of the computation in both cases rests in computing the solution paths of the $V$ perturbed data sets. $V$ in this case can be small as demonstrated in Section 3.3. This is in contrast to related work [4, 34] which requires a much larger $V$.

3.5 Conclusion

Regularization methods are employed to deal with problems in the increasingly common high dimensional setting. However, the difficult problem of selecting the associated regularization parameter $\lambda$ is not well studied. Our method $SSCV$ is based on statistical stability but also takes into account model fit. We have demonstrated that $SSCV$ is a viable alternative to the popular cross validation $CV$ for choosing $\lambda$ for the Lasso. This is especially the case when prediction is not the primary objective. We have shown through various simulations and on two real data sets that $SSCV$, at the same computation cost, can often outperform $CV$. We believe this result is not restricted to the Lasso but holds for other regularization methods as well.

We also believe that this method can also be readily extended to the classification problem through the generalized linear model, and leave this to future work.
### Table 3.5: Performance of $SSCV$, $CV$ and $BIC$ in picking the regularization parameter for the Lasso for $p = 50$.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\sigma$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.357</td>
<td>0.303</td>
<td>0.327</td>
<td>0.357</td>
<td>0.303</td>
<td>0.327</td>
<td>0.776</td>
<td>0.556</td>
<td>0.740</td>
<td>15.8</td>
<td>26.0</td>
<td>17.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.665</td>
<td>0.597</td>
<td>0.654</td>
<td>0.665</td>
<td>0.597</td>
<td>0.654</td>
<td>0.712</td>
<td>0.559</td>
<td>0.746</td>
<td>17.6</td>
<td>25.7</td>
<td>16.4</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1.23</td>
<td>1.17</td>
<td>1.39</td>
<td>1.23</td>
<td>1.17</td>
<td>1.39</td>
<td>0.586</td>
<td>0.549</td>
<td>0.664</td>
<td>18.8</td>
<td>23.4</td>
<td>9.96</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td>0.294</td>
<td>0.296</td>
<td>0.291</td>
<td>0.294</td>
<td>0.276</td>
<td>0.284</td>
<td>0.702</td>
<td>0.603</td>
<td>0.706</td>
<td>18.5</td>
<td>23.2</td>
<td>18.3</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>0.581</td>
<td>0.589</td>
<td>0.584</td>
<td>0.566</td>
<td>0.548</td>
<td>0.564</td>
<td>0.687</td>
<td>0.606</td>
<td>0.701</td>
<td>19.0</td>
<td>22.9</td>
<td>18.4</td>
</tr>
<tr>
<td>0.2</td>
<td>2</td>
<td>1.10</td>
<td>1.17</td>
<td>1.12</td>
<td>1.07</td>
<td>1.06</td>
<td>1.12</td>
<td>0.647</td>
<td>0.593</td>
<td>0.682</td>
<td>18.6</td>
<td>21.8</td>
<td>16.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.352</td>
<td>0.364</td>
<td>0.355</td>
<td>0.266</td>
<td>0.268</td>
<td>0.268</td>
<td>0.660</td>
<td>0.614</td>
<td>0.681</td>
<td>20.3</td>
<td>22.6</td>
<td>19.4</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.704</td>
<td>0.727</td>
<td>0.712</td>
<td>0.531</td>
<td>0.534</td>
<td>0.538</td>
<td>0.658</td>
<td>0.613</td>
<td>0.679</td>
<td>20.1</td>
<td>22.4</td>
<td>19.1</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>1.31</td>
<td>1.36</td>
<td>1.33</td>
<td>1.00</td>
<td>1.01</td>
<td>1.04</td>
<td>0.613</td>
<td>0.580</td>
<td>0.628</td>
<td>18.7</td>
<td>20.7</td>
<td>17.2</td>
</tr>
<tr>
<td>0.9</td>
<td>0.5</td>
<td>0.784</td>
<td>0.799</td>
<td>0.784</td>
<td>0.275</td>
<td>0.262</td>
<td>0.262</td>
<td>0.632</td>
<td>0.614</td>
<td>0.659</td>
<td>21.0</td>
<td>22.1</td>
<td>19.8</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>1.41</td>
<td>1.47</td>
<td>1.42</td>
<td>0.473</td>
<td>0.485</td>
<td>0.485</td>
<td>0.597</td>
<td>0.572</td>
<td>0.612</td>
<td>18.9</td>
<td>20.4</td>
<td>17.7</td>
</tr>
<tr>
<td>0.9</td>
<td>2</td>
<td>2.21</td>
<td>2.31</td>
<td>2.22</td>
<td>0.793</td>
<td>0.775</td>
<td>0.803</td>
<td>0.461</td>
<td>0.449</td>
<td>0.455</td>
<td>14.4</td>
<td>15.9</td>
<td>13.0</td>
</tr>
</tbody>
</table>

