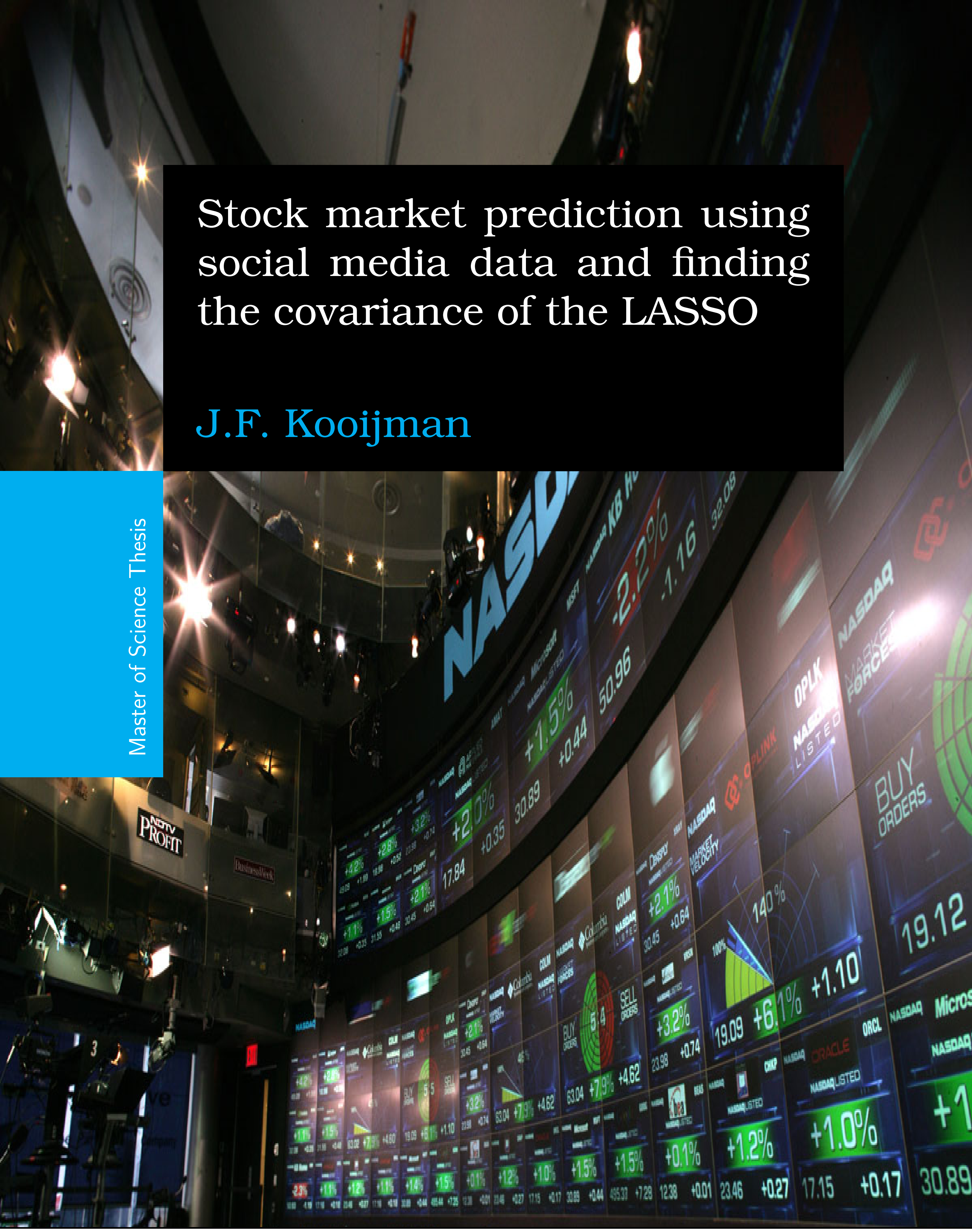


Stock market prediction using social media data and finding the covariance of the LASSO

J.F. Kooijman

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Stock market prediction using social media data and finding the covariance of the LASSO

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J.F. Kooijman

December 8, 2014

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Stock market prediction has been a research topic for decades; recently, efforts to increase the accuracy by including data from social media like Google and Twitter received a lot of attention. Social media can be regarded as indicator for sentiments and sentiments are known to influence the stock market. Current models lack interpretation; it is difficult to determine what data is relevant for stock market prediction, since there is an abundance of social media data. A regression method that induces sparsity is thus required; data that is not useful is discarded automatically. The LASSO induces sparsity via L1-regularization; however, the covariance and confidence of the found regression coefficients cannot be derived easily, while this is important for interpretation. This thesis therefore reviews all known methods for approximating the covariance and confidence interval for the LASSO and determines their accuracy using numerical simulations. A new method is proposed based on the Unscented Transform, which outcompetes all methods in the underdetermined scenario, where there are more features than data points. Unfortunately, linear regression via the LASSO has limited use for stock markets as the achieved prediction accuracy is low. Nonlinear models are often applied for stock market prediction to achieve higher accuracies. Therefore a new feature selection method is proposed for the nonlinear Support Vector Regression (SVR) to select the correct data for stock market prediction using the SVR. This method yields accurate feature selection when the number of features to select from is low.

Table of Contents

List of Figures

List of Tables

“One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man.” — *Elbert Hubbard*

Acknowledgements

The idea for this thesis was found at the end of 2012, where the notion of using social media for stock market prediction was something of the imagination. After some research it was found to be a new and growing literature and became a proposal for this MSc. project. This was of course more challenging than originally expected, but a clear contribution has been made.

I would like to thank my supervisor, prof. dr. ir. Michel Verhaegen, for his support and guidance on this project; his enthusiasm was a great motivation. Tomas Burki, with whom I collaborated for most of the project, I would like to thank for his feedback and the discussions we had; working together was not only fun but also instructive. I would also like to thank my family and friends for the support over the last year. Especially my parents, Frank and Nora, and my sister Saskia for their sympathetic ear to the frustrations encountered. Lastly, of course her, Pieternella, for being her.

Delft, University of Technology J.F. Kooijman
December 8, 2014

Chapter 1

Introduction

For decades, prediction of the stock markets has been of great interest among traders, economists, businesses and consumers. Recently, the theory of behavioral economics is gaining attention in problems where the prediction of the stock market is of interest (Smailovic et al., 2013). Briefly stated, behavioral economics argues that decisions made on the stock market are influenced by social and emotional factors. If these emotions could be captured and used for prediction of the stock market, more reliable predictions may be obtained. For example, during the recent financial crisis no one seemed to be able to predict what would happen on the stock market since the models used for prediction are based on fundamental price movements and not on sentiments. So, even though the models are quite involved and the trading strategies carefully chosen, the return rates were negative when the stock market was dominated by emotions (Anderluh, 2011).
This emphasizes the need for developing predictive models that can also capture sentiments and emotions, since we suspect that they will play a large role in the determination of the stock price.
Sentiments and emotions can be captured using data extracted from social media; using social media data for stock market prediction is a relative new idea and is drawing more and more attention in the scientific world. For example (Bollen et al., 2010) added extracted information from Twitter to nonlinear prediction models and showed a significant increase in accuracy.
One of the problems of this method is that high accuracy is achieved when useful social media information is added, but if information is added with low predictive power the accuracy drops.
Thus, a method should be found that automatically detects what social media data is useful for prediction and discards the non-useful data from the prediction model. This would give a robust accuracy of the predictions and furthermore make the results more interpretable as it is visible what data is used.
In this thesis work the first idea was to apply a sparse linear regression method, which automatically discards the unnecessary data. A well-known and easy to apply linear sparse regression method is the Least Absolute Shrinkage and Selection Operation (LASSO). The LASSO uses a linear relation and thus gives a regression coefficient per

feature, which is equal to zero if the feature is not used. To allow more interpretation it would be desirable to know not only this regression coefficient but also its variance and confidence interval. Finding an accurate method to retrieve the variance and confidence interval is the first subject of this thesis.

Applying linear models, such as the LASSO, to real stock market data it was found that their accuracy is lower compared to nonlinear prediction models. For stock market prediction models accuracy is very important and nonlinear models are often applied, just like (Bollen et al., 2010) did. Thus, the linear prediction model using the LASSO did not give the desired results. The logical continuation of the research was to search for a new feature selection method, now for nonlinear prediction models. The second subject covered in this thesis is on feature selection for nonlinear prediction models.

In this chapter first the working mechanism of stock markets is explained in Section 1-1, followed by a short overview of the literature on using social media data for stock market predictions in Section 1-2. Then in Section 1-3 the applied mathematical framework of this report is presented. In Section 1-4 for the subject of the confidence interval of the LASSO a short introduction and an explanation of the used research questions are given. The same is done in Section 1-5 for the subject of finding the feature selection of nonlinear models. Finally, an outline of the remainder of the report is given in Section 1-6.

1-1 Stock markets

Stock markets are complex systems and are not simply governed by linear or nonlinear equations. The price of stocks is directed by the buyers and sellers on the market. These buyers and sellers are placing bids, and a market clearing mechanism matches the demand (buyers) and supply (sellers) to determine the price. This clearing mechanism is illustrated in Figure 1-1. Thus, an accurate model of the stock market would be a multi-agent model, where the agents represent the buyers and sellers. (LeBaron, 2006) and (E. Samanidou and Lux, 2007) give an overview of these models.

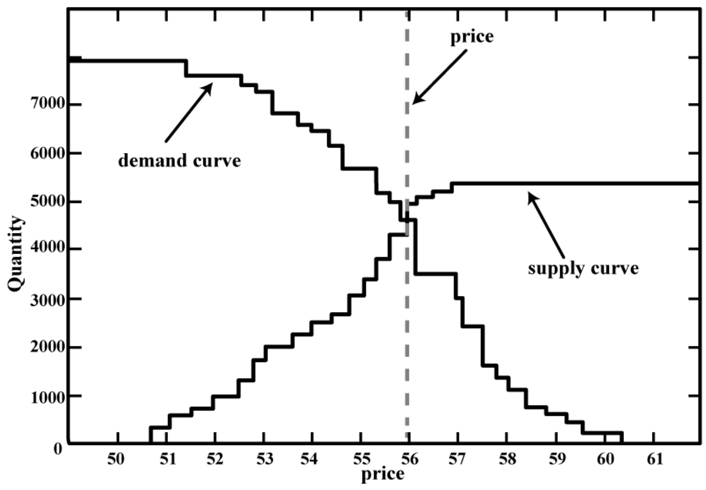


Figure 1-1: Deriving a market price by matching the supply curve with the demand curve.

1-2 Using social media in stock market prediction

Nowadays, a huge amount of information is shared by people on social media sites such as Twitter, Google and Facebook. For the first time, sentiments of large groups of people can be used for prediction by extracting data from the vast amount of data in social media. One of the earliest papers on this is by (Bollen et al., 2010), who show that extracted moods from Twitter data have predictive power for stock markets. In this article a refined algorithm by Google is used to extract mood-indices, such as the calmness, happiness and alertness, from the massive amount of Twitter data by classifying them looking at words in the tweets. A Granger causality analysis is used to show the correlation of these moods with the stock market and upon adding these moods to a nonlinear stock market prediction model an increase of accuracy is found. Similar examples are the work of (Asur and Huberman, 2010) and (Jiang et al., 2013). This method is not limited to Twitter data, (Mao et al., 2011) show similar results using Google Insights for Search (GIS). GIS provides data on the use of a search term on Google. Selecting a few search terms Mao shows a correlation with the usage of these search terms and the Dow Jones stock market value.
A similar study is done by (Rao and Srivastava, 2012) who use classificated data from Twitter as well as GIS to predict oil funds, gold price and Dow Jones. (Beer et al., 2013) used GIS to extract investor sentiment and finding its influence on the stock market.
(Preis et al., 2013) looked closely at which search queries of GIS had the largest influence on the Dow Jones index during the recent financial crisis in order to obtain "early warning signs".

1-3 Mathematical framework

The phenomenon that we want to predict, e.g. the daily closing price of the stocks is defined in this report as the vector $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^T$. Thus, the vector contains the time series data from time $n = 1, ..., N$. We assume that the prediction of $\mathbf{y}$ depends on (i) previous values of $\mathbf{y}$, (ii) technical indicators, e.g. averages and momentum indicators, and (iii) data from social media. Data from these three different data sources will be called features and their time series are represented in the feature vector $\mathbf{f}_p = \begin{bmatrix} f_{1.p} & \dots & f_{N,p} \end{bmatrix}^T$ for feature $p = 1, ..., P$. These features are collected in data matrix $A^{N \times P}$. The rows of A are denoted by $\mathbf{a}_n$, which contain all P features at time n. A specific entry of the matrix is referred to as $a_{n,p}$. Then, we assume the following relation to predict the stock market closing price:

$$\mathbf{y} = f(A) + \boldsymbol{\varepsilon}$$

Where ϵ is zero mean white noise: $E[\epsilon] = 0$ and $\text{var}(\epsilon) = \sigma_\epsilon^2$. For the linear scenario, this becomes:

$$\mathbf{y} = A\mathbf{x} + \boldsymbol{\varepsilon}$$

Here $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_P \end{bmatrix}^T$, is a vector that contains the coefficients for each feature. For identifying a regression model between A and $\mathbf{y}$, the following minimization problem is of interest:

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 \tag{1-1}$$

The minimization problem of (1-1) is illustrated in Figure 1-2.

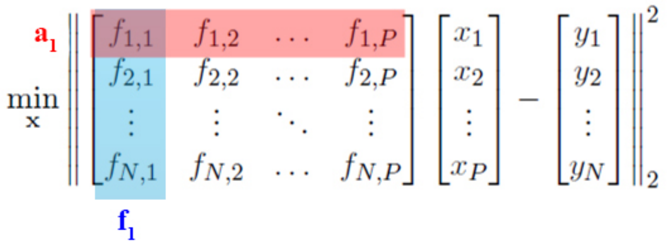

$\mathbf{a}_1$
$\min_{\mathbf{x}} \left\| \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,P} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N,1} & f_{N,2} & \cdots & f_{N,P} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_P \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \right\|_2^2$
$\mathbf{f}_1$

Figure 1-2: Illustration of the minimization problem $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2$ and the definition of the vectors.

1-4 Covariance and confidence interval of the LASSO

The LASSO is a regression technique that gives a sparse solution of the regression problem of (1-1), i.e. several elements of $\mathbf{x}$ are set to zero. The minimization problem of the LASSO as proposed by (Tibshirani, 1994) is given in (1-2), where the ℓ_1-norm is implemented in order to obtain a sparse vector $\mathbf{x}$. The parameter λ in (1-2) is the tuning parameter of the LASSO minimization problem and controls the amount of shrinkage that is applied.

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{1-2}$$

When applying this technique it is useful to know what the covariance and the confidence interval is of a regression coefficient. For example, the coefficient x_2 is between 0.45 - 0.55 with 90% accuracy. Having an estimate of variance of the found regression coefficients is valuable as it increases the interpretability and indicates the accuracy of the found model. A possible result of the method is illustrated in Figure 1-3.

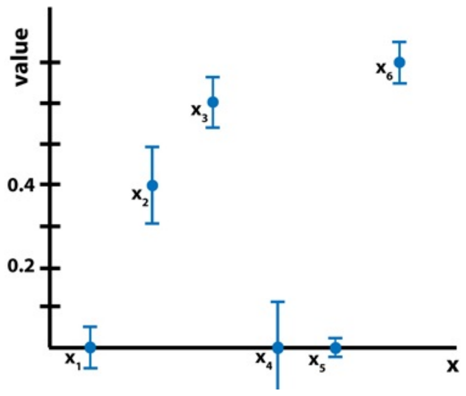


Figure 1-3: Illustration of estimates of the LASSO with confidence intervals.

1-4-1 The difficulty of finding the covariance

For simple regression methods the covariance can be found using an analytical formula. For example for the Ordinary Least Squares (OLS) given in (1-1), the solution of the covariance can be derived as in (1-3).

$$\begin{aligned}
J &= \|A\mathbf{x} - \mathbf{y}\|_2^2 = \mathbf{x}^T(A^TA)\mathbf{x} - \mathbf{y}^TA\mathbf{x} - \mathbf{x}^TA^T\mathbf{y} + \mathbf{y}^T\mathbf{y} \\
\frac{\delta J}{\delta \mathbf{x}} &= 2(A^TA)\mathbf{x} - 2A^T\mathbf{y} \qquad = 0 \text{ (solution)} \\
\mathbf{x} &= (A^TA)^{-1}A^T\mathbf{y} \\
E[\mathbf{x}] &= E[(A^TA)^{-1}A^T(A\mathbf{x} + \epsilon)] = E[\mathbf{x}] + E[\epsilon] = \mathbf{x} \\
\Sigma_{\mathbf{x}} &= E[\mathbf{x}^2] - E[\mathbf{x}]^2 = E\Big[\big((A^TA)^{-1}A^T(A\mathbf{x} + \varepsilon)\big)\big((A^TA)^{-1}A^T(A\mathbf{x} + \varepsilon)\big)^T\Big] - \mathbf{x}\mathbf{x}^T \\
&= E\Big[\big(\mathbf{x} + (A^TA)^{-1}A^T\varepsilon)\big)\big(\mathbf{x} + (A^TA)^{-1}A^T\varepsilon)\big)^T\Big] - \mathbf{x}\mathbf{x}^T \\
&= (A^TA)^{-1}\hat{\sigma}_\varepsilon^2
\end{aligned} \tag{1-3}$$

However, this method is not applicable for the LASSO as no analytical solution can be found, since its objective function is non-differentiable; $\frac{\delta J}{\delta \mathbf{x}}$ cannot be derived because of the ℓ_1-norm in the LASSO objective. This makes the finding the covariance of the LASSO difficult.
Therefore, many methods are based on bootstrapping, which is a computationally expensive way to find the covariance and confidence intervals. However, these methods are said to be biased estimators; even for an infinite amount of data points, $N \to \infty$, the true value of covariance and confidence intervals are not found.

1-4-2 Goals

Finding an accurate method to retrieve the covariance and confidence intervals of the LASSO regression coefficients is the goal of this subject. Specifying this goal the three following research questions are formulated:

Does the theoretical probability density function of the LASSO coefficients correspond with the numerical version?
In the literature the asymptotic behavior of the approximation methods was analyzed; and it was found that bootstrap methods give an inconsistent estimate. This means that based on the mathematical properties in case of infinite amount of samples it still has a bias. For finite sample behavior similar conclusions have been drawn, a finite sample probability density function has been used to show the large errors of bootstrap methods. To better understand where these biases comes from, the (mathematical) probability density function for a finite number of samples will be compared with the numerical version, which is retrieved using a Monte Carlo method. The Monte Carlo method is a valid method to give a distribution of $\mathbf{x}$ that would be obtained in practice.

Which of the existing methods has the best finite sample behavior for finding the covariance and confidence interval of the regression coefficients of the LASSO?
A complete overview of all the available methods for finding the covariance of the LASSO is to be made. In the literature many methods are mentioned but only very limited comparisons of accuracy exist. These methods vary greatly in approach and associated numerical complexity; a comparison of their performance in different situations is valuable. Thus the goal is to find out which one performs best in finite sample behavior. Only finite sample behavior is considered, with a real number of data points ($N \in \mathbb{R}$). The results of asymptotic behavior carry little meaning for the finite sample behavior, as crucial differences are not present in the asymptotic behavior. (Leeb and Potscher, 2005)

Can the Unscented Transform be used to recover the covariance and confidence interval of the regression coefficients of the LASSO?
Many of the proposed methods in literature are computationally expensive. Therefore, the Unscented Transform (UT) is applied in order to find a more computational efficient method to find the covariance and confidence interval of the regression coefficients of the LASSO. The UT has, to the best of our knowledge, not been applied for finding the covariance of the LASSO before or any other regression method in literature. It will be exciting to see if this method is able to recover the covariance and confidence interval of the LASSO and to find the relation between accuracy and computational complexity.