### Table 3.6: Performance of $SSCV$, $CV$ and $BIC$ in picking the regularization parameter for the Lasso for $p = 500$.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\sigma$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
<th>$SSCV$</th>
<th>$CV$</th>
<th>$BIC$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.626</td>
<td>0.531</td>
<td>0.593</td>
<td>0.626</td>
<td>0.531</td>
<td>0.593</td>
<td>0.546</td>
<td>0.309</td>
<td>0.204</td>
<td>26.0</td>
<td>54.7</td>
<td>88.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1.17</td>
<td>1.04</td>
<td>1.17</td>
<td>1.17</td>
<td>1.04</td>
<td>1.17</td>
<td>0.458</td>
<td>0.317</td>
<td>0.197</td>
<td>27.3</td>
<td>49.7</td>
<td>87.7</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1.80</td>
<td>1.76</td>
<td>2.08</td>
<td>1.80</td>
<td>1.76</td>
<td>2.08</td>
<td>0.300</td>
<td>0.291</td>
<td>0.159</td>
<td>27.1</td>
<td>33.1</td>
<td>78.9</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td>0.511</td>
<td>0.513</td>
<td>0.624</td>
<td>0.479</td>
<td>0.469</td>
<td>0.563</td>
<td>0.408</td>
<td>0.342</td>
<td>0.203</td>
<td>39.0</td>
<td>48.5</td>
<td>88.5</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>1.01</td>
<td>1.02</td>
<td>1.24</td>
<td>0.945</td>
<td>0.931</td>
<td>1.12</td>
<td>0.392</td>
<td>0.332</td>
<td>0.195</td>
<td>38.8</td>
<td>47.9</td>
<td>88.6</td>
</tr>
<tr>
<td>0.2</td>
<td>2</td>
<td>1.67</td>
<td>1.71</td>
<td>2.19</td>
<td>1.58</td>
<td>1.58</td>
<td>1.99</td>
<td>0.326</td>
<td>0.287</td>
<td>0.154</td>
<td>33.6</td>
<td>40.9</td>
<td>87.8</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.626</td>
<td>0.636</td>
<td>0.783</td>
<td>0.454</td>
<td>0.458</td>
<td>0.559</td>
<td>0.376</td>
<td>0.347</td>
<td>0.203</td>
<td>43.1</td>
<td>47.5</td>
<td>88.4</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>1.21</td>
<td>1.23</td>
<td>1.52</td>
<td>0.888</td>
<td>0.888</td>
<td>1.09</td>
<td>0.356</td>
<td>0.325</td>
<td>0.184</td>
<td>41.1</td>
<td>46.3</td>
<td>88.9</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>1.89</td>
<td>1.93</td>
<td>2.62</td>
<td>1.41</td>
<td>1.41</td>
<td>1.87</td>
<td>0.281</td>
<td>0.256</td>
<td>0.133</td>
<td>33.1</td>
<td>38.5</td>
<td>88.0</td>
</tr>
<tr>
<td>0.9</td>
<td>0.5</td>
<td>1.32</td>
<td>1.35</td>
<td>1.67</td>
<td>0.430</td>
<td>0.433</td>
<td>0.533</td>
<td>0.335</td>
<td>0.315</td>
<td>0.178</td>
<td>43.0</td>
<td>46.3</td>
<td>89.0</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>2.02</td>
<td>2.06</td>
<td>2.87</td>
<td>0.667</td>
<td>0.670</td>
<td>0.917</td>
<td>0.251</td>
<td>0.235</td>
<td>0.122</td>
<td>34.4</td>
<td>38.0</td>
<td>88.4</td>
</tr>
<tr>
<td>0.9</td>
<td>2</td>
<td>2.58</td>
<td>2.72</td>
<td>4.85</td>
<td>0.891</td>
<td>0.903</td>
<td>1.56</td>
<td>0.135</td>
<td>0.125</td>
<td>0.0689</td>
<td>22.6</td>
<td>26.9</td>
<td>85.8</td>
</tr>
</tbody>
</table>
### Table 3.7: Performance of \( \text{SSCV} \), \( \text{CV} \) and \( \text{BIC} \) in picking the regularization parameter for the Lasso for the block correlation design.

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>( \sigma )</th>
<th>Estimation error</th>
<th>Prediction error</th>
<th>Model Selection F-measure</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.5</td>
<td>0.437 ( \text{SSCV} )</td>
<td>0.406 ( \text{CV} )</td>
<td>0.378 ( \text{BIC} )</td>
<td>0.636 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.3</td>
<td>1</td>
<td>0.845 ( \text{SSCV} )</td>
<td>0.798 ( \text{CV} )</td>
<td>0.747 ( \text{BIC} )</td>
<td>0.594 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.3</td>
<td>2</td>
<td>1.48</td>
<td>1.47</td>
<td>1.39</td>
<td>0.465</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.457 ( \text{SSCV} )</td>
<td>0.445 ( \text{CV} )</td>
<td>0.363 ( \text{BIC} )</td>
<td>0.586 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.908 ( \text{SSCV} )</td>
<td>0.899 ( \text{CV} )</td>
<td>0.735 ( \text{BIC} )</td>
<td>0.544 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>1.56</td>
<td>1.57</td>
<td>1.32</td>
<td>0.454 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.9</td>
<td>0.5</td>
<td>0.876</td>
<td>0.904</td>
<td>0.345</td>
<td>0.501 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>1.54</td>
<td>1.59</td>
<td>0.629</td>
<td>0.444 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.9</td>
<td>2</td>
<td>2.17</td>
<td>2.33</td>
<td>1.03</td>
<td>0.310 ( \text{SSCV} )</td>
</tr>
</tbody>
</table>

### Table 3.8: Performance of \( \text{SSCV} \), \( \text{CV} \) and \( \text{BIC} \) in picking the regularization parameter for the Lasso for the Toeplitz correlation design.

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>( \sigma )</th>
<th>Estimation error</th>
<th>Prediction error</th>
<th>Model Selection F-measure</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.5</td>
<td>0.460 ( \text{SSCV} )</td>
<td>0.411 ( \text{CV} )</td>
<td>0.383 ( \text{BIC} )</td>
<td>0.649 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>0.877 ( \text{SSCV} )</td>
<td>0.813 ( \text{CV} )</td>
<td>0.761 ( \text{BIC} )</td>
<td>0.578 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.9</td>
<td>2</td>
<td>1.53</td>
<td>1.48</td>
<td>1.40</td>
<td>0.444 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.99</td>
<td>0.5</td>
<td>0.696 ( \text{SSCV} )</td>
<td>0.706 ( \text{CV} )</td>
<td>0.625 ( \text{BIC} )</td>
<td>0.532 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.99</td>
<td>1</td>
<td>1.28</td>
<td>1.31</td>
<td>0.663</td>
<td>0.488 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.99</td>
<td>2</td>
<td>1.93</td>
<td>1.99</td>
<td>1.14</td>
<td>0.372 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.999</td>
<td>0.5</td>
<td>1.72</td>
<td>1.76</td>
<td>0.572</td>
<td>0.400 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.999</td>
<td>1</td>
<td>2.31</td>
<td>2.39</td>
<td>0.954</td>
<td>0.290 ( \text{SSCV} )</td>
</tr>
<tr>
<td>0.999</td>
<td>2</td>
<td>2.81</td>
<td>2.96</td>
<td>1.45</td>
<td>0.188 ( \text{SSCV} )</td>
</tr>
</tbody>
</table>
Chapter 4

Robustness in High Dimensions

4.1 Introduction

In the high dimensional data setting, there is high instability across different samples under standard regression models: estimators vary wildly depending on the sample observed. We have seen from Chapter 3 how focusing on stability properties can help in this regime. In this chapter, we look at some classical robust statistics results and put them in the high dimensional data setting. We are interested in these robust procedures because they yield stable estimators when faced with outliers in the classical setting. This results in better prediction performance compared to ordinary least squares when the error in the regression model has a heavy-tailed distribution. We will investigate whether this robustness against error distributions and improvement to prediction performance extend to the high dimensional setting.