1-5 Feature selection for nonlinear stock market prediction

The linear LASSO did not give the desired results, as their accuracy is lower compared to nonlinear prediction models. The logical continuation of the research was to search for a new feature selection method, now for nonlinear prediction models.
The first question would be which nonlinear model would be investigated, the choice here is made on the Support Vector Regression (SVR). The SVR is selected as it has been shown to achieve high generalization and is fast to solve and tune (Li and Liu, 2011). A good alternative as nonlinear prediction method would be neural networks, also often applied to stock market prediction. But in contrast to Neural Networks, the SVR has a global optimum and exhibits better prediction accuracy due to taking into account the training error and the complexity of the regression model (Yeh et al., 2011). The SVR has been applied successfully on numerous occasions to predicting stock markets. For example (Kao et al., 2013) have applied it to the Shanghai Stock Exchange Composite and the Nikkei index and (Dai and Lu, 2008) have applied it to the Nikkei index.

1-5-1 Feature selection methods: wrapper and filter

Feature selection is the process of selecting which features are relevant for the prediction and which should be discarded. Many feature selection methods exist for Support Vector Regression and there are generally two approaches: filter and wrapper feature selection. Filter feature selection are methods that perform the feature selection independent of the SVR, for example they use the Pearson Correlation Coefficient to decide which features are selected (Guyon and Elisseeff, 2003) (Hai-xian Zhao, 2011). This means that a

different relation is used to measure which features are useful and the feature selection becomes a preprocessing step. Wrapper feature selection methods do rely on the SVR model and usually yield better performance compared to filter methods. This makes sense, since the wrapper feature selection is based on the predictive power of the data in the SVR.
In general many filter feature selection methods have been applied as a preprocessing step for stock market prediction leading to a complex and ambiguous method consisting of several steps. In this report a more embedded approach is sought by using a wrapper feature selection method. The wrapper feature selection method relies on the SVR and not on other (linear) relations between the features and output data.

1-5-2 Goals

Finding a SVR feature selection method for stock market prediction is the goal of this subject. Specifying this goal the three following research questions are formulated:

What is the most suited wrapper feature selection method for fast and accurate feature selection for the SVR?
The SVR, being a nonlinear model, is sensitive to nonsense-data: information with low predictive power will decrease the accuracy of the model. This means that a feature selection method should be found for the SVR. This feature selection method should not only be accurate, but also with a reasonable computational complexity. The computational complexity is a very important aspect as evaluation of all possible combinations of features is infeasible. When there are P features present, 2^P subsets of the features are possible; this is even for low numbers of P infeasible to calculate.

Does a time-varying SVR yield a higher accuracy in predicting stock markets compared to a time-invarying SVR?
Stock markets can be seen as time-varying systems; the correlation between social media data and the stock market is definitely time-varying as will be shown in this report. Taking in account time-varying behavior could yield increased accuracy as now time-varying correlations can be taken into account. This is especially useful for using social media data in stock market prediction, as no time-invariant correlation can be expected. For example, a search term related to the credit crisis is probably not equally relevant for months. Assuming such a constant correlation for a long horizon would probably lead to incorrectly taking the data into account when it is no longer useful or excluding it in time periods for which the data could provide useful information.
Truly time-varying extensions to SVR are not known. To implement time-varying behavior into the SVR the approach of horizoning is applied; the SVR is trained at each time-step over a limited set of previous data points to only focus on the most recent history. This method is simple to implement and was also previously applied by (Elmer, 2011) using the LASSO.

What is the most suited wrapper feature selection method for fast and accurate feature selection for time-varying SVR?
For the time-varying SVR fewer data points are available as the horizon is desired to be short. Thus, a feature selection method should be found which is capable of selecting the features, even when the amount of data points is low.

1-6 Outline

In Chapter 2 the research regarding the covariance is presented and in Chapter 3 the research regarding the feature selection for nonlinear prediction models is given. The conclusions and recommendations regarding both subjects are given in Chapter 4.

Chapter 2

Covariance and confidence interval of the LASSO

2-1 Introduction

Having an estimate of variance of the found regression coefficients is valuable as it increases the interpretability and indicates the accuracy of the found model. In the literature many methods for finding the covariance are described, their accuracy will be investigated in this chapter. As an alternative to these existing methods the UT will be also applied to find the covariance, which has, to the best of our knowledge, not been applied for finding covariance of the LASSO before or any other regression method in literature.

First, an overview of the different methods of finding the covariance of the LASSO is described in Section 2-2. The general algorithm for the UT is described in Section 2-3 along with the designed algorithms for the LASSO. The choice of the LASSO solver is important for the numerical simulations and will be discussed in Section 2-4. To make a link between the theoretical and practical properties of the LASSO, the probability distribution function will be compared with a Monte Carlo approximation in Section 2-5. Then the methods are compared in a several scenarios in Section 2-6. Finally a discussion is given in Section 2-7.

2-2 Methods of finding covariance and confidence intervals

Many methods for finding the covariance of the regression coefficients and their respective confidence intervals exist. Three analytical approaches were found, the simplest being an approximation of the ℓ_1 norm by an ℓ_2 norm. Many bootstrap variants also exist, (Sartori, 2011) gives an overview of them. The analytical formula of the finite sample distribution function was derived by (Pötscher and Leeb, 2009). For all these methods their respective algorithm is given and their properties are discussed. First the Monte Carlo approach is given, since it used to obtain the 'true covariance'.

2-2-1 Monte Carlo (true covariance)

In a controlled experiment data sets can be generated based on statistical properties. This allows generating a large quantity of data sets based on exactly the same statistical properties. Applying the LASSO on each of these data sets, the distribution of $\mathbf{x}$, along with the covariance and confidence intervals, can be found. This will be is used as a benchmark for comparison of the approximations. This principle is written out in Algorithm 1; in all the simulations of this chapter the parameter for the amount of iteration M is set to $M = 10^5$. The generating function for the feature matrix A is denoted as $f_{generating}((\boldsymbol{\mu}^A)^2, (\boldsymbol{\sigma}^A)^2)$. This function uses a mean and variance defined for each element of the regression vector: $\boldsymbol{\mu}^A = \left[\mu_1^A \dots \mu_P^A\right]^T$, $\boldsymbol{\sigma}^A = \left[\sigma_1^A \dots \sigma_P^A\right]^T$.
The notation $\{\mathbf{x}^1 \dots \mathbf{x}^M\}_{(16\%)}$ refers to finding the 16% percentile value of the set of $\{\mathbf{x}^1 \dots \mathbf{x}^M\}$. The algorithm to find this value is written in Algorithm 2 for completeness; note that these percentiles are found for each regression coefficient separately.

Algorithm 1 Monte Carlo Method for covariance of the LASSO

```
   Iterate to generate set of solutions
1: for $i = 1 \to M$ do
2:     $A^i \leftarrow f_{generating}(\boldsymbol{\mu}^A, (\boldsymbol{\sigma}^A)^2)$
3:     $\mathbf{y}^i \leftarrow A^i\mathbf{x} + \mathcal{N}(\mathbf{0}, I_N\sigma_\varepsilon^2)$
4:     $\mathbf{x}^i = \text{argmin}_{\mathbf{x}} \|A^i\mathbf{x} - \mathbf{y}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
5: end for
   Calculate mean and covariance
6:     $\bar{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}^i$
7:     $\Sigma_{\mathbf{x}}^{MC} = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})^T$
   Calculate confidence intervals
8:     $\mathbf{x}_{(16\%)}^{MC} = \{\mathbf{x}^1 \dots \mathbf{x}^M\}_{(16\%)}$
9:     $\mathbf{x}_{(84\%)}^{MC} = \{\mathbf{x}^1 \dots \mathbf{x}^M\}_{(84\%)}$
```

2-2-2 Analytical approximation of Tibshirani

In the first paper on the LASSO by (Tibshirani, 1994) an analytical approximation for finding the covariance was suggested. It is based on the fact that the LASSO can be

Algorithm 2 Finding the α-percentiles value of a data set $\mathbf{x}^1 \dots \mathbf{x}^M$

```
1: for $p = 1 \rightarrow P$ do
2:     Sort the values of $x_p^1 \dots x_p^M$ from high to low: $x_p^{*1} \dots x_p^{*M}$
3:     $x_{p(\alpha\%)} \leftarrow$ linear interpolation between $x_p^{*floor(\alpha M/100)}$ and $x_p^{*ceil(\alpha M/100)}$
4: end for
5: $\mathbf{x}_{(\alpha\%)} \leftarrow \left[x_{1(\alpha\%)} \dots x_{P(\alpha\%)}\right]^T$
```

written as a generalized Ridge regression by rewriting the one norm as a two norm, see (2-1). The analytical solution of the covariance matrix of Ridge regression problem is known. The derivation is given in Appendix A-1 and the resulting equation in (2-2), where $W^- = (G^T G)^{-1}$. It can be seen that it is required that $A^T A + \lambda W^-$ must be invertible, $A^T A$ is not invertible when $P > N$.
Note that this is an approximation because the matrix G is dependent on the solution $\mathbf{x}$. Furthermore, since this formula gives the covariance and not the distribution of $\mathbf{x}$ it does not give a confidence interval. This requires an assumption on the underlying distribution function. If a Gaussian distribution is assumed, the 68% confidence interval for regressor p is given by (2-3). $\Sigma_{\mathbf{x}}(p,p)$ refers to the (p,p) entry of the matrix $\Sigma_{\mathbf{x}}$.
The Approximation by Tibshirani has the disadvantage that the entry $\sqrt{\frac{1}{|x_i|}}$ is undefined if x_i is zero. This can cause the approximation to break down. Therefore, a Moore-Penrose pseudoinverse is applied in (2-2). This effect was recognized by (Osborne et al., 2000), who rewrote the formula to prevent the breaking down, which is described in the next section.

$$\begin{aligned} \min_{\mathbf{x}} \quad ||A\mathbf{x} - \mathbf{y}||_2^2 + \lambda||\mathbf{x}||_1 &= \min_{\mathbf{x}} \quad ||A\mathbf{x} - \mathbf{y}||_2^2 + ||G\mathbf{x}||_2^2 \\ G &= \begin{bmatrix} \sqrt{\frac{\lambda}{|x_1|}} & 0 & .. & 0 \\ 0 & \sqrt{\frac{\lambda}{|x_2|}} & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & \sqrt{\frac{\lambda}{|x_p|}} \end{bmatrix} \end{aligned} \tag{2-1}$$

$$\begin{aligned} \Sigma_{\mathbf{x}} &= \Big(A^T A + \frac{\lambda}{2} W^-\Big)^{-1} A^T A \Big(A^T A + \frac{\lambda}{2} W^-\Big)^{-1} \\ W^- &= \text{ Moore-Penrose pseudoinverse of } \begin{bmatrix} |x_1| & 0 & .. & 0 \\ 0 & |x_2| & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & |x_p| \end{bmatrix} \end{aligned} \tag{2-2}$$

$$[x_{p(16\%)}, x_{p(84\%)}] = [x_p - \sqrt{\Sigma_{\mathbf{x}}(p,p)}, x_p + \sqrt{\Sigma_{\mathbf{x}}(p,p)}] \tag{2-3}$$

2-2-3 Analytical approximation of Osborne

(Osborne et al., 2000) rewrote the formula of Tibshirani into (2-4) to prevent the breaking down because of coefficients being zero. The resulting formula is quite similar, only the major advantage of this formula is that the division now occurs over $||\mathbf{x}||_1$ instead of over the individual absolute values. This prevents the breaking down when elements of $\mathbf{x}$ are zero. The derivation of the formula is given in Appendix A-2. If a Gaussian distribution is assumed, the 68% confidence interval for regressor p is given by (2-3). Note that this analytical approximation has the same requirement of $A^TA + W_2$ being invertible.

$$\begin{aligned}\Sigma_{\mathbf{x}} &= (A^TA + W_2)^{-1}A^TA(A^TA + W_2)^{-1}\hat{\sigma}_\varepsilon^2 \\ A^TA + W_2 &= A^T\Big(I_N + \frac{(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^T}{||\mathbf{x}||_1||A^T(\mathbf{y} - A\mathbf{x})||_\infty}\Big)A\end{aligned} \tag{2-4}$$

2-2-4 Sandwich formula

Almost all literature on statistical properties of the LASSO mentions the Sandwich formula proposed by (Fan and Li, 2001). This formula only recovers the covariance of the non-zero elements of $\mathbf{x}$ and is therefore rarely applied in comparisons. Their formula for the covariance is derived for a class of general sparse regression methods. After rewriting this formula for the LASSO in specific, I found out that it is same formula as the approximation by Tibshirani. This proof is given in Appendix A-3.

2-2-5 Vector resampling

Vector resampling is the simplest bootstrap method. Bootstrapping is a simulation method for finding the statistical inference and is straightforward to apply. In vector resampling a new data set is created from the data set of A and $\mathbf{y}$ by randomly choosing rows from A and $\mathbf{y}$. The resampling chooses N rows denoted as $\tilde{\mathbf{a}}_j$ and N output values $\tilde{y}_j$from the N original elements of $\mathbf{y}$ and A with an equal probability of $1/N$. So, probably not all the original samples are present in the generated data set and some duplicates are present. Using this new data set an estimate $\mathbf{x}^i$ can be calculated. After performing this a larger number of times, B, a distribution of $\mathbf{x}$ is obtained. From this distribution the covariance and confidence intervals can easily be obtained. The method is written out in Algorithm 3 (Dekking et al., 2005)(Sartori, 2011).
Bootstrapping in general has the advantage that it also gives a confidence interval of the regression coefficients, since a distribution of $\mathbf{x}$ is retrieved. For the analytical approximations this could only be done by assuming a Gaussian distribution, which is not a valid assumption. The disadvantage is that the method is more computational expensive, since a LASSO solution must be calculated B times.
Vector resampling is a nonparametric bootstrapping method. This means that it does not require any assumptions on the underlying model, for example $\mathbf{y} = A\mathbf{x} + \boldsymbol{\varepsilon}$. It is a good approach if the input matrix A is randomly generated and N is large. If the A-matrix is not randomly generated or there a few data points, residual resampling is a better approach.

Algorithm 3 Vector resampling
$u_{dist}(n)$ refers to a uniform discrete distribution with range [1 .. N]

```
    Calculate LASSO solution
1:    $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    Iterate to generate set of solutions
2:  for $i = 1 \to B$ do
3:      for $n = 1 \to N$ do
4:          $index \leftarrow u_{dist}(N)$
5:          $\tilde{\mathbf{a}}_n^i = \mathbf{a}_{index}$
6:          $\tilde{y}_n^i = \mathbf{y}_{index}$
7:      end for
8:      $\tilde{\mathbf{x}}^i = \text{argmin}_{\mathbf{x}} \|\tilde{A}^i\mathbf{x} - \tilde{\mathbf{y}}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
9:  end for
    Calculate the covariance
10:     $\bar{\mathbf{x}} = \frac{1}{B}\sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
11:     $\Sigma_{\mathbf{x}} = \frac{1}{B-1}\sum_{i=1}^{B}(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
    Calculate confidence intervals
12:     $\mathbf{x}_{(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(84\%)}$
13:     $\mathbf{x}_{(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

2-2-6 Residual resampling

Residual resampling is a semi-parametric bootstrapping method which involves resampling of model errors; it relies on the assumption of exchangeability of the error terms.
First the error values $r_n = y_n - \mathbf{a}_n\mathbf{x}$ are centered. From this set of centered errors samples are drawn and added to the estimated $\hat{y}_n = \mathbf{a}_n\mathbf{x}$ to make the new sampled output: $\tilde{y}_n^i = \mathbf{a}_n + r^i$. The bootstrapping procedure is written out in Algorithm 4 (Freedman, 1981)(Sartori, 2011).
It has been shown that this residual resampling fails to give a consistent estimation of the distribution in the case components of the regression vector are equal to zero. This was initially shown by (Knight and Fu, 2000) and more extensively by (Chatterjee A., 2010).
With bootstrapping the distribution $T = \sqrt{N}(\mathbf{x} - \mathbf{x}^{true})$ is approximated by using the generated samples: $\tilde{T} = \sqrt{N}(\tilde{\mathbf{x}} - \mathbf{x})$. By increasing the bootstrap size B one would expect the approximated distribution $\tilde{T}$ to converge to the real distribution T. However, this is not the case, the distribution converges to a random measure. The explanation of this effect is based on incorrect sign estimation. For regression coefficients that are non-zero, the estimations of the sign are correct with a high probability. For the zero coefficients both positive and negative values are found, while the correct sign value is zero. Residual resampling fails to reproduce the sign of the zero components of $\mathbf{x}$ with enough accuracy, thereby leading to inconsistent estimation.

Algorithm 4 Residual resampling

```
    **Calculate LASSO solution**
1:    $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    **Process the errors**
2:    $r_n = y_n - \mathbf{a}_n\mathbf{x}$    for $n = 1 \dots N$
3:    $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:    $r_n = r_n - \bar{r}$    for $n = 1 \dots N$
    **Iterate to generate set of solutions**
5:  for $i = 1 \to B$ do
6:      for $n = 1 \to N$ do
7:          $index \leftarrow u_{dist}(N)$
8:          $\tilde{r}_n^i = r_{index}$
9:      end for
10:     $\tilde{\mathbf{y}}^i = A\mathbf{x} + \tilde{\mathbf{r}}^i$
11:     $\tilde{\mathbf{x}}^i = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
12: end for
    **Calculate the covariance**
13:     $\bar{\mathbf{x}} = \frac{1}{B}\sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
14:     $\Sigma_{\mathbf{x}} = \frac{1}{B-1}\sum_{i=1}^{B} (\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
    **Calculate confidence intervals**
15:     $\mathbf{x}_{(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(84\%)}$
16:     $\mathbf{x}_{(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

2-2-7 Parametric bootstrap

Parametric bootstrapping assumes knowledge of the distribution of the error terms, for example a Gaussian distribution. This can be beneficial in the case of few data points, where more diverse error terms can be generated. The method for the LASSO assumes the model of $\mathbf{y} = A\mathbf{x} + \boldsymbol{\varepsilon}$ and estimates the variance σ_ε^2. Using this estimated variance it resamples error distributions from a Gaussian distribution. Thus, the method is the same as the residual resampling, only now the terms $\tilde{r}_n^i$ are drawn from the known distribution. This method is written out in Appendix B-1.

2-2-8 Fast residual resampling

As seen in the approximation method by Osborne, the LASSO can be estimated by Ridge regression. This can be used to speed the bootstrapping as is proposed by (Sartori, 2011). It is a variation of the well-known one-step bootstrap, developed for more complex models by (Moulton and Zeger, 1991). Using the definition of the LASSO as a Ridge regression problem in (2-1) the solution can be written as $(A^TA + \lambda/2W^-)^{-1}A^T\mathbf{y} = \Psi\mathbf{y}$. Here W^- is the pseudo inverse of the matrix W which has the absolute values of the regression vector $\mathbf{x}$ on its diagonal. In the bootstrap method approximations can be obtained using this matrix Ψ and the generated samples $\tilde{\mathbf{y}}$. This way the optimization of the LASSO does not have to be rerun for each sample, only a multiplication has to take place resulting in a significant reduction of computational expense. This fast residual resampling method is given in Algorithm 17 found in Appendix B-2.

$$\begin{aligned} W^{-} &= \text{Moore-Penrose pseudoinverse of } \begin{bmatrix} |x_1| & 0 & .. & 0 \\ 0 & |x_2| & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & |x_p| \end{bmatrix} \\ \Psi &= \left(A^T A + \frac{\lambda}{2} W^{-}\right)^{-1} A^T \end{aligned} \tag{2-5}$$

2-2-9 Proposed fast residual resampling

The fast residual resampling is a good method for reducing the computational burden. However, the method has the same problem as the approximation formula of Tibshirani, it breaks down if elements of the regression vector are zero. The use of Moore-Penrose pseudoinverse prevents this problem partially. However it would be more logical to use the approximation method proposed by Osborne, which does not have the break-down. Therefore I propose a new fast residual resampling method, the same as in the algorithm of fast residual resampling, only now using Ψ as defined in (2-6). The method is written out in Appendix B-3.

$$\Psi = \left(A^T\left(I_N + \frac{(\mathbf{y}-A\mathbf{x})(\mathbf{y}-A\mathbf{x})^T}{||\mathbf{x}||_1 ||A^T(\mathbf{y}-A\mathbf{x})||_\infty}\right)A\right)^{-1} A^T \tag{2-6}$$

2-2-10 Double bootstrap

The double bootstrap is a method for obtaining higher accuracy in calculating confidence intervals. The confidence intervals are often too small due to finite sample behavior. First a bootstrap is run and by estimating the coverage error of a confidence interval of these first stage estimates they are adjusted to increase the accuracy. The double bootstrap is described by (McCullough and Vinod, 1998) and the method is given in Algorithm 5. In the double bootstrap the transformation of the samples to roots $\tilde{q}_p$ as in (2-7) is applied. This transformation reduces the dependency of the distribution on x_p. For example if x_p would sampled from $\mathcal{N}(3, 0.5)$, the roots would form a standard normal distribution, independent from x_p.

$$\tilde{q}_p^b = \frac{\tilde{x}_p^b - x_p}{\sigma_{x_p}} \tag{2-7}$$

Then a second stage is applied to each root $\tilde{q}_p^b$ to generate the roots $\check{q}_p^{b,d}$. These second stage roots are created by sampling D times from the error $\tilde{\mathbf{r}}^b$ to form the samples $\check{\mathbf{y}}^{b,d} = \tilde{\mathbf{y}}^b + \check{\mathbf{r}}^{b,d}$. The $\check{\mathbf{x}}^{b,d}$ associated to these samples are used to find the second stage roots $\check{q}_p^{b,d}$. This gives an estimation of the error of $\tilde{q}_p^b$; if the distribution of $\check{q}_p^{b,d}$ would be the same as $\tilde{q}_p^b$, the distribution $\tilde{q}_p^b$ is completely accurate. Thus, the confidence interval is improved by using $\tilde{g}_p^b = \#\{\check{q}_p^{b,d} \leq \tilde{q}_p^b\}/D$. This index $\tilde{g}_p^b$ counts the number of times $\check{q}_p^{b,d}$ is smaller or equal to q_p^b and thus has a value between 0 and 1. If the distribution

of $\tilde{q}_p$ is completely accurate he distribution of $\breve{g}_p$ would be an uniform distribution on $[0, 1]$. The correction is applied by using this set of $\tilde{g}_p$ to determine the percentile level of $\tilde{\mathbf{x}}$. For example if the 5% percentile is desired, first the 5% percentile of $\breve{g}_p$ will be determined. If this is for example 0.03, then the 3% percentile of $\tilde{\mathbf{x}}$ will be used for the confidence interval of $\mathbf{x}$.
A more intuitive explanation is given by (Steigerwald, 2010). The idea of double bootstrap is to find the correction factor $\tilde{h}_{p(84\%)}$ to yield the result of (2-8), by finding the factor that satisfies (2-9). This is done using the roots as in (2-10).

$$Pr\left(x_p^{true} > 2x_p - \{\tilde{x}_p^1 \dots \tilde{x}_p^B\}_{\tilde{h}_{p(84\%)}}\right) = 84\% \tag{2-8}$$

$$Pr\left(x_p > 2\tilde{x}_p^b - \{\breve{x}_p^{b,1} \dots \breve{x}_p^{b,D}\}_{\tilde{h}_{p(84\%)}}\right) = 84\% \tag{2-9}$$

$$\begin{aligned} &Pr\left(\frac{\tilde{x}_p^b - x_p}{\sigma_{x_p}} < \frac{\{\breve{x}_p^{b,1} \dots \breve{x}_p^{b,D}\}_{\tilde{h}_{p(84\%)}} - \tilde{x}_p^b}{\tilde{\sigma}_{x_p}^b}\right) = 84\% \\ &Pr\left(\tilde{q}_p^b < \{\breve{q}_p^{b,1} \dots \breve{q}_p^{b,D}\}_{\tilde{h}_{p(84\%)}}\right) = 84\% \\ &\tilde{q}_p^b = \{\breve{q}_p^{b,1} \dots \breve{q}_p^{b,D}\}_{\tilde{h}_{p(84\%)}} \quad \rightarrow \quad \tilde{h}_{p(84\%)} = \frac{\#\{\breve{q}_p^{b,d} \leq \tilde{q}_p^b\}}{D} \\ &Pr\left(\tilde{h}_{p(84\%)} > \frac{\#\{\breve{q}_p^{b,d} \leq \tilde{q}_p^b\}}{D}\right) = 84\% \\ &Pr\left(\tilde{g}_p^b < \tilde{h}_{p(84\%)}\right) = 84\% \quad \rightarrow \quad \tilde{h}_{p(84\%)} = \{\tilde{g}_p^1 \dots \tilde{g}_p^B\}_{(84\%)} \end{aligned} \tag{2-10}$$

2-2-11 Fast double bootstrap

Proposed by (Sartori, 2011) to speed up the double bootstrap is to use an approximation of the LASSO solution, just as the adjustment of the residual resampling to fast residual resampling. On line 11 and 24 of Algorithm 5 finding the LASSO solution is replaced by $\tilde{\mathbf{x}}^i = \Psi\tilde{\mathbf{y}}^i$ and $\breve{\mathbf{x}}^{b,d} = \Psi\breve{\mathbf{y}}^{b,d}$ respectively, with Ψ as defined in (2-5). Note that in this case the Ψ is only calculated for $\mathbf{x}$ and not recalculated in the second stage based on $\tilde{x}^b$. This means that the approximations in the second stage for $\tilde{\mathbf{x}}^b$ use the Ψ matrix based on $\mathbf{x}$. The method is written out in Algorithm 19 found in Appendix B-4.

2-2-12 Proposed fast double bootstrap

Just as with the fast residual resampling using the Tibshirani approximation for the calculation of the LASSO is suboptimal. Using Ψ as in (2-6) will result in better estimation when elements of $\mathbf{x}$ are equal to zero. A second difference is that recalculating Ψ for each sample of the first stage would increase the accuracy of the second stage at the cost of more computational complexity. The method is written out in Algorithm 20 found in Appendix B-5.

Algorithm 5 Double bootstrap

```
    **Calculate LASSO solution**
 1:   $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    **Process the errors**
 2:   $r_n = y_n - \mathbf{a}_n\mathbf{x}$      for  $n = 1 \dots N$
 3:   $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
 4:   $r_n = r_n - \bar{r}$      for  $n = 1 \dots N$
    **First stage bootstrap: iterate to generate set of solutions**
 5: **for** $b = 1 \to B$ **do**
 6:     **for** $n = 1 \to N$ **do**
 7:         $index \leftarrow u_{dist}(N)$
 8:         $\tilde{r}_n^b = r_{index}$
 9:     **end for**
10:     $\tilde{\mathbf{y}}^b = A\mathbf{x} + \tilde{\mathbf{r}}^b$
11:     $\tilde{\mathbf{x}}^b = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^b\|_2^2 + \lambda\|\mathbf{x}\|_1$
12: **end for**
    **Calculate covariance and roots of the first stage**
13:   $\bar{\mathbf{x}} = \frac{1}{B}\sum_{b=1}^{B} \tilde{\mathbf{x}}^b$
14:   $\Sigma_{\mathbf{x}} = \frac{1}{B-1}\sum_{b=1}^{B}(\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})^T$
15:   $\sigma_{x_p} = \frac{1}{B-1}\sum_{b=1}^{B}(\tilde{x}_p^b - \bar{x}_p)(\tilde{x}_p^b - \bar{x}_p)^T$      for  $p = 1 \dots P$
16:   $\tilde{q}_p^b = (\tilde{x}_p^b - x_p)/\sigma_{x_p}$      for  $b = 1 \dots B$  and  $p = 1 \dots P$
    **Calculate roots in the second stage for each sample of the first stage**
17: **for** $b = 1 \to B$ **do**
18:     **for** $d = 1 \to D$ **do**
19:         **for** $n = 1 \to N$ **do**
20:             $index \leftarrow u_{dist}(N)$
21:             $\breve{r}_n^{b,d} = \tilde{r}_{index}^b$
22:         **end for**
23:         $\breve{\mathbf{y}}^{b,d} = \tilde{\mathbf{y}}^b + \breve{\mathbf{r}}^{b,d}$
24:         $\breve{\mathbf{x}}^{b,d} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \breve{\mathbf{y}}^{b,d}\|_2^2 + \lambda\|\mathbf{x}\|_1$
25:     **end for**
26:     $\overline{\tilde{\mathbf{x}}^b} = \frac{1}{D}\sum_{d=1}^{D} \breve{\mathbf{x}}^{b,d}$
27:     $\tilde{\sigma}_{x_p^b} = \frac{1}{D-1}\sum_{D=1}^{D}(\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})(\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})^T$      for  $p = 1 \dots P$
28:     $\breve{q}_p^{b,d} = (\breve{x}_p^{b,d} - \tilde{x}_p^b)/\tilde{\sigma}_{x_p^b}$      for  $d = 1 \dots D$  and  $p = 1 \dots P$
29:     $\tilde{g}_p^b = \#\{\breve{q}_p^{b,d} \leq \tilde{q}_p^b\}/D$      for  $p = 1 \dots P$
30: **end for**
    **Calculate confidence intervals**
31: **for** $p = 1 \to P$ **do**
32:         $\tilde{h}_{p(84\%)} = \{\tilde{g}_p^1 \cdots \tilde{g}_p^B\}_{(84\%)}$
33:         $x_{p(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{p(84\%)}}$
34:         $\tilde{h}_{p(16\%)} = \{\tilde{g}_p^1 \cdots \tilde{g}_p^B\}_{(16\%)}$
35:         $x_{p(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{p(16\%)}}$
36: **end for**
```

2-2-13 Modified bootstrapping

Residual resampling does not provide a distribution estimation. This problem is solved by (Chatterjeea and Lahiria, 2011), who propose a threshold bootstrapping method to correctly get the distributions of the LASSO estimators. This bootstrap method is derived from the residual resampling procedure and is written out in Algorithm 6. The idea of the procedure is to force components of $\mathbf{x}$ to be exactly zero when they are close to zero by introducing extra shrinkage. This is seen in the first step of the algorithm where the elements of $\mathbf{x}$ are set to zero if they are smaller then c_N. This way the zero coefficients are estimated more often with the correct sign value of zero. (Chatterjeea and Lahiria, 2011) show that this leads to a consistent approximation also in the case that elements of the regression vector are zero.
The intuitive explanation of the effect of this extra shrinkage step is seen in Figure 2-1. By applying a threshold the bootstrapping of elements of $\mathbf{x}$ around zero is done after they are set to zero. This leads to a reduction of over-estimating the covariance when using an estimate $x_p \neq 0$.

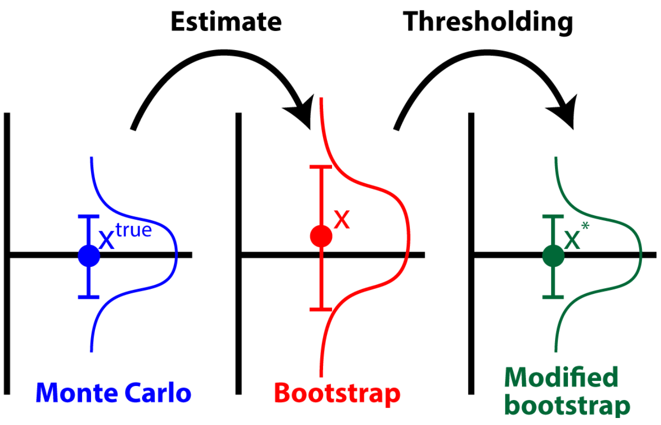


Figure 2-1: Illustration of the working principle of the modified bootstrap. Applying the extra shrinkage assures that the bootstrapping occurs at $x = 0$ for the coefficients estimated close to zero.

Finding the threshold parameter of c_N is critical in this framework, the goal is to correctly estimate the distribution $T = \sqrt{N}(\mathbf{x} - \mathbf{x}^{true})$ with $\tilde{T} = \sqrt{N}(\tilde{\mathbf{x}} - \mathbf{x}^*)$. In the paper of (Chatterjeea and Lahiria, 2011) a complex and computationally very expensive jackknife-after-bootstrap is suggested. In the simulations of this chapter this became too much of a computational burden, therefore their other and simpler suggestion is used, by minimizing the norm of $\tilde{T}$ directly as in (2-11). This leads to an appropriate choice of the thresholding factor.
The working principle of this tuning can also be explained by Figure 2-1. By minimizing the distribution, the overestimation is reduced when the elements are incorrectly estimated with $x_p \neq 0$. If too much shrinkage is applied element of the support could be set to zero. This would then increase the found distribution.

$$c_n = \underset{c_N}{\operatorname{argmin}} \quad \sum_{i=1}^{B} \sqrt{N} \|\tilde{\mathbf{x}}^i(c_N) - \mathbf{x}^*(c_N)\|_2^2 \tag{2-11}$$