4.1.1 Model

Consider the usual linear model. Let $X \in \mathbb{R}^p, Y \in \mathbb{R}$ be random variables, where

$$Y = X'\beta + \epsilon,$$

$\beta \in \mathbb{R}^p$ is the model parameter, and $\epsilon \in \mathbb{R}$ is unobservable and independent of $X$, with $E(\epsilon) = 0$ and $E(\epsilon^2) = \sigma^2$. We observe data $(X_i, Y_i)$ for $i = 1, \ldots, n$, independent and identically distributed copies of $(X, Y)$, and wish to minimize the mean square prediction error (MSPE) for future observations,

$$MSPE(\hat{\beta}) = E_{X,Y}(Y - X'\hat{\beta})^2.$$
4.1.2 Classical Setting

The simplest and most common approach is ordinary least squares (OLS), which minimizes the empirical mean square error, also known as the $L_2$ loss,

$$\hat{\beta}_{OLS} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X_i'\beta)^2.$$ 

When $\epsilon$ is normally distributed, $\hat{\beta}_{OLS}$ is the maximum likelihood estimator (MLE). It is an efficient estimator: an unbiased estimator with variance achieving the Cramér-Rao lower bound (CRLB). In particular, it has the lowest MSPE among all unbiased estimators.

More generally, for a large class of distributions of $\epsilon$, the corresponding MLE is asymptotically efficient. That is, it is an efficient estimator under classical asymptotics of letting $n \to \infty$ while $p$ is fixed.

4.1.3 Robustness

Robustness has been studied extensively under the classical setting [19, 3, 20, 2, 21, 50, 5, 6, 24]. It has various definitions. Here, we refer to distributional robustness of the estimator. We are particularly concerned with the impact on the MSPE when $\epsilon$ deviates from its assumed distribution.

The OLS while efficient when $\epsilon$ is normal, is not robust. The $L_2$ loss is sensitive to outliers: a single extreme point would change the estimator drastically. Consequently, the OLS works poorly for heavy tail distributions. One popular method that address this flaw is using least absolute deviations (LAD) instead. It minimizes the $L_1$ loss instead of the $L_2$ loss.

$$\hat{\beta}_{LAD} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} |Y_i - X_i'\beta|$$

It is easy to see why $L_1$ loss is more robust against heavier tails in classical settings. The relative contribution of an outlier to the $L_1$ loss function is much lower than to the $L_2$ loss function. Indeed, the LAD estimator corresponds to the MLE when $\epsilon$ has the double exponential distribution, which has a heavier tail than the normal distribution. In that case, the LAD estimator is twice as efficient as the OLS estimator. Correspondingly, it is not surprising that LAD continues to outperform OLS for $\epsilon$ distributed with heavier tails when $p$ is fixed.

4.1.4 High Dimensions

There has been various works studying efficiency results in the high dimensional setting [38, 32]. In this chapter, we are not too concerned with finding efficient estimators. We focus instead on the above result that the $L_1$ loss function is more robust than the $L_2$ loss.
function to noise distributions with heavy tails. This is well-established in the classical regime where \( n \to \infty \) with \( p \) fixed. Our aim is to investigate if this continues to hold in high dimensional settings where \( p \) is allowed to grow with \( n \), and if so, to what extent.

We will look at two regimes. The first is the \( p > n \) case. This corresponds to the practical problem of dealing with modern data where there are more variables than samples. Since minimizing either loss function is an underdetermined problem, we necessarily have to include a variable selection step. The second is the \( p < n \) case, where \( p/n \) tends to a constant. There, we can compare more directly the robustness properties of the estimators with the classical setting results.

### 4.2 \( p > n \) case

We will rely on a series of simulations using sparse high dimensional true models to investigate the behavior of the estimators based on \( L_1 \) and \( L_2 \) losses. Our goal is to verify results we expect like better relative performance of using the \( L_1 \) loss instead of \( L_2 \) loss under error distributions with heavy tails, as well as to discover and shed light into cases in the high dimensional regime where there are fundamentally differences from the classical regime.

#### 4.2.1 Exploring Tail Distributions

We start by looking at a typical high dimensional setup. The data \((X_i, Y_i)\) for \( i = 1, \ldots, n \) follows our linear model with \( n = 100 \) and \( p = 200 \). Here, \( \beta = (1, \ldots, 1, 0, \ldots, 0)' \) where there are \( s = 10 \) entries of 1 and \( p - s \) entries of 0. \( s \) denotes the underlying true dimension of the model. That is, even though the ambient dimension of the data \( p \) is high, it only lies in an \( s \) dimensional subspace. We let \( X_i \in \mathbb{R}^p \) be normally distributed with \( Cov(X_i) = I_p \), the \( p \)-dimensional identity matrix.

We make no claims that the simple design we choose here is representative of all high dimensional problems. The rationale here is to keep things plain so that we do not conflate the effects of the tail distributions we want to investigate with other unexpected effects. This is simply intended as an exploratory stage to give us results to compare with what we would expect in the classical regime.

We vary the distribution of \( \epsilon \) here by looking at some familiar distributions with increasingly heavy tails: Gaussian, Logistic, Double Exponential and Cauchy. We will vary the scale for each of the distributions to reflect the entire range of reasonable estimators.