Algorithm 6 Modified Bootstrap
$\mathbf{1}(\dots)$ refers to the indicator function, resulting in 1 if the statement is true and 0 if it is untrue

```
    Calculate LASSO solution
 1:   $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    Threshold LASSO estimate
 2:     $x_p^* = x_p \mathbf{1}(|x_p| \geq c_N) \qquad p = 1 \dots P$
    Prepare samples for bootstrap
 3:     $r_n = y_n - \mathbf{a}_n \mathbf{x}^* \qquad \text{for} \;\; n = 1 \dots N$
 4:     $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
 5:     $r_n = r_n - \bar{r} \qquad \text{for} \;\; n = 1 \dots N$
    Iterate to generate set of solutions
 6: for $i = 1 \to B$ do
 7:     for $n = 1 \to N$ do
 8:         $index \leftarrow u_{dist}(N)$
 9:         $\tilde{r}_n^i = r_{index}$
10:         $\tilde{y}_n^i = \mathbf{a}_n \mathbf{x}^* + \tilde{r}_n^i$
11:     end for
12:     $\tilde{\mathbf{x}}^i = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
13: end for
    Calculate the covariance
14:     $\bar{\mathbf{x}} = \frac{1}{B} \sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
15:     $\Sigma_{\mathbf{x}} = \frac{1}{B-1} \sum_{i=1}^{B} (\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
    Calculate confidence intervals
16:     $\mathbf{x}_{(16\%)} = \mathbf{x} + \mathbf{x}^* - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(84\%)}$
17:     $\mathbf{x}_{(84\%)} = \mathbf{x} + \mathbf{x}^* - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

2-2-14 Probability density function

A finite sample probability distribution function of $\mathbf{x}$ is derived by (Pötscher and Leeb, 2009). This would make all the approximations unnecessarily if it were not for the many restrictive assumptions: A is filled with ones and $\sigma_\varepsilon^2 = 1$. Furthermore, the true value of the regressor $\mathbf{x}^{true}$ must also be known. This makes the formula not a method for finding the covariance or confidence intervals. It is however a way to see how the Monte-Carlo method reflects the theoretical distribution, which will be done in Section 2-5.
The assumptions for deriving the theoretical distribution function are strong: A consists of only ones and $P = 1$, $\mathbf{x}$ is scalar. The LASSO problem then simplifies to (2-12). The OLS estimate in this scenario is simply the mean of the output vector, $\bar{\mathbf{y}}$, and thus its distribution function is easily derived: $\mathcal{N}(x^{true}, 1/\sqrt{N})$. The LASSO estimate is given in (2-13) as a soft-thresholding function, of which the proof is given in Appendix A-4. Here the soft-thresholding function is defined as $(\dots)_+ = max(\dots, 0)$.

$$J = \sum_{n=1}^{N} (x - y_n)^2 + \lambda|x| \tag{2-12}$$

$$x = \text{sign}(\bar{\mathbf{y}})\left(|\bar{\mathbf{y}}| - \frac{\lambda}{2N}\right)_+ \tag{2-13}$$

The derived formula under these assumptions of the distribution function $p(x)$ is given in (2-14). The function $\phi(x)$ gives the value of the probability density function of the standard normal distribution at point x and $\Phi(x)$ gives the value of the cumulative density function of the standard normal distribution function. $\mathbf{1}(\dots)$ gives a 1 as output if the function statement is true and 0 otherwise. δ_0 is the Dirac impulse function, which represents the discrete point of $x_p = 0$ and is called a singular normal distribution. The multiplier of this event is determined by the probability of $x = 0$: $Pr(x = 0)$.

$$\begin{aligned} p(x) = &\underbrace{\sqrt{N}\left\{\Phi\left(\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) - \Phi\left(-\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right)\right\}}_{Pr(x=0)}\delta_0 + \\ &\sqrt{N}\phi\left(\sqrt{N}(x - x^{true}) - \frac{\lambda}{2\sqrt{N}}\right)\mathbf{1}(x<0) + \\ &\sqrt{N}\phi\left(\sqrt{N}(x - x^{true}) + \frac{\lambda}{2\sqrt{N}}\right)\mathbf{1}(x>0) \end{aligned} \tag{2-14}$$

The LASSO distribution is constructed of three segments: $x < 0$, $x = 0$, $x > 0$. For the $x = 0$ segment the probability must be calculated that x is estimated zero, which is derived in (2-15).

$$\begin{aligned} Pr(x = 0) &= Pr\left(|\bar{\mathbf{y}}| \leq \frac{\lambda}{2N}\right) = Pr\left(|\mathcal{N}(x^{true}, 1/\sqrt{N})| \leq \frac{\lambda}{2N}\right) \\ &= Pr\left(\mathcal{N}(x^{true}, 1/N) \leq \frac{\lambda}{2N}\right) - Pr\left(\mathcal{N}(x^{true}, 1/N) \leq -\frac{\lambda}{2N}\right) \\ &= \sqrt{N}Pr\left(\mathcal{N}(0,1) \leq \frac{\lambda}{2\sqrt{N}} - \mathbf{x}^{true}\sqrt{N}\right) - \sqrt{N}Pr\left(\mathcal{N}(0,1) \leq -\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) \\ &= \sqrt{N}\Phi\left(\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) - \sqrt{N}\Phi\left(-\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) \end{aligned} \tag{2-15}$$

In a similar fashion the probability that $Pr(x < c)$ conditional to $x < 0$ and $x > 0$ is derived in (2-16) and (2-17) respectively. The probability density function in (2-14) is derived as the sum of these components.

$$\begin{aligned} Pr(x < c | x < 0) &= Pr\left(\bar{\mathbf{y}} + \frac{\lambda}{2N} < c\right) \\ &= Pr\left(\mathcal{N}(x^{true}, 1/\sqrt{N}) < c - \frac{\lambda}{2N}\right) \\ &= Pr\left(\mathcal{N}(0, 1/\sqrt{N}) < c - x^{true} - \frac{\lambda}{2N}\right) \\ &= \sqrt{N}Pr\left(\mathcal{N}(0,1) < \sqrt{N}(c - x^{true}) - \frac{\lambda}{2\sqrt{N}}\right) \\ &= \sqrt{N}\phi\left(\sqrt{N}(c - x^{true}) - \frac{\lambda}{2\sqrt{N}}\right) \end{aligned} \tag{2-16}$$

$$\begin{aligned}
Pr(x < c|x > 0) &= Pr\Big(\bar{\mathbf{y}} - \frac{\lambda}{2N} < c\Big) \\
&= Pr\Big(\mathcal{N}(x^{true}, 1/\sqrt{N}) < c + \frac{\lambda}{2N}\Big) \\
&= Pr\Big(\mathcal{N}(0, 1/\sqrt{N}) < c - x^{true} + \frac{\lambda}{2N}\Big) \\
&= \sqrt{N}Pr\Big(\mathcal{N}(0,1) < \sqrt{N}(c - x^{true}) + \frac{\lambda}{2\sqrt{N}}\Big) \\
&= \sqrt{N}\phi\Big(\sqrt{N}(c - x^{true}) + \frac{\lambda}{2\sqrt{N}}\Big)
\end{aligned} \tag{2-17}$$

2-3 Applying the Unscented Transform to the LASSO

The Unscented Transform is a method to calculate the covariance of a random (vector) variable which undergoes a nonlinear transformation. For example for the nonlinear transformation $f(\mathbf{y})$ of (2-18) the mean and covariance of $\mathbf{x}$ are sought, given a set of vectors $\mathbf{y}$; the vector $\mathbf{y}$ has dimension N. This is illustrated in Figure 2-2, where one method would be to apply the transformation to each $\mathbf{y}$ of the set. This is the most accurate method, but computationally expensive. The UT calculates $\bar{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ using the mean of the set $\bar{\mathbf{y}}$ and its covariance $\Sigma_{\mathbf{y}}$ to calculate specific points, called sigma points. By applying the nonlinear transformation only on this small set of sigma points $\tilde{\mathbf{y}}$ the transformed set of $\tilde{\mathbf{x}}$ is obtained from which $\bar{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ can be calculated. This way only few transformations have to be done, resulting in a computational reduction. The UT is often used in the Kalman filter, where it can achieve higher accuracies compared to linearizing the nonlinear transformation $f(\mathbf{y})$. The idea presented here is to apply the UT to finding the covariance of the LASSO, resulting in high accuracy and low computational costs (Julier and Uhlmann, 1996).

$$\mathbf{x} = f(\mathbf{y}) \tag{2-18}$$

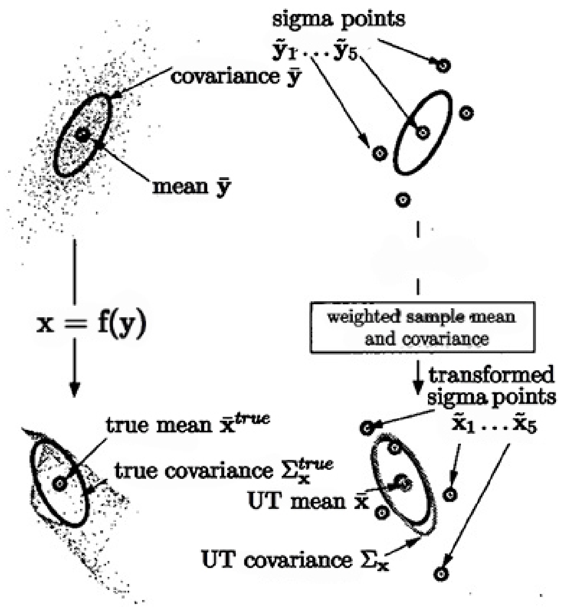

sigma points
$\tilde{\mathbf{y}}_1 \ldots \tilde{\mathbf{y}}_5$
covariance $\bar{\mathbf{y}}$
mean $\bar{\mathbf{y}}$
$\mathbf{x} = \mathbf{f}(\mathbf{y})$
weighted sample mean and covariance
transformed sigma points
$\tilde{\mathbf{x}}_1 \ldots \tilde{\mathbf{x}}_5$
true mean $\bar{\mathbf{x}}^{true}$
true covariance $\Sigma_{\mathbf{x}}^{true}$
UT mean $\bar{\mathbf{x}}$
UT covariance $\Sigma_{\mathbf{x}}$

Figure 2-2: Illustration of the principle of the unscented transform. On the left the true mean and covariance are displayed and on the right the unscented transform approach is illustrated to get the mean and covariance of $\mathbf{x}$. Figure borrowed from (Van der Merwe and Wan, 2001), adjusted to fit the notations of this report.

First the UT algorithm for the general scenario is reviewed, then the proposed algorithm derived for the LASSO is discussed. Finally a method to increase the accuracy by using more sigma points is discussed.

2-3-1 General Algorithm

Algorithm 7 describes the UT. It starts out by calculating the mean and covariance of $\mathbf{y}$ and uses these to make the sigma points $\tilde{\mathbf{y}}^j$. The transformed sigma points $\tilde{\mathbf{x}}^j$ are then used to derive the estimated mean and covariance of $\mathbf{x}$. The weights w_j are chosen optimally using $N + \kappa = 3$ for the case of Gaussian distribution of $\mathbf{y}$. This choice of weights is based on the fact that this way the sigma points not only yield a correct covariance of $\mathbf{y}$, but also the fourth order moment is correctly estimated (Julier and Uhlmann, 1996).
(Julier and Uhlmann, 1996) indicate that in some cases $w_0 = 0$ should be used for estimating the covariance $\Sigma_\mathbf{x}$ on line 7. This stems from the possibility that the predicted covariance will be non-positive semi-definite in the case that κ and thus w_0 is negative. w_0 becomes negative at large values of N. This is incorporated in the algorithm by checking the positive semi-definiteness of the estimated $\Sigma_\mathbf{x}$ and recalculate it using $w_0 = 0$ if it is non-positive semi-definite.

Algorithm 7 General Unscented Transform
The Square Root S of matrix P is defined as $SS^T = P$
Subscript j on line 3 and 4 refers to the j-th row of this matrix

```
 1: **Calculate** $\bar{\mathbf{y}}$ **&** $\Sigma_\mathbf{y}$
    **Calculate sigma point set and weights:**
 2:     $\tilde{\mathbf{y}}^0 = \bar{\mathbf{y}}$                                   $w_0 = \kappa/(N+\kappa)$
 3:     $\tilde{\mathbf{y}}^j = \bar{\mathbf{y}} + \sqrt{(N+\kappa)}\left(\sqrt{\Sigma_\mathbf{y}}\right)_j$         $w_j = 1/2(N+\kappa)$      for $j = 1 \dots N$
 4:     $\tilde{\mathbf{y}}^{j+N} = \bar{\mathbf{y}} - \sqrt{(N+\kappa)}\left(\sqrt{\Sigma_\mathbf{y}}\right)_j$     $w_{j+N} = 1/2(N+\kappa)$  for $j = 1 \dots N$
    **Transform the sigma points**
 5:     $\tilde{\mathbf{x}}^j = f(\tilde{\mathbf{y}}^j)$      for $j = 0 \dots 2N$
    **Calculate covariance**
 6:     $\bar{\mathbf{x}} = \sum_{j=0}^{2N} w_j \tilde{\mathbf{x}}^j$
 7:     $\Sigma_\mathbf{x} = \sum_{j=0}^{2N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
 8: **if** $\Sigma_\mathbf{x} \neq$ positive semi-definite **then**
 9:     $\Sigma_\mathbf{x} = \sum_{j=1}^{2N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
10: **end if**
```

2-3-2 Algorithms for the LASSO

The LASSO finds a sparse solution of $\mathbf{x}$ using $(A, \mathbf{y})$. To find the covariance of the LASSO we consider the A-matrix as given and derive sigma points for $\mathbf{y}$: $\tilde{\mathbf{y}}^j$. These sigma points will be generated using the covariance matrix $\Sigma_\mathbf{y}$. Each data point of $\mathbf{y}$ is the solution of $y_n = \mathbf{a}_n\mathbf{x} + \varepsilon$. Thus, the variance of y_n is that of the ε: σ_ε^2. So, for each of the points of y_n in $\mathbf{y}$ the same variance is present and no covariance is present. This leads to the form of $\Sigma_\mathbf{y} = I_N\sigma_\varepsilon^2$, with size $N \times N$. This framework will be referred to as the Variance UT and is written out in Algorithm 8. It uses $2N + 1$ sigma-points; each data point n is perturbed in positive and negative direction.
For large data sets this could lead to calculating many sigma-points. A possible way to use less sigma points is to combine the perturbations of the output. For example

the whole output vector could be perturbed at once: $\tilde{\mathbf{y}}^i = \mathbf{y} + \mathbf{1}\sigma_\varepsilon$. Here $\mathbf{1}$ is a vector of length N with all 1s as entries. This would be equivalent to using $\Sigma_{\mathbf{y}} = \mathbf{1}\sigma_\varepsilon^2$. This method is referred to as the Scalar UT. It is written out in Algorithm 9 and only uses three sigma points. In between variants can also be thought of, for example using $N+1$ sigma points as defined in (2-19). These variants will be referred to as reduced Variance UT, for example the 50% Variance UT.

$$\Sigma_{\mathbf{y}}^{\text{Scalar}} = \underbrace{\left.\begin{bmatrix} \sigma_\varepsilon^2 \\ \vdots \\ \sigma_\varepsilon^2 \end{bmatrix}\right\}N}_{1} \quad \Sigma_{\mathbf{y}}^{\text{Variance}} = \underbrace{\left.\begin{bmatrix} \sigma_\varepsilon^2 & 0 & \dots & 0 & 0 \\ 0 & \sigma_\varepsilon^2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_\varepsilon^2 & 0 \\ 0 & 0 & \dots & 0 & \sigma_\varepsilon^2 \end{bmatrix}\right\}N}_{N} \quad \Sigma_{\mathbf{y}}^{50\%} = \underbrace{\left.\begin{bmatrix} \sigma_\varepsilon^2 & \dots & 0 \\ \sigma_\varepsilon^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_\varepsilon^2 \\ 0 & \dots & \sigma_\varepsilon^2 \end{bmatrix}\right\}N}_{N/2} \tag{2-19}$$

Algorithm 8 Variance UT

```
    **Calculate LASSO solution**
1:      $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    **Calculate $\Sigma_{\mathbf{y}}$**
2:      $r_n = y_n - \mathbf{a}_n\mathbf{x}$    for $n = 1 \dots N$
3:      $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:      $\sigma_\varepsilon^2 = 1/(N-1) \sum_{n=1}^{N} (r_n - \bar{r})^2$
5:      $\Sigma_{\mathbf{y}} = I_N \sigma_\varepsilon^2$
    **Calculate sigma point set and weights**
6:      $\tilde{\mathbf{y}}^0 = \mathbf{y}$    $w_0 = \kappa/(1+\kappa)$
7:      $\tilde{\mathbf{y}}^j = \mathbf{y} + \sqrt{N+\kappa}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_j = 1/2(N+\kappa)$    for $j = 1 \dots N$
8:      $\tilde{\mathbf{y}}^{j+N} = \mathbf{y} - \sqrt{N+\kappa}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+N} = 1/2(N+\kappa)$    for $j = 1 \dots N$
    **Transform the sigma points**
9:      $\tilde{\mathbf{x}}^j = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^j\|_2^2 + \lambda\|\mathbf{x}\|_1$    for $j = 0 \dots 2N$
    **Calculate covariance**
10:     $\bar{\mathbf{x}} = \sum_{j=0}^{2N} w_j \tilde{\mathbf{x}}^j$
11:     $\Sigma_{\mathbf{x}} = \sum_{j=0}^{2N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
12: **if** $\Sigma_{\mathbf{x}} \neq$ positive semi-definite **then**
13:     $\Sigma_{\mathbf{x}} = \sum_{j=1}^{2N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
14: **end if**
```

It is highly probable that perturbing the whole output vector at once as in the Scalar UT leads to a biased predictor of $\Sigma_{\mathbf{x}}$. The generated $\tilde{\mathbf{y}}$ using this method is not that probable to occur. However, reduced Variance UT methods could be a balance between using many sigma points and obtaining a reasonable accuracy. If two or three data points are perturbed together a factor two or three less sigma points is used, while the generated $\tilde{\mathbf{y}}$ could still occur in practice.

Algorithm 9 Scalar UT

```
    Calculate LASSO solution
1:      $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    Calculate $\Sigma_{\mathbf{y}}$ :
2:      $r_n = y_n - \mathbf{a}_n\mathbf{x}$    for $n = 1 \dots N$
3:      $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:      $\sigma_\varepsilon^2 = 1/(N-1) \sum_{n=1}^{N} (r_n - \bar{r})^2$
    Calculate sigma point set and weights
5:      $\tilde{\mathbf{y}}^0 = \mathbf{y}$    $w_0 = \kappa/(1+\kappa)$
6:      $\tilde{\mathbf{y}}^1 = \bar{\mathbf{y}} + \mathbf{1}\sqrt{(1+\kappa)}\sigma_\varepsilon$    $w_1 = 1/(2+2\kappa)$
7:      $\tilde{\mathbf{y}}^2 = \bar{\mathbf{y}} - \mathbf{1}\sqrt{(1+\kappa)}\sigma_\varepsilon$    $w_2 = 1/(2+2\kappa)$
    Transform the sigma points
8:      $\tilde{\mathbf{x}}^j = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^j\|_2^2 + \lambda\|\mathbf{x}\|_1$    for $j = 0, 1, 2$
    Calculate covariance
9:      $\bar{\mathbf{x}} = \sum_{j=0}^{2} w_j \, \tilde{\mathbf{x}}^j$
10:     $\Sigma_{\mathbf{x}} = \sum_{j=0}^{2} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
11: if $\Sigma_{\mathbf{x}} \neq$ positive semi-definite then
12:     $\Sigma_{\mathbf{x}} = \sum_{j=1}^{2} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
13: end if
```

2-3-3 Higher order UT

To increase the accuracy extra sigma points can be introduced to match higher orders of the set of $\mathbf{y}$. In the scalar case (Tenne and Singh, 2003) provide the equations for taking the double amount of sigma points in (2-20) to correctly include up to the eight order moments. Note that in these equations M_o refers to the o-th moment of the distribution of y. These equations are derived by writing out the moments of the set of sigma points, using as a constraint $\kappa + P = 3$. In this scalar scenario two sigma points are generated using τ_1 and w_1 and two points using τ_2 and w_2. So the set of sigma points thus becomes $\{\bar{y}, \bar{y}+\tau_1, \bar{y}+\tau_2, \bar{y}-\tau_1, \bar{y}-\tau_2\}$

$$\begin{aligned} 1 =& w_0 + 2(w_1 + w_2) \\ M_2 =& 2 \cdot 3(w_1\tau_1^2 + w_2\tau_2^2) \\ M_4 =& 2 \cdot 3^2(w_1\tau_1^4 + w_2\tau_2^4) \\ M_6 =& 2 \cdot 3^3(w_1\tau_1^6 + w_2\tau_2^6) \\ M_8 =& 2 \cdot 3^4(w_1\tau_1^8 + w_2\tau_2^8) \end{aligned} \tag{2-20}$$

This principle scales to higher orders easily, for example three times the amount of points is written out in (2-21) to correctly match up to the twelfth moment with the sigma points. Tenne uses an analytical solver to find the values of the weights and sigma points. In our calculation this method becomes difficult to solve for a fourth fold or up.

$$\begin{aligned}
1 =& w_0 + 2(w_1 + w_2 + w_3) \\
M_2 =& 2 \cdot 3(w_1\tau_1^2 + w_2\tau_2^2 + w_3\tau_3^2) \\
M_4 =& 2 \cdot 3^2(w_1\tau_1^4 + w_2\tau_2^4 + w_3\tau_3^4) \\
M_6 =& 2 \cdot 3^3(w_1\tau_1^6 + w_2\tau_2^6 + w_3\tau_3^6) \\
M_8 =& 2 \cdot 3^4(w_1\tau_1^8 + w_2\tau_2^8 + w_3\tau_3^8) \\
M_{10} =& 2 \cdot 3^5(w_1\tau_1^{10} + w_2\tau_2^{10} + w_3\tau_3^{10}) \\
M_{12} =& 2 \cdot 3^6(w_1\tau_1^{12} + w_2\tau_2^{12} + w_3\tau_3^{12})
\end{aligned} \tag{2-21}$$

In the case of the LASSO ε is Gaussian noise. This means that the moments can be pre-calculated, which is done for $\sigma_\varepsilon^2 = 1$ in Table 2-1. The found weights and sigma points for the 2-fold and 3-fold are given in Table 2-2. Here 2-fold refers to two times the amount of sigma points, and 3-fold to three times the amount of sigma points. For the LASSO, the 2-fold Variance UT method is given in Algorithm 10. The 3-folds algorithm is given in Appendix B-6.

Table 2-1: Moments of $\mathcal{N}(0,1)$.

M_2	M_4	M_6	M_8	M_{10}	M_{12}
1	3	15	105	945	10395

Table 2-2: Calculated weights and sigma points for Table 2-1.

	w_0	w_1	τ_1^2	w_2	τ_2^2	w_3	τ_3^2
normal	0.6666	0.1666	1				
2-fold	0.5333	0.0112	2.7208	0.2221	0.6126		
3-fold	0.4571	$5.429 \cdot 10^{-4}$	4.6886	0.0308	1.8672	0.24012	0.4442

Algorithm 10 2-fold Variance UT