In the classical regime, where \( n \to \infty \) with \( p \) fixed, we know that minimizing the \( L_1 \) loss gives better results for heavier tails than minimizing the \( L_2 \) loss. Here, we cannot simply minimize the loss function as \( p > n \) leads to 0 loss in both cases. Instead, we introduce an \( L_1 \) penalty term for its straightforward dimension reduction properties, leading to the estimators
\hat{\beta}_{L_2L_1}(\lambda) = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X_i'\beta)^2 + \lambda ||\beta||_1

and

\hat{\beta}_{L_1L_1}(\lambda) = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} |Y_i - X_i'\beta| + \lambda ||\beta||_1.

\hat{\beta}_{L_2L_1} is of course the familiar Lasso [44] estimator. \hat{\beta}_{L_1L_1} is also known as the LAD-Lasso estimator. \lambda \in \mathbb{R} in either case is the regularization parameter. We have discussed the difficulties and intricacies of choosing the regularization parameter in Chapter 3. To not convolute the problems, for now, we will sidestep this issue and use an independent and identically distributed (i.i.d.) test set \((X_i^*, Y_i^*)\) for \(i = 1, \ldots, n\) to select \(\lambda\) based on the empirical MSPE. We do this since we care about prediction error here, and not other performance measures like model selection consistency. For example,

\[ \hat{\beta}_{L_2L_1} := \hat{\beta}_{L_2L_1}(\hat{\lambda}), \]

where

\[ \hat{\lambda} = \arg\min_{\lambda} \frac{1}{n} \sum_{i=1}^{n} (Y_i^* - X_i'^*\hat{\beta}_{L_2L_1}(\lambda))^2. \]

Recall our goal is to minimize the MSPE for future observations, where

\[ MSPE(\hat{\beta}) = E_{X,Y}(Y - X'\hat{\beta}) = \sigma^2 + (\hat{\beta} - \beta)'Cov(X)(\hat{\beta} - \beta). \]

Since the \(\sigma^2\) is common to the estimators, we will only consider the second term. This also provides us with a method of comparing the estimators on Cauchy noise since \(\sigma^2\) is undefined in that case. We repeat each simulation 100 times and take the median value, again because the mean is divergent for the Cauchy case.

Table 4.1 shows the median MSPE of \(\hat{\beta}_{L_2L_1}\) and \(\hat{\beta}_{L_1L_1}\) for the 100 runs. The scale values correspond to the natural scale parameters for each of the distributions (normalized by \(\sqrt{3}\)). They were chosen to compare the two estimators over the full range of reasonable values. (Note that \(\hat{\beta} = 0\) has MSPE 10.)

On the whole, the results showed what was expected: the heavier the tails, the better \(\hat{\beta}_{L_1L_1}\) performed relative to \(\hat{\beta}_{L_2L_1}\). Note that the four error distributions, Gaussian, Logistic, Double Exponential and Cauchy, are in order of tails with increasing weight: the first three of them has excess kurtosis (standardized fourth moment) of 0, 1.2, and 3 respectively, while Cauchy’s fourth moment is divergent. We see that the heavier the tail of the error distribution, the better \(\hat{\beta}_{L_1L_1}\) performs relative to \(\hat{\beta}_{L_2L_1}\). The performance gain from the robustness property of the \(L_1\) loss function extends to the high dimensional setting.
4.2.2 Bias-Variance Breakdown for the Double Exponential Case

There are two main disconnects between the classical case and our results. First, we are nowhere near the classical asymptotics. Second, we have an extra $L_1$ penalty term around. Both of these relates to the Bias-Variance breakdown of the MSPE.

In the classical case, we have unbiased estimators, and the MSPE is purely variance. The efficiency result is a statement about minimal variance in the class of unbiased estimators. It is reasonable to conjecture that it is only this minimal variance property that is preserved in the high dimensional setting rather than the whole minimal MSPE. This fits in with the fact that the bias term is due to the presence of the $L_1$ penalty. However, it turns out surprisingly that coupled with the $L_1$ penalty, the $L_1$ estimator has higher variance than the $L_2$ estimator in the high dimensional setting.

As before, we will only consider the second term of the MSPE expansion

$$ (\hat{\beta} - \beta)' \text{Cov}(X) (\hat{\beta} - \beta), $$

Table 4.1: Median MSPE of $\hat{\beta}_{L_2 L_1}$ and $\hat{\beta}_{L_1 L_1}$ for the basic setup: $n = 100$, $p = 200$, $s = 10$ and non-zero $\beta_j = 1$. The four error distributions, Gaussian, Logistic, Double Exponential and Cauchy, are in order of tails with increasing weight. Numbers are based on 100 runs.

<table>
<thead>
<tr>
<th>Scale</th>
<th>$L_2 L_1$</th>
<th>$L_1 L_1$</th>
<th>$L_2 L_1$</th>
<th>$L_1 L_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gaussian Error</td>
<td>Logistic Error</td>
<td>Double Exp. Error</td>
<td>Cauchy Error</td>
</tr>
<tr>
<td>0.05</td>
<td>0.0163</td>
<td>0.0220</td>
<td>0.0546</td>
<td>0.0688</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0647</td>
<td>0.0961</td>
<td>0.205</td>
<td>0.261</td>
</tr>
<tr>
<td>0.5</td>
<td>1.78</td>
<td>2.37</td>
<td>5.06</td>
<td>5.86</td>
</tr>
<tr>
<td>1</td>
<td>5.95</td>
<td>7.09</td>
<td>9.43</td>
<td>9.74</td>
</tr>
<tr>
<td>2</td>
<td>9.73</td>
<td>10.1</td>
<td>10.30</td>
<td>10.32</td>
</tr>
</tbody>
</table>
whose expectation can be written as the sum of the expectations of

$$\text{Bias}^2(\hat{\beta}) = (E(\hat{\beta}) - \beta)'Cov(X)(E(\hat{\beta}) - \beta)$$

and

$$\text{Var}(\hat{\beta}) = (\hat{\beta} - E(\hat{\beta}))'Cov(X)(\hat{\beta} - E(\hat{\beta})).$$