```
    Calculate LASSO solution
1:      $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    Calculate $\Sigma_{\mathbf{y}}$ :
2:      $r_n = y_n - \mathbf{a}_n\mathbf{x}$    for $n = 1 \dots N$
3:      $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:      $\sigma_\varepsilon^2 = 1/(N-1) \sum_{n=1}^{N} (r_n - \bar{r})^2$
5:      $\Sigma_{\mathbf{y}} = I_N \sigma_\varepsilon^2$
    Calculate sigma point set and weights
6:      $\tilde{\mathbf{y}}^0 = \mathbf{y}$    $w_0 = 1 - 2N(0.0112 + 0.2221)$
7:      $\tilde{\mathbf{y}}^j = \mathbf{y} + \sqrt{3 \cdot 2.7208}\,(\sqrt{\Sigma_{\mathbf{y}}})_j$    $w_j = 0.0112$    for $j = 1 \dots N$
8:      $\tilde{\mathbf{y}}^{j+N} = \mathbf{y} - \sqrt{3 \cdot 2.7208}\,(\sqrt{\Sigma_{\mathbf{y}}})_j$    $w_{j+N} = 0.0112$    for $j = 1 \dots N$
9:      $\tilde{\mathbf{y}}^{j+2N} = \mathbf{y} + \sqrt{3 \cdot 0.6126}\,(\sqrt{\Sigma_{\mathbf{y}}})_j$    $w_{j+2N} = 0.2221$    for $j = 1 \dots N$
10:     $\tilde{\mathbf{y}}^{j+3N} = \mathbf{y} - \sqrt{3 \cdot 0.6126}\,(\sqrt{\Sigma_{\mathbf{y}}})_j$    $w_{j+3N} = 0.2221$    for $j = 1 \dots N$
    Transform the sigma points
11:     $\tilde{\mathbf{x}}^j = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^j\|_2^2 + \lambda\|\mathbf{x}\|_1$    for $j = 0 \dots 4N$
    Calculate covariance
12:     $\bar{\mathbf{x}} = \sum_{j=0}^{4N} w_j\, \tilde{\mathbf{x}}^j$
13:     $\Sigma_{\mathbf{x}} = \sum_{j=0}^{4N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
14: if $\Sigma_{\mathbf{x}} \neq$ positive semi-definite then
15:     $\Sigma_{\mathbf{x}} = \sum_{j=1}^{4N} w_j (\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
16: end if
```

2-4 Choice of LASSO solver

Many solvers for finding the solution to the LASSO problem exist with varying solving speed and accuracy. When comparing the distributions of obtained numerical solutions, the choice of the LASSO solver is an important one. The solver should be accurate and fast; if the solver is inaccurate the comparison becomes invalid and if it is too slow few comparisons can be performed. Furthermore, it should not incorporate any preprocessing steps; the solvers should solve the given objective function and perform no additional steps. Four solvers will be discussed below and the motivation for the used solver will be given.

2-4-1 State-of-the-art solvers

State-of-the-art LASSO solvers are built towards increasing the speed by making use of distributed algorithms. Alternating Direction Method of Multipliers (ADMM) is one of these methods. The ADMM solvers make it possible to find the LASSO problem in a distributed way using an iterative algorithm which splits steps over multiple processing units. The speed gain is high only for convergence to mediocre accuracy (Boyd et al., 2011). The convergence rate to high accuracy is rather slow, making an ADMM solver not an applicable solver for this research.

2-4-2 Built-in Matlab solver

In recent versions of Matlab a solver for the LASSO is included, it is a very fast and accurate solver. The solver is optimized for daily use, so it includes some preprocessing steps that cannot be disabled. One of the main problems is that the solver automatically removes the means from the data, it removes the means of the matrix A per row and the output vector $\mathbf{y}$. Normally this is a very useful feature, only a true solution to the problem is desired not including removal of the mean. In many of the methods described in the previous sections the output vector is manipulated, i.e. different quantities of noise are added. The removal of the meaning feature would alter this manipulated output vector. This is not desired and therefore the built-in Matlab solver is not used.

2-4-3 Disciplined convex programming (CVX)

A commonly used toolbox for Matlab is the Disciplined convex programming toolbox. This toolbox allows the user to easily define of a convex objective function and solve the regression problem. The found accuracy of this algorithm is rather high, only it is quite slow. Finding a solution to a regression problem takes about 10 times longer compared to the built-in lasso solver from Matlab. Since millions of LASSOs have to be solved in the comparisons of this chapter, such a slow solver is a liability. Therefore this toolbox is not used.

2-4-4 Quadratic programming solver

A Matlab implementation of a LASSO solver using quadratic programming was found by (Schmidt, 2005). The `LassoConstrained` function is a fast implementation using Matlab's `quadprog` function and validated to give the same results as the built-in Matlab solver when data has a mean of zero. The LASSO problem is rewritten as a constrained quadratic programming problem as in (2-22) and then solved using the function `quadprog`.

$$\begin{aligned} \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{x}\|_1 \quad &= \quad \min_{\mathbf{x},\boldsymbol{\alpha}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \sum_{p=1}^{P} \alpha_p \\ &\quad\ \text{s.t.} \quad \alpha_p \geq x_p \qquad \text{for } x = 1 \dots P \\ &\qquad\qquad \alpha_p \geq -x_p \qquad \text{for } x = 1 \dots P \end{aligned} \tag{2-22}$$

2-5 Theoretical and practical covariance

In the literature the asymptotic behavior of the approximation methods was analyzed; it was concluded by (Knight and Fu, 2000) that using bootstrap methods gives in an inconsistent estimate. This means that in the case of infinite amount of samples it still has a bias. This analysis was based on the mathematical properties of the LASSO and it was shown that the bootstrap method did not converge to the distribution given based on mathematical properties. For a finite amount of samples the probability density function was recently derived as given in Section 2-2. A comparison has been done by (Kyung and Casella, 2010), who compared this finite sample probability density function with bootstrap estimates. He concluded that the bootstrap had major errors in the finite sample behavior. The question is however, if this 'error' is due to the bootstrap method or other numerical aspects. Therefore, here the probability density function will be compared with the Monte Carlo method given in Section 2-2. The Monte Carlo method is a valid method to give a distribution of $\mathbf{x}$, which would be obtained in practice; for a very large amount of data sets a numerical solution $\mathbf{x}$ is found. If no other numerical aspects are the cause of the error of the bootstrap methods as described by (Kyung and Casella, 2010), the probability density function and the results of the Monte Carlo method should be the same.
Two numerical simulations will be run and the results will be compared by looking at histograms and Quantile-Quantile plots (Q-Q plots). Finally a discussion is given.

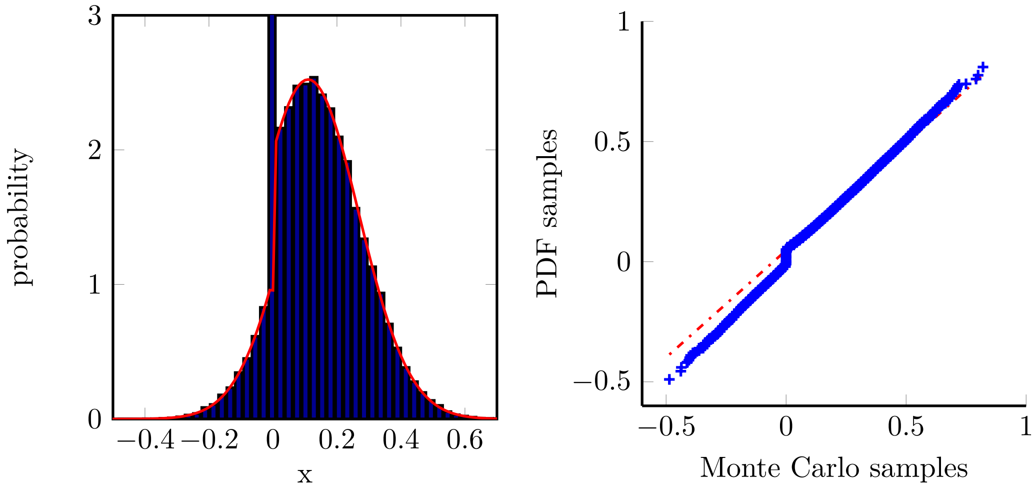
2-5-1 Simulations

To investigate how the mathematical properties of the LASSO are reflected in the Monte Carlo method a simulation is run using the parameters of Table 2-3. In this simulation a scalar scenario is assumed with a sample size of $N = 40$ and the regression coefficient $x^{true} = 0.16$. To meet the requirements of the probability density function A is a vector with only ones and the variance of the noise is set to $\sigma_\varepsilon^2 = 1$. These parameters are the same as in the scenario used by (Pötscher and Leeb, 2009), who derived the probability density function.

Table 2-3: Parameters for comparison of the Monte Carlo method with theoretical distribution

$y = x^{true} + \varepsilon$	$x^{true} = 0.16$	$A = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T$	$\varepsilon \sim \mathcal{N}(0,1)$
$\lambda = 4$	$N = 40$	$P = 1$	

The resulting histogram of the Monte Carlo simulation is shown in Figure 2-3a, along with the probability density function. They overlap nicely for the most part, except at the point $x = 0$. The calculated probability is about 1 at this point, but from the Monte Carlo simulation the probability at this point is about 3. To further asses the comparison in figure 2-3b a Q-Q plot is given. Here samples drawn from the probability density function are compared with the Monte Carlo samples. The resulting data points lie quite nice on the diagonal line from $x > 0$, but there is a clear jump at $x = 0$.
This indicates the distribution from the Monte Carlo simulations does not reflect the



(a) Histogram of the samples the Monte Carlo method and the probability density function in a solid red line.

(b) Q-Q plot of samples drawn from the distribution function on the y-axis versus samples generated by the Monte Carlo method on the x-axis

Figure 2-3: Results of the simulation of Table 2-3 comparing the probability density function (2-14) and the Monte Carlo method

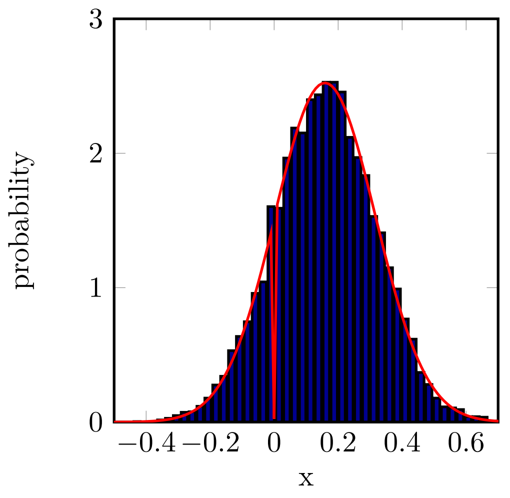
theoretical distribution of the LASSO.
When λ, is lower as in the parameters of Table 2-4, the Q-Q plot of Figure 2-3b shows almost a straight line. However a small deviation around $x = 0$ is still visible. In the histogram of Figure 2-4a a clear difference is visible around $x = 0$. Thus, for a lower λ the distribution from the Monte Carlo simulations reflects the theoretical distribution of the LASSO a lot better.

Table 2-4: Parameters for comparison of the Monte Carlo method with theoretical distribution

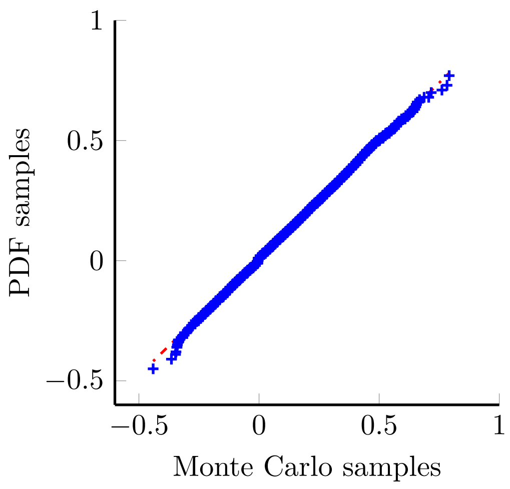
$y = x^{true} + \varepsilon$	$x^{true} = 0.16$	$A = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T$	$\varepsilon \sim \mathcal{N}(0,1)$
$\lambda = 0.2$	$N = 40$	$P = 1$	

2-5-2 Discussion

The numerical behavior of the LASSO differs from the mathematical behavior. In both scenarios the distribution of x from the Monte Carlo method does not correspond with the theoretically derived probability density function. This is not dependent on the choice of solver, even using the simple formula of (2-13) no radical change in the numerical behavior was found.
It is difficult to state where this difference stems from; it is at least clear that the way the LASSO is numerically solved is not of influence. The difference in behavior is around the point $x = 0$, in practice the solutions are more often set to zero than the theory describes. In the probability density function the possibility of x being exactly zero is



(a) Histogram of the samples the Monte Carlo Algorithm and the probability density function in a solid red line.



(b) QQ-plot of samples drawn from the distribution function on the y-axis versus samples generated by the Monte Carlo method on the x-axis

Figure 2-4: Results of the simulation of Table 2-4 comparing the probability density function (2-14) and the Monte Carlo method

calculated. This is not that high, in Figure 2-4a the probability is really low at $x = 0$. In (2-23) the limits around zero are calculated and compared to the probability of x being exactly zero. Here a clear difference is visible, for approaching $x = 0$ a probability of 3.35 is found, while at $x = 0$ the probability 0.96 is found. The probability of 3.35 matches much closer with the distribution found using the Monte Carlo method. Thus, the difference probably stems from the limiting behavior around the event $x = 0$.

$$\begin{aligned}
p(0) &= \sqrt{N}\left\{\Phi\left(\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) - \Phi\left(-\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right)\right\} = 0.96 \\
\lim_{x\to 0^+} p(x) &= \sqrt{N}\left\{\Phi\left(\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) - \Phi\left(-\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right)\right\} + \\
&\quad \sqrt{N}\phi\left(\frac{\lambda}{2\sqrt{N}}\right) = 3.35 \qquad (2\text{-}23) \\
\lim_{x\to 0^-} p(x) &= \sqrt{N}\left\{\Phi\left(\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right) - \Phi\left(-\frac{\lambda}{2\sqrt{N}} - x^{true}\sqrt{N}\right)\right\} + \\
&\quad \sqrt{N}\phi\left(-\frac{\lambda}{2\sqrt{N}}\right) = 3.35
\end{aligned}$$

Thus the errors found by (Kyung and Casella, 2010), on the basis of which they argued that the bootstrapping methods have large errors, is due to the difference between the limit behavior of the probability density function and the calculated probability at $x = 0$. It is a choice which distribution is valid: the mathematical or practical one. This research focuses on a practical measure of covariance and confidence intervals. That this practical behavior incorrectly displays the mathematical framework of the LASSO is another problem which will not be addressed here further. In practice the distribution given by the Monte Carlo simulation will be found when applying the LASSO, thus the focus is to find an approximation close to this distribution. This also backs up the argument that conclusions on the asymptotic behavior carry very little meaning about finding the covariance in practice.

2-6 Comparison of the approximations

As said, the accuracy of the approximation methods has not been found. All comparisons found where only for simple scenarios and included a limited set of the methods presented in Section 2-2. In this section numerical simulations will be used to compare the methods for finding the covariance and confidence intervals of the LASSO in different scenarios. First the comparison method is introduced with the parameters to describe the accuracy. Then a comparison is done in the scalar scenario, i.e. $\mathbf{x}$ having dimension 1. This scalar scenario simplifies the calculations and the results, so an easier comparison can be made. Then the accuracy is assessed in a nominal multidimensional case, where some of the regression coefficients are equal to zero. Three more exceptional scenarios are then reviewed: a situation with very high shrinkage, i.e. very large value of λ, an undetermined scenario with more regressors than data points $(P > N)$ and a scenario with few data points. Then the computational expense of the methods is compared with regard to the required computational time and the number of operations. Finally, a discussion is given.

2-6-1 Comparison method

The method for testing the different approximation formulae is written out in Algorithm 11. A Monte Carlo method is required to find the accuracy of the approximations. This is due to the fact that the accuracy of the approximations is dependent on the generated data set. In other words for example for data set 1 residual resampling is very precise and for data set 2 the analytical formula from Osborne is very precise. So, to say something on the accuracy in of the approximations with respect to the statistical properties of the scenario and not to one specific generated data set a Monte Carlo method is required. Thus, many data sets are generated and for each of these data set an $\mathbf{x}$ the approximation of $\boldsymbol{\sigma}_{\mathbf{x}}$ and of the confidence interval are calculated. By generating many solutions not only the average performance of $\boldsymbol{\sigma}_{\mathbf{x}}$ is obtained, but also its variance. If the method is on average very accurate, but with a large variance, i.e. the estimation of $\boldsymbol{\sigma}_{\mathbf{x}}$ varies heavily, a small drop in average accuracy with a lower variance may be preferable.
Note that for all the bootstrap variants $B = 1000$ is used unless specified otherwise and for the double bootstrap $B = 100, D = 100$ is used. More iterations on the double bootstrap became too numerical expensive.
The performance indicators that will be used are related to $\boldsymbol{\sigma}_{\mathbf{x}}$ and the 68% confidence interval. For the accuracy of $\boldsymbol{\sigma}_{\mathbf{x}}$ the bias, as defined in (2-24), is used as a performance indicator. The spread is measured with the standard deviation of $\boldsymbol{\sigma}_{\mathbf{x}}$. The confidence interval accuracy is assessed using the level error as defined by (McCullough and Vinod, 1998). The level error measures the deviation of the coverage. The coverage is the average probability of the true value $\mathbf{x}^{true}$ being inside the estimated confidence interval and is given in (2-25). If the average coverage is 68% the level error is zero and the estimation of the confidence interval is thus perfect.
In the multidimensional case $\boldsymbol{\sigma}_{\mathbf{x}}$,$\mathbf{x}^i_{(16\%)}$,$\mathbf{x}^i_{(84\%)}$ and $\mathbf{ce}^i$ are all vectors of dimension P. The accuracies in the algorithm are determined per coefficient. So $b(\sigma_{x_p})$ is the bias for regressor p and σ_{x_p} is the standard deviation of x_p and the p-th element in the vector $\boldsymbol{\sigma}_{\mathbf{x}}$.

Algorithm 11 Testing the accuracy of the 'approximation' method
The function approximation(...) refers to testing an approximation method for example Residual resampling

```
    **Iterate to generate set of approximations**
 1: for $i = 1 \to 250$ do
 2:     $A^i \leftarrow f_{generating}(\boldsymbol{\mu}^A, (\boldsymbol{\sigma}^A)^2)$
 3:     $\mathbf{y}^i \leftarrow A^i\mathbf{x} + \mathcal{N}(\mathbf{0}, I_N\sigma_\varepsilon^2)$
 4:     $\mathbf{x}^i = \text{argmin}_{\mathbf{x}} \|A^i\mathbf{x} - \mathbf{y}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
 5:     $\boldsymbol{\sigma}_{\mathbf{x}}^i \leftarrow \text{approximation}(A^i, \mathbf{y}^i, \mathbf{x}^i)$
 6:     $[\mathbf{x}_{(16\%)}^i \mathbf{x}_{(84\%)}^i] \leftarrow \text{approximation}(A^i, \mathbf{y}^i, \mathbf{x}^i)$
 7:     $\mathbf{co}^i = f_{coverage}(\mathbf{x}_{(16\%)}^i, \mathbf{x}_{(84\%)}^i, \mathbf{x}^{true})$
 8: end for
    **Find accuracies per coefficient**
 9: for $p = 1 \to P$ do
10:     $\bar{\sigma}_{x_p} = \frac{1}{250}\sum_{i=1}^{250} \sigma_{x_p}^i$
11:     $b(\sigma_{x_p}) = (\bar{\sigma}_{x_p} - \sigma_{x_p}^{MC})/(\sigma_{x_p}^{MC})$
12:     $\sigma(\sigma_{x_p}) = \frac{1}{249}\sum_{i=1}^{250}(\sigma_{x_p}^i - \bar{\sigma}_{x_p})^2$
13:     $ce_p = (\frac{1}{250}\sum_{i=1}^{250} co_p^i) - 0.68$
14: end for
    **Find average accuracies**
15: $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|} = \frac{1}{P}\sum_{p=1}^{P}|b(\sigma_{x_p})|$   $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\not{\mathcal{S}}} = \frac{1}{P-|\mathcal{S}|}\sum_{p\notin\mathcal{S}}|b(\sigma_{x_p})|$   $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{p\in\mathcal{S}}|b(\sigma_{x_p})|$
16: $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|} = \frac{1}{P}\sum_{p=1}^{P}|ce(\sigma_{x_p})|$   $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\not{\mathcal{S}}} = \frac{1}{P-|\mathcal{S}|}\sum_{p\notin\mathcal{S}}|ce(\sigma_{x_p})|$
17: $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{p\in\mathcal{S}}|ce(\sigma_{x_p})|$
18: $\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_{\mathbf{x}})} = \frac{1}{P}\sum_{p=1}^{P}\sigma(\sigma_{x_p})$   $\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_{\mathbf{x}})}_{\not{\mathcal{S}}} = \frac{1}{P-|\mathcal{S}|}\sum_{p\notin\mathcal{S}}\sigma(\sigma_{x_p})$   $\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_{\mathbf{x}})}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{p\in\mathcal{S}}\sigma(\sigma_{x_p})$
```

This means that there are lots of accuracies and the comparison between the different methods is difficult. To make the comparison more interpretable the averages are calculated. For the bias average $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}$ and the confidence interval error average $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|}$ the average of absolute values is taken. This way the positive errors on one coefficient do not cancel out negative errors of another coefficient. Next to the total averages, also the averages over the support and non-support part are calculated. The support of the vector $\mathbf{x}$ are the non-zero coefficients and the set of these indices is denoted with $\mathcal{S}$. For $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\not\mathcal{S}}$ and $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\not\mathcal{S}}$ the averages are taken over all the coefficients that are not in $\mathcal{S}$. For $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\mathcal{S}}$ and $\overline{|ce(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\mathcal{S}}$ the averages are taken over all the coefficients that are in $\mathcal{S}$.$|\mathcal{S}|$ refers to the cardinality (size) of the set $\mathcal{S}$, i.e. how many coefficients are non-zero.

$$b(\sigma_{x_p}) = \frac{\sigma_{x_p} - \sigma_{x_p}^{MC}}{\sigma_{x_p}^{MC}} \tag{2-24}$$

$$f_{coverage}(x^i_{p(16\%)}, x^i_{p(84\%)}, x_p^{true}) = c0^i_p = \begin{cases} 1 & \text{if } x_p^{true} \in [x^i_{p(16\%)} \cdots x^i_{p(84\%)}] \\ 0 & \text{if } x_p^{true} \notin [x^i_{p(16\%)} \cdots x^i_{p(84\%)}] \end{cases} \tag{2-25}$$

2-6-2 Scalar scenario

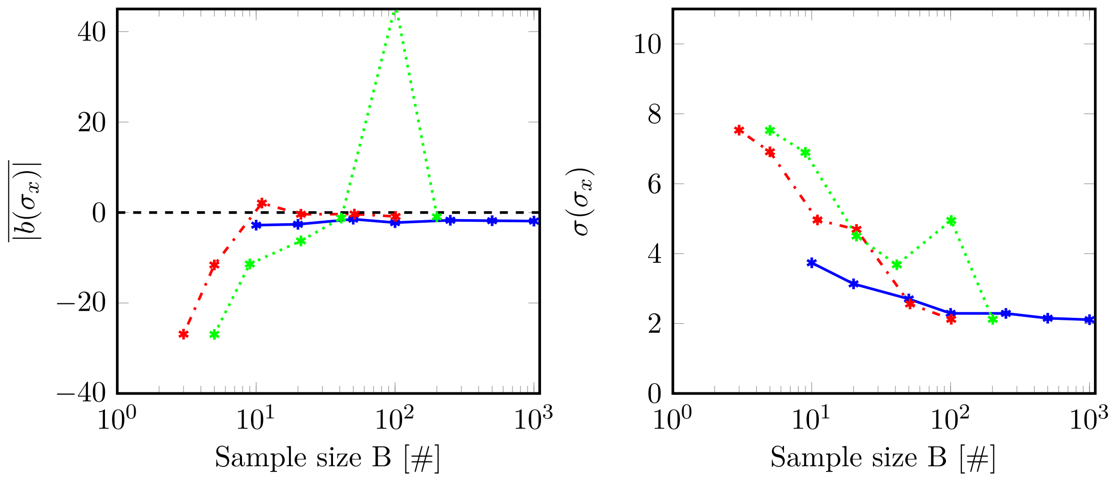
To assess the accuracy, a scalar scenario, the same situation as described by (Kyung and Casella, 2010), will be used. The parameters of the scalar test setting are defined in Table 2-5. The data-generation is done according to formula $y = ax + \varepsilon$. x is here set to 0.16 and the variance $\sigma_\varepsilon^2 = 0.01$. a is generated as a standard zero mean Gaussian input, so $f_{generating}(\boldsymbol{\mu}^A, (\boldsymbol{\sigma}^A)^2) = \mathcal{N}(0, 0.01)$. Note that $\lambda = 0.19$ is chosen based on cross-validation (CV), corresponding to the parameter used in the paper by (Kyung and Casella, 2010). The results of the different methods are listed in Table 2-6. For the Modified bootstrap the parameter c_N of the was found to be 0.04.
For several reductions of the Variance and 2-fold Variance UT the comparison is also done, resulting in Figures 2-5a and 2-5b. In Figure 2-5a the bias is shown for the reduced Variance and reduced 2-fold Variance UT in red and green respectively; the bootstrapping results with different sample sizes in blue. The same is done for the standard deviation of $\sigma_{\mathbf{x}}$ in Figure 2-5b.

Table 2-5: Parameters of the scalar scenario.

$y = ax^{true} + \varepsilon$	$x^{true} = 0.16$	$a \sim N(0,1)$	$\varepsilon \sim \mathcal{N}(0, 0.01)$
$\lambda = 0.19$	$N = 50$	$P = 1$	$c_N = 0.04$

Table 2-6: Accuracies in the scalar scenario, with parameters of Table 2-5.

	$\bar{\sigma}_x$	$b(\sigma_x)$	$\sigma(\sigma_{\mathbf{x}}) \cdot 1000$	$ce(\sigma_x)$
Monte Carlo	0.0145			
Approximation of Tibshirani	0.0141	-2.1%	2.07	-1.3%
Approximation of Osborne	0.0141	-2.1%	2.07	-1.3%
Vector resampling	0.0143	-0.9%	2.65	2.3%
Residual resampling B = 100	0.0141	-2.2%	2.29	0.7%
Residual resampling B = 1000	0.0142	-1.9%	2.11	0.7%
Parametric bootstrap	0.0143	-1.0%	2.20	1.9%
Fast residual resampling	0.0140	-3.1%	2.05	1.9%
Proposed fast residual resampling	0.0140	-3.2%	2.04	2.7%
Double bootstrap	0.0142	-1.9%	2.35	5.5%
Fast double bootstrap	0.0140	-3.4%	2.28	6.7%
Proposed fast double bootstrap	0.0141	-2.2%	2.29	7.1%
Modified bootstrapping	0.0142	-1.8%	2.14	3.1%
Scalar UT	0.0106	-26.9%	7.53	21.1%
2-fold Scalar UT	0.0106	-26.9%	7.53	21.1%
3-fold Scalar UT	0.0106	-26.9%	7.53	21.1%
Variance UT	0.0143	-0.9%	2.12	-1.3%
2-fold Variance UT	0.0143	-1.0%	2.12	-1.3%
3-fold Variance UT	0.0143	-0.9%	2.12	-1.3%



(a) Bias of σ_x **(b)** Standard deviation of σ_x

Figure 2-5: Results for the scalar scenario of Table 2-5 using different amount of sigma points for the reduced Variance UT (dashed dotted red) and reduced 2-fold Variance UT (dotted green) respectively. The number of sigma points used is shown on the horizontal axis. The vector resampling results are presented with a solid blue line, where B is the number of bootstrap samples.

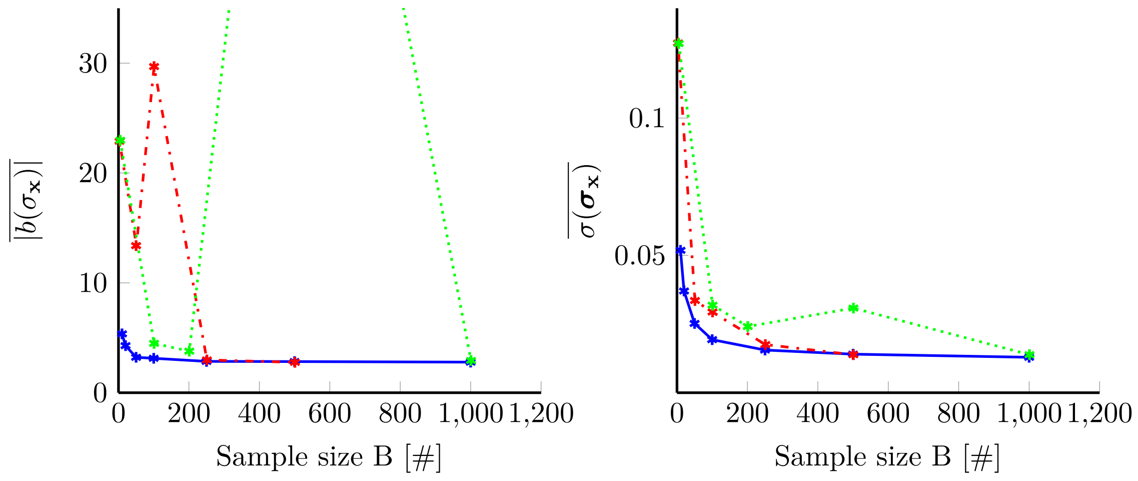
2-6-3 Nominal multidimensional scenario

To assess the accuracy in a multidimensional case, the scenario of (Sartori, 2011) is used. This scenario uses an $\mathbf{x}$ of which 6 regressors are non-zero and four are zero. The parameters of this scenario are given in Table 2-7; the λ parameter was not given, only that it was based on CV. Using CV for optimizing the MSE, a value of 0.2 was found. The $f_{generating}$ function is now a uniform distribution function on the interval $[-0.5, 0.5]$ for each value in A. The value of c_N was found to be 0.2. The results of this comparison per regression coefficient are given in Appendix C-1 and the averages are given in Table 2-8.

For several reduction of the Variance UT and 2-fold Variance UT the comparison is also done, resulting in Figures 2-6a and 2-6b. In Figure 2-6a the average absolute bias is shown for the reduced Variance and reduced 2-fold Variance UT Transform in red and green respectively; the bootstrapping results with different sample sizes in blue. The same is done for the standard deviation of $\sigma_{\mathbf{x}}$ in Figure 2-6b.

Table 2-7: Parameters of the nominal scenario

$y = \mathbf{a}\mathbf{x}^{true} + \varepsilon$	$\mathbf{x}^{true} = \begin{bmatrix} 0.75 & 2 & 0 & 0 & 0 & 10 & -3 & 1.5 & 0 & -0.65 \end{bmatrix}^T$		$a_{n,p} \sim u[-0.5, 0.5]$	
$\lambda = 0.2$	$\varepsilon \sim \mathcal{N}(0, 1)$	$N = 250$	$P = 10$	$c_N = 0.2$



(a) Average absolute bias. **(b)** Average Standard deviation of $\sigma_{\mathbf{x}}$.

Figure 2-6: Results for the nominal scenario of Table 2-7 using different amount of sigma points for the reduced Variance UT (dashed dotted red) and reduced 2-fold Variance UT (dotted green) respectively. The number of sigma points used is shown on the horizontal axis. The vector resampling results are presented with a solid blue line, where B is the number of bootstrap samples.

Table 2-8: Average accuracies in the nominal scenario, with parameters of Table 2-7.

	$\overline{\lvert b(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\mathcal{S}}$	$\overline{\lvert b(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\not{\mathcal{S}}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\not{\mathcal{S}}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_{\mathbf{x}})}_{\mathcal{S}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_{\mathbf{x}})}_{\not{\mathcal{S}}}$
Approximation of Tibshirani	3.3%	7.2%	2.9%	2.1%	0.012	0.024
Approximation of Osborne	2.8%	1.1%	2.7%	1.3%	0.012	0.012
Vector resampling	1.0%	0.7%	1.8%	1.9%	0.016	0.015
Residual resampling B = 100	3.4%	2.7%	4.0%	2.6%	0.019	0.019
Residual resampling B = 1000	3.0%	2.5%	2.0%	2.7%	0.013	0.013
Parametric bootstrap	2.8%	2.4%	2.5%	2.1%	0.013	0.013
Fast residual resampling	2.7%	2.8%	2.6%	2.2%	0.016	0.119
Proposed fast residual resampling	3.2%	1.3%	2.3%	2.2%	0.013	0.013
Double bootstrap	3.1%	2.8%	3.4%	5.4%	0.019	0.019
Fast double bootstrap	2.9%	2.5%	2.7%	2.4%	0.021	0.120
Proposed fast double bootstrap	3.4%	1.4%	3.5%	3.9%	0.020	0.019
Modified bootstrap	2.9%	2.5%	2.4%	2.2%	0.013	0.013
Scalar UT	23.4%	22.1%	19.0%	19.8%	0.127	0.127
2-fold Scalar UT	23.5%	22.2%	19.0%	20.1%	0.127	0.127
3-fold Scalar UT	23.4%	22.1%	19.0%	20.1%	0.127	0.127
Variance UT	2.8%	2.7%	2.7%	6.9%	0.012	0.016
2-fold Variance UT	2.9%	3.0%	2.8%	6.7%	0.012	0.016
3-fold Variance UT	2.8%	2.9%	2.8%	6.6%	0.012	0.018

2-6-4 High shrinkage scenario

To see the effect of the shrinkage on the accuracies a scenario is defined with a very high value of λ. In Table 2-9 the parameters of the simulation are shown, they are exactly the same as the nominal scenario, only now λ is set to $\lambda = 5$. The results of this comparison per regression coefficient are given in Appendix C-2 and the averages are given in Table 2-10. The value of c_N was found to be 0.275.

Table 2-9: Parameters for the high shrinkage scenario

$y = \mathbf{a}\mathbf{x}^{true} + \varepsilon$	$\mathbf{x}^{true} = \begin{bmatrix} 0.75 & 2 & 0 & 0 & 0 & 10 & -3 & 1.5 & 0 & -0.65 \end{bmatrix}^T$		$a_{n,p} \sim u[-0.5, 0.5]$	
$\lambda = 5$	$N = 250$	$\varepsilon \sim \mathcal{N}(0,1)$	$P = 10$	$c_N = 0.275$

Table 2-10: Average accuracies in the high shrinkage scenario, with parameters of Table 2-9.

	$\overline{\lvert b(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\mathcal{S}}$	$\overline{\lvert b(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\not\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\not\mathcal{S}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})}_{\mathcal{S}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})}_{\not\mathcal{S}}$
Approximation of Tibshirani	11.3%	8.9%	6.3%	2.3%	0.015	0.070
Approximation of Osborne	1.5%	59.9%	2.5%	18.8%	0.012	0.012
Vector resampling	0.9%	16.9%	2.0%	13.3%	0.017	0.027
Residual resampling B = 100	3.7%	5.1%	8.4%	2.4%	0.022	0.025
Residual resampling B = 1000	3.4%	5.4%	8.4%	3.1%	0.016	0.021
Parametric bootstrap	3.2%	5.7%	8.6%	2.8%	0.016	0.022
Fast residual resampling	85.0%	572.3%	7.3%	2.2%	1.700	5.244
Proposed fast residual resampling	1.8%	59.4%	7.2%	2.4%	0.013	0.013
Double bootstrap	3.5%	4.7%	25.6%	26.3%	0.022	0.026
Fast double bootstrap	85.0%	576.4%	23.6%	14.9%	1.699	5.471
Proposed fast double bootstrap	1.9%	59.3%	25.2%	26.2%	0.019	0.019
Modified bootstrap	3.3%	1.4%	9.0%	2.2%	0.017	0.020
Scalar UT	21.6%	4.5%	19.0%	8.7%	0.131	0.117
2-fold Scalar UT	21.5%	4.7%	19.0%	5.9%	0.131	0.115
3-fold Scalar UT	21.4%	4.7%	18.9%	4.9%	0.131	0.115
Variance UT	1.4%	4.0%	2.3%	28.2%	0.014	0.100
2-fold Variance UT	1.5%	3.5%	2.6%	16.1%	0.014	0.099
3-fold Variance UT	1.4%	3.5%	2.6%	12.2%	0.014	0.099

2-6-5 $P > N$ scenario

An even more difficult scenario is the undetermined case where $P > N$, a scenario with more features than data points. The parameters of this scenario are given in Table 2-11. 25 features are used, with 20 data points provided. λ was cross-validated to be 0.045 and c_N was found to be 0.30. The results of this comparison per regression coefficient are given in Appendix C-3 and the averages are given in Table 2-12.

Table 2-11: Parameters of the $P > N$ scenario

$y = \mathbf{a}\mathbf{x}^{true} + \varepsilon$	$\mathbf{x}^{true} = [0.75 \;\; 2 \;\; 0 \;\; 0 \;\; 0 \;\; 10 \;\; -3 \;\; 1.5 \;\; 0 \;\; -0.65 \;\; 0 \;\; \dots \;\; 0]^T$			
$a_{n,p} \sim u[-0.5, 0.5]$	$\lambda = 0.045$	$\varepsilon \sim \mathcal{N}(0, 0.05)$	$N = 20$	$P = 25$

Table 2-12: Average accuracies in the $P > N$ scenario, with parameters of Table 2-11.

	$\overline{\lvert b(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\mathcal{S}}$	$\overline{\lvert b(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\not\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_\mathbf{x})\rvert}_{\not\mathcal{S}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})}_{\mathcal{S}}$	$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})}_{\not\mathcal{S}}$
Approximation of Tibshirani	20.8%	11.7%	28.8%	16.9%	0.011	0.021
Approximation of Osborne	26.6%	45.4%	32.8%	7.4%	0.014	0.014
Vector resampling	1092.0%	898.2%	2.6%	14.0%	0.151	0.101
Residual resampling B = 100	34.1%	37.8%	38.8%	44.3%	0.011	0.011
Residual resampling B = 1000	34.3%	37.4%	38.1%	44.5%	0.010	0.010
Parametric bootstrap	32.5%	35.4%	37.6%	44.2%	0.011	0.011
Fast residual resampling	415.6%	656.2%	46.4%	45.3%	1.318	0.912
Proposed fast residual resampling	28.4%	41.7%	65.0%	64.7%	0.014	0.014
Double bootstrap	34.4%	38.1%	42.9%	39.1%	0.010	0.011
Fast double bootstrap	406.6%	648.8%	57.2%	56.3%	1.248	0.879
Proposed fast double bootstrap	28.4%	41.5%	66.9%	66.6%	0.014	0.014
Modified bootstrap	62.7%	77.8%	7.0%	10.1%	0.038	0.030
Scalar UT	37.3%	38.3%	41.4%	30.7%	0.021	0.015
2-fold Scalar UT	33.3%	31.7%	39.2%	19.8%	0.021	0.016
3-fold Scalar UT	32.9%	31.2%	38.8%	16.9%	0.021	0.016
Variance UT	23.2%	25.8%	34.4%	32.2%	0.013	0.017
2-fold Variance UT	23.0%	24.7%	34.6%	19.4%	0.013	0.017
3-fold Variance UT	22.9%	24.5%	35.1%	13.7%	0.013	0.017

2-6-6 Few data points scenario

To make a clear distinction between having little data points and the underdetermined case, a scenario with few data points is defined. The parameters are given in Table 2-13; 25 regressors are used, with 30 data points provided. λ was cross-validated to be 0.082 and c_N was found to be 0.38. The results of this comparison per regression coefficient are given in Appendix C-4 and the averages are given in Table 2-14.

Table 2-13: Parameters of the few data points scenario.

$y = \mathbf{a}\mathbf{x}^{true} + \varepsilon$	$\mathbf{x}^{true} = \begin{bmatrix} 0.75 & 2 & 0 & 0 & 0 & 10 & -3 & 1.5 & 0 & -0.65 & 0 & \dots & 0 \end{bmatrix}^T$
$a_{n,p} \sim u[-0.5, 0.5]$	$\lambda = 0.082 \quad \varepsilon \sim \mathcal{N}(0, 0.05) \quad N = 30 \quad P = 25$

Table 2-14: Average accuracies in the few data points scenario, with parameters of Table 2-13.

	$\overline{\lvert b(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\mathcal{S}}$	$\overline{\lvert b(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\not\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\mathcal{S}}$	$\overline{\lvert ce(\boldsymbol{\sigma}_{\mathbf{x}})\rvert}_{\not\mathcal{S}}$	$\overline{\sigma(\boldsymbol{\sigma}_{\mathbf{x}})}_{\mathcal{S}}$	$\overline{\sigma(\boldsymbol{\sigma}_{\mathbf{x}})}_{\not\mathcal{S}}$
Approximation of Tibshirani	29.5%	2.6%	13.3%	14.9%	0.007	0.014
Approximation of Osborne	24.8%	148.0%	9.4%	26.1%	0.023	0.024
Vector resampling	214.6%	140.0%	2.1%	12.1%	0.065	0.029
Residual resampling B = 100	38.8%	39.9%	28.6%	33.8%	0.006	0.006
Residual resampling B = 1000	38.6%	39.4%	27.8%	34.0%	0.006	0.006
Parametric bootstrap	37.5%	38.0%	27.8%	33.5%	0.006	0.006
Fast residual resampling	146.5%	379.3%	27.1%	24.9%	0.438	0.567
Proposed fast residual resampling	22.7%	143.9%	22.0%	16.1%	0.023	0.023
Double bootstrap	38.6%	39.9%	38.9%	33.9%	0.006	0.006
Fast double bootstrap	137.3%	359.3%	46.4%	43.1%	0.396	0.511
Proposed fast double bootstrap	22.6%	144.1%	41.7%	39.2%	0.023	0.024
Modified bootstrap	13.2%	11.1%	16.4%	13.6%	0.008	0.006
Scalar UT	42.1%	35.5%	26.8%	15.7%	0.018	0.014
2-fold Scalar UT	40.2%	30.8%	25.2%	7.2%	0.018	0.014
3-fold Scalar UT	40.1%	30.4%	25.0%	4.2%	0.018	0.014
Variance UT	32.7%	33.0%	17.0%	24.9%	0.007	0.013
2-fold Variance UT	32.7%	31.7%	17.2%	7.1%	0.007	0.013
3-fold Variance UT	32.6%	31.6%	17.6%	3.4%	0.007	0.013

2-6-7 Computational expense

To have an indication of computational expense of the methods the number of required LASSO to be solved and inverses has been summarized in Table 2-15. The time necessary to run the find the covariance and confidence interval for one data set in the nominal scenario is given in the third column. This simulation was run in Matlab on a PC with a 2.7GHz processor and 4GB RAM. These values of course vary with the length of the data set and the used computer, but give a relative indication of the relative computational expense. In the last column the normalized time is given, which is the time divided by the time it took to do the approximation of Tibshirani. The approximations are sorted from fast at the top to slow at the bottom. Note that in this case $N = 250$, $B = 1000$ for all bootstrap methods and $B = D = 100$ for the double bootstrap methods.

Table 2-15: Comparison of computational expense, the time is measured using the nominal scenario.

	# LASSOs	**# inverses**	**time [s]**	**normalized time**
Approximation of Tibshirani	-	1	$5.6 \cdot 10^{-3}$	1
Approximation of Osborne	-	1	$2.0 \cdot 10^{-2}$	4
UT Scalar 2 fold	4	-	$4.6 \cdot 10^{-2}$	8
UT Scalar	2	-	$5.6 \cdot 10^{-2}$	10
UT Scalar 3 fold	6	-	$6.6 \cdot 10^{-2}$	12
Fast residual resampling	-	1	1.23	221
Proposed fast residual resampling	-	1	1.29	232
UT Variance	$2N$	-	3.44	620
UT Variance 2 fold	$4N$	-	6.46	1164
Modified bootstrap	B	-	7.54	1359
Residual resampling	B	-	7.58	1366
Vector resampling	B	-	8.44	1522
UT Variance 3 fold	$6N$	-	9.28	1671
Fast Double bootstrap	-	1	12.0	2164
Proposed Fast Double bootstrap	-	B	13.9	2501
Double bootstrap	$B \times D$	-	84.6	15238

2-7 Discussion

The performance of the methods clearly differs between the scenarios. In the nominal and scalar situation most of the methods perform quite well, big differences are observed when λ is large, when the $P > N$ or when there are few data points. The found accuracy will be discussed per method below.

Analytical approximations
For the scalar scenario both approximations give exactly the same results, which is not strange as there is no regression coefficient equal to zero. In the nominal scenario the approximation of Osborne outperforms the approximation of Tibshirani. This is to be expected, since it is a rewritten formula to better handle the scenario in which some of the coefficients are equal to zero. In the nominal scenario four coefficients are zero and a clear difference is observed in the resulting bias: $\overline{|b(\boldsymbol{\sigma}_{\mathbf{x}})|}_{\mathcal{S}}$ is about a sevenfold lower for the approximation by Osborne. It is remarkable how well both methods recover the confidence interval, since they have the incorrect assumption of a Gaussian distribution of $\mathbf{x}$. Apparently in these scenarios this assumption is not unrealistic.
In the scenario of high shrinkage the approximation of Osborne has a much higher bias of 60% for the non-support elements versus 9% of the approximation of Tibshirani. The reverse is true for the support, which is much better estimated by Osborne. The large standard deviation of the approximation of Tibshirani however makes the low average accuracy not valuable, the spread is too high to rely on this method. Therefore the approximation of Osborne is preferred, only the results for the support part should be trusted.
In the $P > N$ scenario the approximation of Tibshirani outperforms all other methods, which is completely unexpected. No clear explanation has been found for this phenomenon. It could be that the simplicity of the formula helps and the use of the pseudoinverse prevents the non-invertability.
In general the approximation of Osborne should be used instead of the approximation of Tibshirani in relative simple scenarios as it gives the best confidence interval reconstruction and has a very low bias for the support elements. For scenarios with high shrinkage only the results for the support part should be trusted. The approximation of Osborne is about four times slower than approximation of Tibshirani, but still very fast.

Vector resampling
Vector resampling is performing really well in the nominal and scalar scenario. In the high shrinkage scenario the performance is poor and in the $P > N$ and few data points it gives a dramatic performance and should definitely not be applied. The latter is due to the very low amount of samples, $N = 20$. In the literature the Vector resampling method is recommended in the case of A being randomly generated, which was the case for all the simulations in this section. The results showed this effect, only in the case of few data points this effect negated. This can be explained by the fact that no new data is generated. The bootstrap elements are generated using samples which are chosen from the original data set. This means that always the same combination of $\mathbf{a}_n$ and y_n is used. In the residual resampling new data points are generated, making the algorithm more robust for few data points. This is visible in the $P > N$ and few data points scenario, where the biases are enormous. But also in the high shrinkage scenario the bias and level error are very large for the non-support elements. For correctly estimation the

covariance of the non-support with high shrinkage, shuffling the data set as is done in this method offers not enough information; sample generation as in residual resampling is necessary.
Concluding, the vector resampling should not be applied to the LASSO, as the shrinkage effects are not well captured.

Residual resampling
Residual resampling is a simple method and has a very consistent performance. In almost all cases both the bias and the level error is low, so all-round it is a very consistent method. The gain over the analytical approximations is low in the scalar and nominal scenario and the method is outcompeted in the high shrinkage and few data points scenarios by the modified bootstrap and in the $P > N$ scenario by the UT Variance 3-fold.
Concluding, it is a good method, but for simple scenario too computationally complex and for difficult scenario's a more sophisticated method could better be used.

Parametric bootstrap
The parametric bootstrap behaves very similar to the residual resampling and in many scenarios little difference can be observed. Even in the few data points scenario no major difference is observed. It could be that in a small sample size the residual from which are sampled are of insufficient diversity. Then first estimating the standard deviation and sampling from the Gaussian distribution with this standard deviation, as is done in the parametric bootstrap, would offer more diverse error terms. This is however no significant gain and thus should not be applied.

Modified bootstrap
The Modified bootstrapping is claimed to give a consistent distribution estimation. In the nominal and scalar scenario it performs very similar to residual resampling. In the high shrinkage scenario the effect of the extra shrinkage step is clearly visible: the biases and level error are lower compared to residual resampling. In fact in this scenario the modified bootstrap has overall the best performance. This effect is even more pronounced in the few data points scenario, here the modified bootstrap has a clearly higher performance.
In the $P > N$ scenario the performance is more difficult to assess: the bias is very large but the level error is low. Furthermore, it also has a high standard deviation. This is exactly the same in this scenario for the Vector resampling: very high bias and standard deviation, but low level error. This effect can be partially explained by the fact that if the covariance differs a lot per iteration and is on average wrong, the confidence intervals also varies a lot and do not correspond to the true distribution function. However in such a case the found level error could still be low, as the constructed confidence intervals could contain close to 68% of the times $\mathbf{x}^{true}$, because this can be done using numerous confidence intervals. In this case they do not correspond to the true distribution function. Thus the performance for an individual data set of the modified bootstrap is poor, since the found confidence interval and covariance do not correspond to the true distribution function. For the $P > N$ scenario the UT Variance 3-fold has in this case a much better all-round performance: the balance between bias and level error is better for this method.
The computational aspect as in Table 2-15 is very similar to Residual resampling.

However, the extra tuning step of c_N in the modified bootstrap is not included in the comparison. This is a significant extra computational burden as it requires running the Modified bootstrap for a large amount of times. The small increase of performance in the high shrinkage scenario compared to the residual resampling is not worth the much higher computational burden. For the few data points scenario this the increase of performance is larger and the extra computational costs is worth the gain.
Concluding, in simple scenarios the Modified bootstrap does not offer much extra performance, while it comes at a much higher computational cost. Only in the case of high shrinkage or few data points it offers a significant gain and is the recommended method. In the case of $P > N$ it performs worse and should not be used.

Double bootstrap
The double bootstrap method should recover the confidence intervals with a higher accuracy. This is not visible in any of the simulations, in fact the recovered confidence intervals are worse. This may be due to the fact that B and D are set to 100. Increasing this to for example 1000 could show a better recovery of the confidence intervals. However this is an unacceptable high computational burden, as it is more than a thousand times more time consuming than Residual resampling. In our simulations this would result in a computational time of more than two hours per data set.

Fast resampling
The fast resampling methods works reasonable in the nominal and scalar scenario. It is clear that the proposed fast resampling methods outperform the original fast resampling method in the more complex situations: high shrinkage, few data points and $P > N$. This is not strange, since the approximation by Osborne outperforms the approximation by Tibshirani. So, substituting the approximation by that of Osborne in the algorithm increases the performance. The residual resampling has a much higher accuracy, even using few iterations. Thus, these methods should not be considered further.

Variance UT
The Variance Unscented Transform has a good performance, very similar to that of the Residual resampling in the scalar and nominal scenario. This is a good achievement as the UT has never been applied for recovering the covariance of regression methods. In the high shrinkage scenario it does not recover the non-support elements well, the level error is high and there is a large standard deviation.
Introducing the higher orders does not decrease the bias, but the level error clearly decreases when more sigma points are used. This gives us a useful method to tune the required accuracy. It is remarkable that even though many of the data points have a very low weight as seen in Table 2-2 a clear increase of performance is visible.
The most interesting effect is visible in the $P > N$ scenario where the 3-fold Variance UT has a higher accuracy than all other bootstrap method. Note that in this scenario only 60 sigma points are used, while the performance is better than Residual resampling with 1000 samples. The first explanation could be the extra assumption on the noise model, Gaussian noise is assumed. This is also the case for the parametric bootstrap, but the performance of this bootstrap method is worse. This means that in the $P > N$ perturbating the output vector in a structured way, as in done in the Unscented Transform, is a better method compared to adding noise terms randomly sampled from a distribution as is done in the residual resampling. This effect is unexpected and very

useful.

Scalar UT / reduced Variance UT
The Scalar UT has a very poor performance in all the scenarios and should not be considered. Using only 2 data points oversimplifies the situation and the covariance cannot be captured accurately. In Figure 2-5a and Figure 2-5b it is clear that both the bias and the standard deviation converge to that of the Variance UT by increasing the size of $\Sigma_{\mathbf{y}}$ in the reduced Variance UTs. In the nominal scenario some unexplainable deviations from this convergence visible. This indicates that perturbating multiple samples at once, as is done in the Reduced Variance UT, is not a reliable method and should not be applied. Thus, a way of reducing the sigma points in a reliable manner below $2N$ has not been found.

Chapter 3

Including social media in stock market predictions

3-1 Introduction

In this chapter a wrapper feature selection method is sought for the nonlinear SVR, for a time-invariant and a time-varying SVR. This feature selection method shows which information source leads to which prediction, thereby also enhancing the interpretability of the results. To be able to compare the results to current literature, the found methods are tested using the same data set as used in the paper by (Bollen et al., 2010).

The first question is what data is available for stock market prediction, which is the subject of Section 3-2. The method of representing this data, the used models, will be discussed in Section 3-3. The regression methods used for predictions and Granger causality test for correlation are discussed in Section 3-4. Then the feature selection method for the SVR will be investigated and a new method will be developed for the time varying situation in Section 3-5. The used scenario (time periods) is presented in Section 3-6. The results of the correlation of the input data to the stock market data are investigated and the stock market predictions of the different methods are compared in Section 3-7. Finally a discussion is given in Section 3-8.

3-2 Data

Several sources of information can be used that can contain predictive information for the stock market. In this section these sources are discussed with respect to found results and to the availability of the data. First the VIX index is discussed, then technical indicators followed by social media information from Google.

3-2-1 Market Volatility index (VIX)

The VIX index is an index published by Chicago Board Options Exchange (CBOE) and shows the market's expectation of 30-day volatility. It is constructed using the implied volatilities of a wide range of index options (CBOE, 2014). In the analysis of (Mao et al., 2011) it was shown to correlate with the Dow Jones Index. In figure 3-1 the index is shown along with the Dow Jones opening price. At the different drops the VIX clearly has a peak, a visual indication of this correlation.

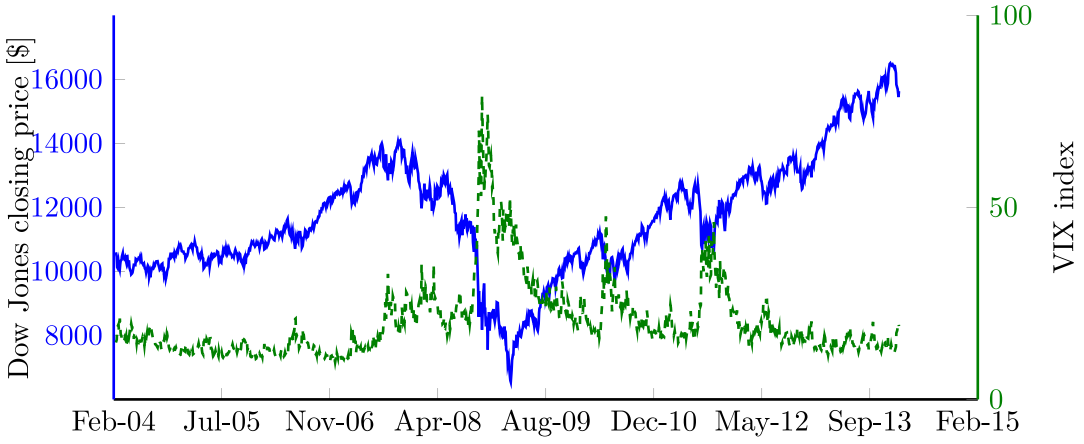


Figure 3-1: The Dow Jones closing price in blue and the VIX index dotted in green in the period of 2004 - 2013.

3-2-2 Technical indicators

In literature many representations of stock data are used. These data representations are often used for making decisions about buying or selling stocks. For example a certain representation index goes from positive to negative that is a sell signal. (Larsen, 2010) gives an overview of a few common representations used for prediction. Commonly used presentations have been listed in Table 3-1. These indicators are also added to the predictive models as additional data, to see if they can increase the prediction power.
In this table the following notations are used for the stock market data: the opening price o_n on day n, the closing price c_n on day n, the highest price of day h_n and the lowest price of the day l_n. The notation v_n is used for the trading volume.
The moving averages are the averages of the prices over a finite window length. These averages provide an indicator of the trend, where the Exponential moving averages EMA

provides a smoother trend-line that responds faster to trend shifts.
Candlestick representations are designed to predict short term trend reversals. They take into account the behavior of the stock market during the day, the highest and the lowest price value. Usually patterns of candlesticks are matched to the current behavior to lead to decision on the stock movement.
The Stochastic indices are momentum indicators. The idea is to detect when a stock is overbought or oversold. If a price has increased drastically over a short period, the price may reverse. The same holds for when a price has dropped drastically over a short period, then the price may reverse to rise again.
The money flow index is also a momentum indicator, but takes into account the volume. It uses a typical price (the average of closing, high and low) multiplied with the volume to determine the raw money flow. The raw money flow is positive when the typical price advances from one period to the next and negative when the typical price declines. The ratio is determined by looking at the proportion of positive, denoted by $(\dots)^+$, to negative raw money flow, denoted by $(\dots)^-$, over the last 14 days (Stockcharts, 2014a).
The rate of change is a pure momentum oscillator. The rate of change calculates the percentage change in price of the current price with the price 12 days ago (Stockcharts, 2014b).

Table 3-1: Commonly used technical indicators.

Description	Formula
Simple Moving Average 5 days	$SMA_t^5 = 1/5 \sum_{i=0}^{4} o_{n-i}$
Simple Moving Average 20 days	$SMA_n^{20} = 1/20 \sum_{i=0}^{19} o_{n-i}$
Exponential moving average	$EMA_n^1 = 0.05 o_n + (1-0.05) EMA_{n-1}^1$
Exponential moving average 5 days	$EMA_n^5 = 0.05 SMA_n^5 + (1-0.05) EMA_{n-1}^5$
Exponential moving average 20 days	$EMA_n^{20} = 0.05 SMA_n^{20} + (1-0.05) EMA_{n-1}^{20}$
Candlestick Range	$Candle_{1,n} = h_n - l_n$
Candlestick Body	$Candle_{2,n} = \frac{\lvert c_n - o_n \rvert}{h_n - l_n}$
Candlestick Upper Shadow	$Candle_{3,n} = \begin{cases} \frac{h_n - c_n}{h_n - l_n}, & \text{if } c_n > o_n \\ \frac{h_n - o_n}{h_n - l_n}, & \text{if } c_n < o_n \end{cases}$
Candlestick Lower Shadow	$Candle_{4,n} = \begin{cases} \frac{o_n - l_n}{h_n - l_n}, & \text{if } c_n > o_n \\ \frac{c_n - l_n}{h_n - l_n}, & \text{if } c_n < o_n \end{cases}$
Stochastic %K line	$K_n = 100 \frac{c_n - L_n}{H_n - L_n}$ $\quad H_n = max\{h_n \dots h_{n-13}\}$, $L_n = min\{l_n \dots l_{n-13}\}$
Stochastic $\%D^{fast}$ line	$D_n^{fast} = 1/3 \sum_{i=0}^{2} K_{n-i}$
Stochastic $\%D^{slow}$ line	$D_n^{slow} = 1/3 \sum_{i=0}^{2} D_{n-i}^{fast}$
Money flow index, typical price	$MFI_{1,n} = (h_n + c_n + l_n)/3$
Money flow index, raw money flow	$MFI_{2,n} = MFI_{1,n} v_n$
Money flow index	$MFI_{3,n} = \sum_{i=0}^{13} (MFI_{2,n-i})^+ / \sum_{i=0}^{13} (MFI_{2,n-i})^-$
Money flow index, ratio	$MFI_{4,n} = 100 - \frac{100}{1 - MFI_{3,n}}$
Rate of Change	$ROC_n = \frac{o_n - o_{n-12}}{o_{n-12}}$

3-2-3 Google Insights for Search (GIS)

For a certain search term, Google provides information on its use over a certain period. (Mao et al., 2011) have shown that for related search terms as ‘DJIA’ a correlation exists with the stock market price.
The data-stream gives values between 0 and 100 for its use for each week in the specified period, as seen in Figure 3-2. This normalization is done for the time frame provided, so the maximum in that period is set to 100. Furthermore, normalization is done with respect to the number of Internet uses. So if the absolute number of searches was constant, but more people gained access to the Internet the normalized value dropped. This means that a normalized decrease in search term does not necessarily mean an absolute decrease. For an increase, however, it is safe to assume that the absolute number of searches also increased. The data can be retrieved easily from Google.

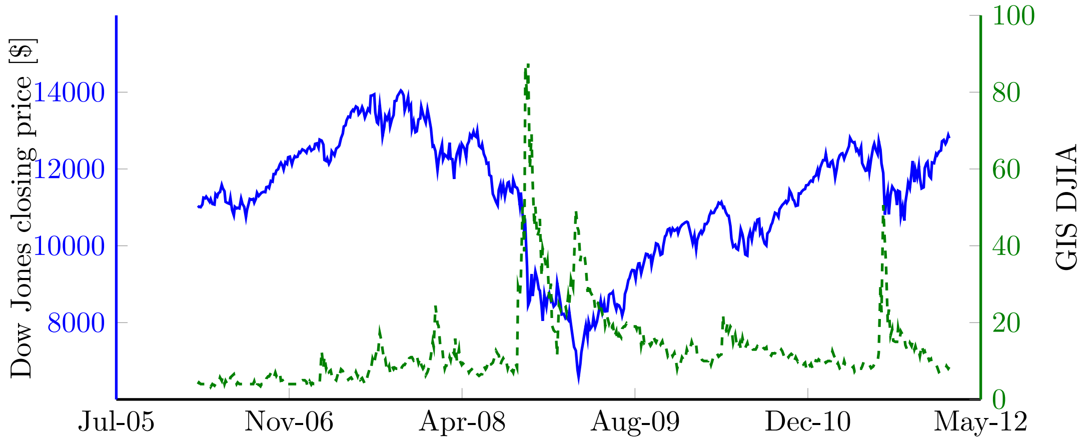


Figure 3-2: The Dow Jones closing price in blue and the GIS ‘DJIA’ dotted in green in the period of 2004 - 2012.

Normally the Google Insights for Search (GIS) information is available as daily information, but for retrieving information longer than 3 months ago, only weekly data is available. This makes it not directly implementable for regression in combination with the daily stock market data. Therefore the Google data is linearly interpolated to get ‘daily’ data of the search terms. This is done based on what day it is in the week, as shown in the example (3-1).

$$\text{google}_{\text{Thursday, week 2}} = \frac{4}{7}\text{google}_{\text{week 2}} + \frac{3}{7}\text{google}_{\text{week 3}} \tag{3-1}$$

(Mao et al., 2011) have shown correlation with 10 search terms to the Dow Jones Index from January 2008 to September 2011. Four of these indices with high correlation are included into this analysis. The search terms used are: ‘DJIA’, ‘Finance’, ‘Stock Market Today’, ‘Stock’, ‘Economy’, ‘Happy’, ‘Interest’, and ‘Crisis’. The first four are from (Mao et al., 2011) and the four others are added to this analysis.

3-3 Models

For the stock market prediction simulations different models will be used, where different data is included or excluded. These models will be used to see how well the used regression methods use the data with predictive power and discard the data without predictive power.
First the different models, numbered 1 to 6, are reviewed. Then the short-horizoning approach will be explained to deal with more time-varying behavior. Finally, the used normalization of the data is discussed.

3-3-1 Defined models

Model 1 (nom.): nominal
In the nominal model, the closing price of the stock is predicted using the previously three closing prices as in (3-2). This model is also used by (Bollen et al., 2010) and (Mao et al., 2011). Using a linear regression method this becomes an autoregressive (AR) model. The main disadvantage of this model is that linear models are often close to the estimate $\hat{y}_n = y_{n-1}$. A possible solution to this would be to try to predict the difference, $c_n - c_{n-1} = \begin{bmatrix} c_{n-1} - c_{n-2} & c_{n-2} - c_{n-3} & c_{n-3} - c_{n-4} \end{bmatrix} \mathbf{x}_n + \varepsilon$. This did not improve the situation, because the output that is to be predicted is fluctuating too heavily.

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{bmatrix} c_{n-1} & c_{n-2} & c_{n-3} \end{bmatrix} \mathbf{x}_n + \varepsilon \tag{3-2}$$

Model 2: nominal + specific GIS term
A specific search term of Google Insights for Search is added to the model by adding the three delayed values as in (3-3). This is the same as the implementation of social media by (Bollen et al., 2010).

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{bmatrix} c_{n-1} & c_{n-2} & c_{n-3} & google_{n-1} & google_{n-2} & google_{n-3} \end{bmatrix} \mathbf{x}_n + \varepsilon \tag{3-3}$$

Model 3 (gis): nominal + multiple GIS terms
All the Google Insights for Search terms are added to the model by adding the three delayed for each search term values as in (3-4).

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{matrix} [c_{n-1} & c_{n-2} & c_{n-3} & DJIA_{n-1} & DJIA_{n-2} & \dots \\ & \dots & Finance_{n-1} & Finance_{n-2} & \dots & Crisis_{n-3}] \end{matrix} \mathbf{x}_n + \varepsilon \tag{3-4}$$

Model 4 (vix): nominal + VIX
The Volatility index is added to the model by adding the three delayed values as in (3-5).

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{bmatrix} c_{n-1} & c_{n-2} & c_{n-3} & VIX_{n-1} & VIX_{n-2} & VIX_{n-3} \end{bmatrix} \mathbf{x}_n + \varepsilon \tag{3-5}$$

Model 5 (ind.): nominal + technical indicators

The technical indicators are added to the model only for the previous time step. So, no delayed values are taken into account, since these indices are generated to capture moments or trends. The model is given in (3-6).

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{bmatrix} c_{n-1} & c_{n-2} & c_{n-3} & SMA^5_{n-1} & SMA^{20}_{n-1} & \dots & ROC_{n-1} \end{bmatrix} \mathbf{x}_n + \varepsilon \quad (3\text{-}6)$$

Model 6 (tot1): nominal + multiple GIS terms + VIX + technical indicators
In the model of (3-7) all the different data-sources are combined.

$$c_n = \mathbf{a}_n \mathbf{x}_n + \varepsilon = \begin{matrix} [c_{n-1} & c_{n-2} & c_{n-3} & DJIA_{n-1} & \dots & Crisis_{n-3} \\ & VIX_{n-1} & \dots & VIX_{n-3} & SMA^5_{n-1} & \dots & ROC_{n-1}] \end{matrix} \mathbf{x}_n + \varepsilon \quad (3\text{-}7)$$

3-3-2 Short Horizoning (SH)

The importance of the social media data is likely to be time-varying. A clear indication of this time-varying behavior is given based on the time-varying correlation in Section 3-7. To deal with this time-varying behavior, regression methods will be run over a short horizon of the latest 75 data points, which moves along with the prediction. The regression method using this short horizon are denoted by a -SH suffix. If this suffix is not present the method will be trained over the full length of the available data set. Note that for the SH methods the retraining occurs for every step-ahead prediction, while the model is held constant for the whole test data set in the time-invarying methods.

3-3-3 Normalization

The stock market data needs some form of preprocessing before it can be used for prediction. For the linear models all the data is normalized to be zero mean and have a variance of 1. Note that this means all the features and the output are processed individually.
For the nonlinear models the same is done but then to get all the features and output normalized to the interval $[0, 1]$.

3-4 Regression- and correlation methods

In this section the used regression- and correlation methods are discussed. First the Granger Causality test, a common statistical test for correlation, is explained. Then the linear regression methods, OLS and LASSO are reviewed. For the nonlinear regression method the Support Vector Regression is used. After a short motivation the principle and tuning method are discussed.

3-4-1 Granger causality test

The bivariate Granger causality test is an econometric technique to prove that a time-series z is correlated to time-series y; its shows if time-series z is useful for predicting time-series y (Bollen et al., 2010). More formally we compare the variance explained by two linear models: one including a certain exogenous input and one without. Note that the word 'causality' is used, but the test only proves correlation not causation (Gilbert and Karahalios, 2010). The test used here is a F-test on the AR and ARX models using z and y; the AR model is given in (3-8) and the ARX model is given in (3-9). The claim in the test would be that the coefficients belonging to the exogenous input of the unrestricted ARX-model are equal to zero, which would reduce it to the restricted model, the AR-model: $H_0 : x_{z,1}, x_{z,2}, \dots, x_{z,D^z} = 0$. (SAS, 2014)

$$y_n = \sum_{i=1}^{D^y} x_{y,i} y_{n-i} + \varepsilon \tag{3-8}$$

$$y_n = \sum_{i=0}^{D^z-1} x_{z,i} z_{n-i} + \sum_{i=1}^{D^y} x_{y,i} y_{n-i} + \varepsilon \tag{3-9}$$

The method is written out in Algorithm 12. First the coefficients of these models are calculated using on OLS. The residuals using these coefficients provide the means for the test. The test-statistic of the H_0 is based on the Residual Sum of Squares (RSS) of both these models (3-10). The test statistic is defined in (3-11), where the value obtained can be compared against a critical value of the F-distribution with dimensions $(D^z, N - D^z - D^y)$ to accept or reject the H_0. The critical value can be calculated based on a desired p-value. Alternatively the F-value can be calculated into a p-value based on the cumulative distribution function (3-12) (Liao, 2011). The test is sensitive for the choice of parameters, i.e. D^y and D^z. To compare the results of the test with those of Bollen, exactly the same method is applied and both parameters are fixed at 3.

$$\begin{aligned} RSS_{AR} &= \sum_{n=1}^{N} \left(y_n - \sum_{i=1}^{D^y} x_{y,i} y_{n-i} \right)^2 \\ RSS_{ARX} &= \sum_{n=1}^{N} \left(y_n - \sum_{i=0}^{D^z-1} x_{z,i} z_{n-i} + \sum_{i=1}^{D^y} x_{y,i} y_{n-i} \right)^2 \end{aligned} \tag{3-10}$$

$$U = \frac{(RSS_{AR} - RSS_{ARX})/D^z}{RSS_{ARX}/(N - D^z - D^y)} \sim F_{D^z, N-D^z-D^y} \tag{3-11}$$

Algorithm 12 Granger causality test