Again, taking expectations of $\hat{\beta}_{L_1 L_1}$ and $\hat{\beta}_{L_2 L_1}$ is difficult, so we estimate them by simulation. Let $\hat{\beta}^{(k)}$ be the estimator for the $k$th simulation, and let $\bar{\hat{\beta}} = \frac{1}{K} \sum_{k=1}^{K} \hat{\beta}^{(k)}$. Then the mean MSPE across $K$ simulations

$$\frac{1}{K} \sum_{k=1}^{K} (\hat{\beta}^{(k)} - \beta)'Cov(X)(\hat{\beta}^{(k)} - \beta),$$

can be written as the sum of

$$\widehat{\text{Bias}}^2 = (\bar{\hat{\beta}} - \beta)'Cov(X)(\bar{\hat{\beta}} - \beta)$$

and

$$\widehat{\text{Var}} = \frac{1}{K} \sum_{k=1}^{K} (\hat{\beta}^{(k)} - \bar{\hat{\beta}})'Cov(X)(\hat{\beta}^{(k)} - \bar{\hat{\beta}}).$$

In Table 4.2, we see the more detailed results for another run of $K = 100$ simulations for the double exponential case. Note that the estimated standard errors for the mean MSPE is simply the variance divided by $\sqrt{K} = 10$.

<table>
<thead>
<tr>
<th>Scale</th>
<th>$L_2 L_1$</th>
<th>$L_1 L_1$</th>
<th>$L_2 L_1$</th>
<th>$L_1 L_1$</th>
<th>Mean MSPE</th>
<th>Mean $L_1$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.0177</td>
<td><strong>0.0148</strong></td>
<td>0.0181</td>
<td>0.0233</td>
<td><strong>0.0358</strong></td>
<td>9.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0601</td>
<td><strong>0.0479</strong></td>
<td>0.0738</td>
<td>0.0936</td>
<td><strong>0.134</strong></td>
<td>10.1</td>
</tr>
<tr>
<td>0.5</td>
<td>1.69</td>
<td><strong>1.42</strong></td>
<td>1.78</td>
<td>2.03</td>
<td><strong>3.48</strong></td>
<td>9.87</td>
</tr>
<tr>
<td>1</td>
<td>5.96</td>
<td><strong>5.36</strong></td>
<td>2.21</td>
<td>2.96</td>
<td><strong>8.17</strong></td>
<td>5.44</td>
</tr>
<tr>
<td>2</td>
<td>8.54</td>
<td><strong>7.63</strong></td>
<td><strong>1.72</strong></td>
<td>2.70</td>
<td><strong>10.3</strong></td>
<td>2.68</td>
</tr>
</tbody>
</table>

Table 4.2: Bias-Variance breakdown of the mean MSPE for the double exponential case.

We see that the variance numbers for $L_1 L_1$ is higher, not lower across the board. The $L_1 L_1$ estimator actually incurs more variance whereas the $L_2 L_1$ estimator incurs more bias. This is true even in the lower noise cases, where the $L_1$-norm of the estimators are comparable. Indeed this phenomenon is present across a whole slew of other model setups. We vary $p$, the $\beta$ structure, the correlation between predictors, and continue to find that $L_1 L_1$ estimator is
often more variable than $L_2 L_1$ for the double exponential case. The corresponding numbers can be found at the end of the chapter in Tables 4.5 to 4.8.

The interaction of the loss function with the $L_1$ penalty has changed the nature of the estimator, with the traditionally more robust $L_1$ loss estimator being more variable than the $L_2$ loss estimator, at least for the case of the double exponential tails. We will leave the investigation of the effect of the tail distribution on this interaction for future work.

### 4.2.3 Decoupling Selection and Fitting

The $L_1$ penalty gave us an unexpected twist in our estimators. While analysis of its effect is an interesting tangent, our primary objective in this chapter is to investigate the $L_1$ and $L_2$ loss functions in the high dimensional setting. Here, we attempt to divorce the effects of the $L_1$ penalty on our estimators by decoupling the selection and fitting steps.

The $L_1$ penalty was chosen for its sparsity-inducing property. In particular, the resulting estimator is sparse, with many zero entries. We can use this initial estimator as a selection mechanism, and refit the data to only this subset of variables. More precisely, let $\hat{\beta}$ be the initial sparse estimator, e.g. $\hat{\beta}_{L_1 L_1}, \hat{\beta}_{L_2 L_1}$, then

$$\tilde{\beta}_{\text{OLS}} = \arg \min_{\beta \in \mathbb{R}^p; \beta_j = 0, \forall j \notin M} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \sum_{j \in M} X_{ij} \beta_j)^2$$

and

$$\tilde{\beta}_{\text{LAD}} = \arg \min_{\beta \in \mathbb{R}^p; \beta_j = 0, \forall j \notin M} \frac{1}{n} \sum_{i=1}^{n} |Y_i - \sum_{j \in M} X_{ij} \beta_j|,$$

where

$$M = \{ j : \hat{\beta}_j \neq 0 \}.$$

We apply the 4 possible two-step procedures to our double exponential case and compare the mean MSPE in Table 4.3. Neither selection process dominates the other. Neither loss function fitting dominates the other either. OLS worked better when the variables were selected by $L_2 L_1$ and LAD worked better when the variables were selected by $L_1 L_1$. In this case, we did not manage to decouple the selection from the fitting. However, note that given a selection $M$ with $|M| < n$, we are reduced to the $p < n$ case. We can split the overall problem into characterizing the selection properties of $L_2 L_1$ and $L_1 L_1$, e.g. model selection consistency [52, 40], and understanding the conditions when OLS and LAD works best. We will turn to the latter problem in Section 4.3.

### 4.3 $p \leq n$ case, $p/n \to \rho$

In this section, we consider the scenario where $p \leq n$, but unlike the classical asymptotics, we allow $p$ to grow with $n$ so that $p/n \to \rho$ for some $\rho \in (0, 1)$. This allows us to compare
Table 4.3: MSPE performance of two-step regression for the double exponential case. Penalized regression is used to select the variables and regular regression is used to fit. Numbers are based on 1000 runs.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Mean MSPE</th>
<th>SE(Mean MSPE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_2 L_1$</td>
<td>$L_1 L_1$</td>
</tr>
<tr>
<td></td>
<td>OLS</td>
<td>LAD</td>
</tr>
<tr>
<td>0.05</td>
<td>0.0579</td>
<td>0.0613</td>
</tr>
<tr>
<td>0.1</td>
<td>0.229</td>
<td>0.242</td>
</tr>
<tr>
<td>0.5</td>
<td>5.56</td>
<td>5.90</td>
</tr>
<tr>
<td>1</td>
<td>14.9</td>
<td>15.2</td>
</tr>
<tr>
<td>2</td>
<td>25.8</td>
<td>23.9</td>
</tr>
</tbody>
</table>

directly the OLS and LAD estimators without tagging on the extra penalty term.