```
    **Find the RSS of the AR model:**
1:  $\mathbf{x}_z = \text{argmin}_{x_{y,1} \dots x_{y,3}} \sum_{n=1}^{N} (y_n - \sum_{n=1}^{3} x_{y,i} y_{n-i})^2$
2:  $RSS_{AR} = \sum_{n=1}^{N} (y_n - \sum_{i=1}^{D^y} x_{y,i} y_{t-i})^2$
    **Find the RSS of the ARX model:**
3:  $\mathbf{x}_z = \text{argmin}_{x_{y,1} \dots x_{y,3}, x_{z,1} \dots x_{z,d}} \sum_{t=n}^{N} (y_n - \sum_{i=0}^{2} x_{z,i} z_{n-i} + \sum_{i=1}^{3} x_{y,i} y_{n-i})^2$,
4:  $RSS_{ARX} = \sum_{n=1}^{N} (y_n - \sum_{i=0}^{2} x_{z,i} z_{t-i} + \sum_{i=1}^{3} x_{y,i} y_{t-i})^2$
    **Calculate p-value:**
5:  p-value $= 1 - F\left(\frac{(RSS_{AR} - RSS_{ARX})/3}{RSS_{ARX}/(N-6)}\right)_{3,N-6}$
```

$$\text{p-value} = 1 - F(U)_{D^z, N-D^z-D^y} \tag{3-12}$$

Pre-processing

The Granger causality requires stationary time-series. The data from social media and the stock market is not stationary, therefore the z-score is used by (Bollen et al., 2010) as preprocessing. This score is given in (3-13) and it processes the time series x_n. This is done subtracting the mean and normalizing by the variance over an horizon of length $2k + 1$ containing the past and future values of the time series.

$$z(y_n) = \frac{y - \bar{y}_{n\pm k}}{\sigma_{y_{n\pm k}}} \qquad \bar{y}_{n\pm k} = \frac{1}{2k+1} \sum_{i=-k}^{k} y_{n+i} \qquad \sigma_{y_{n\pm k}} = \frac{1}{2k} \sum_{i=-k}^{k} (y_{n+i} - \bar{y}_{n\pm k})^2 \tag{3-13}$$

Windowed Granger causality test

For applying the Granger Causality test to a time-varying system a windowed test can be applied. The Granger causality test gets applied to a window, which goes over the horizon n_{start} to n_{end}. For each time instance in this horizon the Granger Causality p-value is this way found. This method is written out in Algorithm 13. l_{window} is the window length, the length of the subset of data on which the Granger Causality test is done.

Algorithm 13 Windowed Granger Causality test