### 4.3.1 Double Exponential Tails

We recall from last section that when $\epsilon$ is double exponential distributed, $L_2 L_1$ and $L_1 L_1$ had comparable performance. Here, we mirror the same setup, but let $p = 10, 20, \ldots, 50$. With $p < n$, we use OLS and LAD, optimizing the $L_2$ and $L_1$ loss respectively with no extra penalty term. Table 4.4 summarizes the results. Figure 4.1 plots out the MSPE numbers against the scale of $\epsilon$ and $p$ respectively.

We first note that both OLS and LAD estimators are shift invariant, so the numbers are representative of all possible $\beta$ in the underlying model. This also implies that the MSPE scales with $E(\epsilon^2) = \sigma^2$, as verified by the parallel lines in the Figure 4.1.

The more interesting observation is that LAD outperforms OLS at $p = 10$ as predicted by classical asymptotics, but loses its edge gradually as we increase $p$. For $p = 30$ onwards, OLS actually does better. The efficiency result of $\hat{\beta}_{LAD}$, the MLE for the double exponential tails, is upended when $p$ is large enough relative to $n$.

We extend the simulations to include $p = 60, 70, 80, 90$, and also the Gaussian tails case for reference. We plot the ratio $\frac{MSPE(\hat{\beta}_{LAD})}{MSPE(\hat{\beta}_{OLS})}$ in Figure 4.2. We see that OLS continues to do better than LAD as we further increase $p$ for the double exponential case. We also see that for the Gaussian tails case, OLS dominates LAD for all values of $p$.

Comparing the two sets of ratios, we see that the relative performance of LAD and OLS, the respective MLE’s for double exponential and Gaussian tails, start out very differently when $p$ is small, but get closer as we increase $p$. This suggests that as we move further away from the classical asymptotics by increasing $p$, the difference due to the error tail distribution diminishes.
<table>
<thead>
<tr>
<th>$p$</th>
<th>Scale</th>
<th>MSPE</th>
<th>SE(MSPE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OLS</td>
<td>LAD</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0224</td>
<td><strong>0.0185</strong></td>
<td>0.00037</td>
</tr>
<tr>
<td>0.5</td>
<td>0.582</td>
<td><strong>0.473</strong></td>
<td>0.0095</td>
</tr>
<tr>
<td>1</td>
<td>2.25</td>
<td><strong>1.85</strong></td>
<td>0.037</td>
</tr>
<tr>
<td>2</td>
<td>8.81</td>
<td><strong>7.46</strong></td>
<td>0.15</td>
</tr>
<tr>
<td>4</td>
<td>36.7</td>
<td><strong>30.4</strong></td>
<td>0.60</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0533</td>
<td><strong>0.0488</strong></td>
<td>0.00069</td>
</tr>
<tr>
<td>0.5</td>
<td>1.28</td>
<td><strong>1.17</strong></td>
<td>0.016</td>
</tr>
<tr>
<td>1</td>
<td>5.11</td>
<td><strong>4.84</strong></td>
<td>0.066</td>
</tr>
<tr>
<td>2</td>
<td>20.1</td>
<td><strong>19.0</strong></td>
<td>0.24</td>
</tr>
<tr>
<td>4</td>
<td>83.9</td>
<td><strong>76.8</strong></td>
<td>1.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0902</td>
<td>0.0947</td>
<td>0.0010</td>
</tr>
<tr>
<td>0.5</td>
<td><strong>2.22</strong></td>
<td>2.27</td>
<td>0.024</td>
</tr>
<tr>
<td>1</td>
<td><strong>8.68</strong></td>
<td>8.85</td>
<td>0.10</td>
</tr>
<tr>
<td>2</td>
<td><strong>35.3</strong></td>
<td>36.6</td>
<td>0.41</td>
</tr>
<tr>
<td>4</td>
<td><strong>139</strong></td>
<td>143</td>
<td>1.7</td>
</tr>
<tr>
<td>0.1</td>
<td><strong>0.136</strong></td>
<td>0.15</td>
<td>0.0015</td>
</tr>
<tr>
<td>0.5</td>
<td><strong>3.46</strong></td>
<td>3.84</td>
<td>0.038</td>
</tr>
<tr>
<td>1</td>
<td><strong>13.9</strong></td>
<td>15.4</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td><strong>55.6</strong></td>
<td>61.2</td>
<td>0.57</td>
</tr>
<tr>
<td>4</td>
<td><strong>225</strong></td>
<td>244</td>
<td>2.4</td>
</tr>
<tr>
<td>0.1</td>
<td><strong>0.21</strong></td>
<td>0.244</td>
<td>0.0022</td>
</tr>
<tr>
<td>0.5</td>
<td><strong>5.26</strong></td>
<td>6.12</td>
<td>0.057</td>
</tr>
<tr>
<td>1</td>
<td><strong>20.7</strong></td>
<td>24.7</td>
<td>0.23</td>
</tr>
<tr>
<td>2</td>
<td><strong>84.8</strong></td>
<td>99.1</td>
<td>0.96</td>
</tr>
<tr>
<td>4</td>
<td><strong>330</strong></td>
<td>394</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 4.4: MSPE numbers for OLS and LAD for double exponential tails. Numbers are based on 1000 runs.
Figure 4.1: Plot of MSPE in Table 4.4. Circles and solid lines represent OLS values, triangles and dashed lines represent LAD values. The different colors represent the different scale factors for $\epsilon$. Note $n$ is fixed at 100.
Figure 4.2: The ratio of MSPE for LAD to MSPE for OLS. The black circles represent the double exponential tails case, and the red pluses represent the Gaussian tails case. Each panel corresponds to the same $p$. 
4.3.2 Family of $t$ Tails

We saw that in the double exponential case, OLS catches up with LAD as we increase $p$, overtaking it around $p = 30$. Here, we investigate the effect of the tail distribution on this threshold. In particular, we investigate the case where $\epsilon$ is $t$ distributed. We choose the $t$ distribution family, indexed by the degrees of freedom $\nu$, as a way to increase the weight of the tails smoothly. We note that the $t$ family spans the Cauchy distribution ($\nu = 1$) to the Gaussian distribution ($\nu = \infty$).

As before, we will keep $n = 100$ and vary $p$. Figure 4.3 shows the relative MSPE of LAD and OLS for $t$ tails with $\nu = 3, 3.4, 3.8, \ldots, 5$. We see that heavier tails prefer LAD. For example, for $p = 10$, LAD outperforms OLS for $\nu = 3$ but gradually loses its edge as we increase $\nu$. Even though LAD does worse than OLS for larger $p$’s, the ratio of MSPE’s is still lower for heavier tails. That is, the heavier the tails, the more advantageous (or in some cases less disadvantageous), it is to use LAD instead of OLS.

For all the cases here, OLS eventually catches up with LAD for big enough $p$. The heavier the tails, the larger $p$ has to be for OLS to catch up. This is saying the robustness to heavy tail distributions property of LAD eventually fail as we move to higher dimensions. We conjecture that this is true for all heavy tail distributions. See [22] for theoretical work spurred by these simulations.

4.3.3 Asymptotics: $p/n \to \rho$

The simulations so far only dealt with $n = 100$. For the $p \leq n$ case, we conjecture that the results presented here hold for general $n$. In particular, the results hold true when $p/n \to \rho$ as $n \to \infty$, for some constant $\rho \in (0, 1)$.

Consider the OLS estimator in the Gaussian design case ($X_i \sim N(0, \Sigma)$). Let $X = (X_1, X_2, \ldots, X_n)' \in \mathbb{R}^{n \times p}$ be the design matrix for our observations, $Y = (Y_1, Y_2, \ldots, Y_n)' \in \mathbb{R}^n$ be the vector of responses, and $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$. Then our OLS estimator is

$$\hat{\beta}_{OLS} = (X'X)^{-1}X'Y = \beta + (X'X)^{-1}X'\epsilon$$
Figure 4.3: The ratio of MSPE for LAD to MSPE for OLS for a family of $t$ distributed $\epsilon$. Each line represents a $t$-distributed $\epsilon$ with different degrees of freedom $\nu$. 
The expectation of its MSPE over sample data \((X, Y)\) is

\[
E(\text{MSPE}(\hat{\beta}_{OLS})) = E \left[ \sigma^2 + (\hat{\beta}_{OLS} - \beta)' \Sigma (\hat{\beta}_{OLS} - \beta) \right] \\
= \sigma^2 + \text{Tr} \left( E \left[ ((X'X)^{-1}X'\varepsilon)' \Sigma ((X'X)^{-1}X'\varepsilon) \right] \right) \\
= \sigma^2 + \text{Tr} \left( \Sigma E \left[ ((X'X)^{-1}X'\varepsilon\varepsilon'X(X'X)^{-1}) \right] \right) \\
= \sigma^2 + \sigma^2 \text{Tr} \left( \Sigma E \left[ (X'X)^{-1} \right] \right) \\
= \sigma^2 \left( 1 + \frac{p}{n-p-1} \right) \\
= \frac{\sigma^2}{1 - \frac{p}{n-1}}
\]

So, letting \(n \to \infty\) with \(p/n \to \rho\), \(E(\text{MSPE}(\hat{\beta}_{OLS})) \to \frac{\sigma^2}{1-\rho}\). In other words, the expected performance of the OLS estimator only depends on the noise level and \(\rho\). Simulations (not shown) suggests that this is true for the LAD estimator as well. We conclude that our simulations for \(n = 100\) is representative of the high dimensional asymptotic setting.

### 4.4 Conclusion

Our goal in this chapter was to understand if classical robustness results carry over into the high dimensions. In particular, we investigated the robustness of \(L_1\) methods relative to \(L_2\) methods against heavy tail distributions. We find both expected and unexpected results through our simulations.

In the \(p > n\) case, we added an \(L_1\) penalty to the loss functions and verified that \(L_1L_1\) does better than \(L_2L_1\) as the tails get heavier. However, we note that classical results do not translate exactly. In particular, in the double exponential case, both methods yield about the same MSPE. Furthermore, \(L_1L_1\) results in estimates with higher variance than \(L_2L_1\) for the double exponential case across a wide variety of problem scenarios.

In the \(p \leq n\) case, we looked at OLS and LAD directly. We see the classical efficiency results do not hold as we allow \(p\) to grow with \(n\), \(p/n \to \rho\). In particular, for the double exponential case, LAD has a lower MSPE than OLS as we increase \(\rho\) past 0.3. We also studied a family of \(t\) tails, and verified the classical result that the heavier the tails, the more LAD outperforms OLS for small \(\rho\). This again comes with the caveat that as we increase \(\rho\), OLS eventually overtakes LAD.
### Table 4.5: Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j = 1$. $p = 200$. This is the extended version of Table 4.2.

<table>
<thead>
<tr>
<th>Rho</th>
<th>Scale</th>
<th>$\text{Bias}^2$</th>
<th>Var</th>
<th>Mean MSPE</th>
<th>Mean $L_1$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$L_2 L_1$</td>
<td>$L_1 L_1$</td>
<td>$L_2 L_1$</td>
<td>$L_1 L_1$</td>
</tr>
<tr>
<td>0</td>
<td>0.05</td>
<td>0.0177</td>
<td>0.0148</td>
<td>0.0181</td>
<td>0.0233</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.0601</td>
<td>0.0479</td>
<td>0.0738</td>
<td>0.0936</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>1.69</td>
<td>1.42</td>
<td>1.78</td>
<td>2.03</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5.96</td>
<td>5.36</td>
<td>2.21</td>
<td>2.96</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>8.54</td>
<td>7.63</td>
<td>1.72</td>
<td>2.7</td>
</tr>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>0.0289</td>
<td>0.0104</td>
<td>0.0158</td>
<td>0.0186</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.0484</td>
<td>0.0386</td>
<td>0.059</td>
<td>0.0664</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.956</td>
<td>0.935</td>
<td>1.31</td>
<td>1.45</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>2.5</td>
<td>2.29</td>
<td>2.93</td>
<td>3.09</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>4.31</td>
<td>3.87</td>
<td>5.45</td>
<td>6.51</td>
</tr>
</tbody>
</table>

### Table 4.6: Bias-Variance breakdown of the mean MSPE for the double exponential case. Non-zero $\beta_j = 1$. $p = 400$.

<table>
<thead>
<tr>
<th>Rho</th>
<th>Scale</th>
<th>$\text{Bias}^2$</th>
<th>Var</th>
<th>Mean MSPE</th>
<th>Mean $L_1$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$L_2 L_1$</td>
<td>$L_1 L_1$</td>
<td>$L_2 L_1$</td>
<td>$L_1 L_1$</td>
</tr>
<tr>
<td>0</td>
<td>0.05</td>
<td>0.0250</td>
<td>0.0211</td>
<td>0.0228</td>
<td>0.0304</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.107</td>
<td>0.0896</td>
<td>0.0925</td>
<td>0.134</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>2.52</td>
<td>2.22</td>
<td>2.01</td>
<td>2.38</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7.58</td>
<td>6.85</td>
<td>1.53</td>
<td>2.24</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>9.33</td>
<td>8.50</td>
<td>1.13</td>
<td>2.85</td>
</tr>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>0.0412</td>
<td>0.0161</td>
<td>0.0191</td>
<td>0.0227</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.0701</td>
<td>0.0624</td>
<td>0.0653</td>
<td>0.0858</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>1.63</td>
<td>1.52</td>
<td>1.59</td>
<td>1.7</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>3.15</td>
<td>2.79</td>
<td>2.95</td>
<td>3.00</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>5.07</td>
<td>4.76</td>
<td>4.98</td>
<td>5.22</td>
</tr>
</tbody>
</table>
### Table 4.7: Bias-Variance breakdown of the mean MSPE for the double exponential case.

Non-zero $\beta_j \propto \frac{1}{(25 + j^2)}, \sum_j \beta_j^2 = s$. $p = 200$. This represents the case where $\beta$ has a moderate decay.

<table>
<thead>
<tr>
<th>Rho</th>
<th>Scale</th>
<th>Bias$^2$</th>
<th>Var</th>
<th>Mean MSPE</th>
<th>Mean $L_1$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$L_2L_1$</td>
<td>$L_1L_1$</td>
<td>$L_2L_1$</td>
<td>$L_1L_1$</td>
</tr>
<tr>
<td>0</td>
<td>0.05</td>
<td>0.0178</td>
<td>0.0144</td>
<td>0.0191</td>
<td>0.0224</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.0627</td>
<td>0.0503</td>
<td>0.0762</td>
<td>0.0961</td>
</tr>
<tr>
<td>0.5</td>
<td>1.33</td>
<td>1.16</td>
<td>1.34</td>
<td>1.47</td>
<td>2.67</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>8.19</td>
<td>7.14</td>
<td>1.81</td>
<td>2.67</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0209</td>
<td>0.00912</td>
<td>0.0149</td>
<td>0.0158</td>
<td>0.0358</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.0510</td>
<td>0.0429</td>
<td>0.0581</td>
<td>0.0677</td>
</tr>
<tr>
<td>0.5</td>
<td>0.801</td>
<td>0.800</td>
<td>1.15</td>
<td>1.16</td>
<td>1.95</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>2.24</td>
<td>1.96</td>
<td>2.69</td>
<td>2.67</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>4.14</td>
<td>3.86</td>
<td>4.68</td>
<td>4.50</td>
</tr>
</tbody>
</table>

### Table 4.8: Bias-Variance breakdown of the mean MSPE for the double exponential case.

Non-zero $\beta_j \propto \frac{1}{(5 + j^2)}, \sum_j \beta_j^2 = s$. $p = 200$. This represents the case where $\beta$ has a steep decay.

<table>
<thead>
<tr>
<th>Rho</th>
<th>Scale</th>
<th>Bias$^2$</th>
<th>Var</th>
<th>Mean MSPE</th>
<th>Mean $L_1$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$L_2L_1$</td>
<td>$L_1L_1$</td>
<td>$L_2L_1$</td>
<td>$L_1L_1$</td>
</tr>
<tr>
<td>0</td>
<td>0.05</td>
<td>0.0170</td>
<td>0.0119</td>
<td>0.0164</td>
<td>0.0242</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.0617</td>
<td>0.0487</td>
<td>0.0664</td>
<td>0.0832</td>
</tr>
<tr>
<td>0.5</td>
<td>1.11</td>
<td>0.851</td>
<td>0.733</td>
<td>0.832</td>
<td>1.84</td>
</tr>
<tr>
<td>0</td>
<td>3.00</td>
<td>2.49</td>
<td>1.44</td>
<td>1.74</td>
<td>4.44</td>
</tr>
<tr>
<td>0.5</td>
<td>7.08</td>
<td>6.13</td>
<td>1.82</td>
<td>2.63</td>
<td>8.9</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0148</td>
<td>0.00984</td>
<td>0.0150</td>
<td>0.0170</td>
<td>0.0298</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0478</td>
<td>0.0382</td>
<td>0.0584</td>
<td>0.0664</td>
<td>0.106</td>
</tr>
<tr>
<td>0.5</td>
<td>0.619</td>
<td>0.545</td>
<td>0.726</td>
<td>0.726</td>
<td>1.35</td>
</tr>
<tr>
<td>0.5</td>
<td>1.77</td>
<td>1.42</td>
<td>1.85</td>
<td>1.84</td>
<td>3.62</td>
</tr>
<tr>
<td>0.5</td>
<td>3.62</td>
<td>3.21</td>
<td>4.09</td>
<td>3.87</td>
<td>7.70</td>
</tr>
</tbody>
</table>
Bibliography