```
1:  **for** $n = n_{start} \rightarrow n_{end}$ **do**
2:      $y_{test} = \begin{bmatrix} y_{n-l_{window}+1} & y_{n-l_{window}+2} & \cdots & y_n \end{bmatrix}^T$
3:      $z_{test} = \begin{bmatrix} z_{n-l_{window}+1} & z_{n-l_{window}+2} & \cdots & z_n \end{bmatrix}^T$
4:      Calculate p-value of Algorithm 12 using time-series $y_{test}$ and $z_{test}$
5:  **end for**
```

3-4-2 Ordinary Least Squares (OLS)

The OLS is a very simple linear regression method, which minimizes the squared prediction error as in (3-14).

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 \tag{3-14}$$

3-4-3 Least Angular Shrinkage and Selection Operation (LASSO)

The LASSO minimization problem of (Tibshirani, 1994) can be formulated as in (3-15). Here the ℓ_1-norm is implemented in order to obtain a sparse vector $\mathbf{x}$. This way the data that is not useful for prediction will be discarded automatically. Here the built-in solver of Matlab will be used. The tuning parameter λ, which controls the sparsity, is chosen based on a minimal Mean Squared Error (MSE) of 10-fold Cross Validation (CV).

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{x}\|_1 \tag{3-15}$$

3-4-4 Support Vector Regression (SVR)

Support Vector Regression is a nonlinear regression method and is a special class of Kernel Based Regression. It can be viewed as an alternative to neural networks, with the advantage that the problem can be rewritten to a quadratic programming method or to a least squares problem for Least Squares SVR (Smola and Schölkopf, 2004). Furthermore, it has been shown to achieve higher generalization, which is very relevant for stock market prediction (Li and Liu, 2011). First the Framework will be given for the Least Squares SVR, followed by the used tuning method. For the implementation in Matlab the LS-SVMlab by (De Brabanter et al., 2010) is used.

Framework
The optimization problem of the Least Squares SVR is given in (3-16). Here $\mathbf{v}^T\psi(\mathbf{a}_n)+b$ is the nonlinear estimation of y_n. The error values are denoted with e_n, $\psi(\dots)$ denotes the nonlinear transformation, and $\mathbf{v}$ the weight vector. C is a constant parameter that determines the allowance for nonlinearity.

$$\begin{aligned} \min_{\mathbf{v},b,\mathbf{e}} \quad & \mathbf{v}^T\mathbf{v} + C\mathbf{e}^T\mathbf{e} \\ \text{s.t.} \quad & y_n = \mathbf{v}^T\psi(\mathbf{a}_n) + b + e_n \quad \text{for} \quad n = 1 \dots N \end{aligned} \tag{3-16}$$

This quadratic programming problem can be rewritten using the method of Lagrange multipliers, the resulting objective function is given in (3-17). This leads to the Karush-Kuhn-Tucker conditions as in (3-18). Using these conditions the Least Squares problem as in (3-19) can be derived. (Coen et al., 2006)

$$\mathcal{L}(\mathbf{v}, b, \mathbf{e}, \boldsymbol{\alpha}) = \mathbf{v}^T\mathbf{v} + C\mathbf{e}^T\mathbf{e} - \sum_{n=1}^{N} \alpha_n\{\mathbf{v}^T\psi(\mathbf{a}_n) + b + e_n - y_n\} \tag{3-17}$$

$$\begin{aligned}
\frac{\delta\mathcal{L}}{\delta\mathbf{v}} = 0 \quad &\rightarrow \quad \mathbf{v} = \sum_{n=1}^{N} \alpha_n \psi(\mathbf{a}_n) \\
\frac{\delta\mathcal{L}}{\delta b} = 0 \quad &\rightarrow \quad \sum_{n=1}^{N} \alpha_n = 0 \\
\frac{\delta\mathcal{L}}{\mathbf{e}} = 0 \quad &\rightarrow \quad \boldsymbol{\alpha} = C\mathbf{e} \\
\frac{\delta\mathcal{L}}{\alpha_n} = 0 \quad &\rightarrow \quad \mathbf{v}^T \boldsymbol{\psi}(\mathbf{a}_n) + b + e_n - y_n = 0 \quad \text{for} \quad n = 1 \dots N
\end{aligned} \tag{3-18}$$

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ 1 & \boldsymbol{K} + I/C \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \tag{3-19}$$

Here, $K(\mathbf{a}_{n_1}, \mathbf{a}_{n_2}) = \psi(\mathbf{a}_{n_1})^T \psi(\mathbf{a}_{n_2})$ is the positive definite kernel and $\mathbf{1}$ is a vector of length N of ones. The way the prediction can be written out as in (3-20).

$$\hat{y}_n = \sum_{n_2=1}^{N} \alpha_{n_2} K(\mathbf{a}_n, \mathbf{a}_{n_2}) + b \tag{3-20}$$

The choice of the kernel in function (3-20) highly influences the prediction accuracy of the SVR. Several kernels can be chosen, in practice however the radial basis function (RBF) kernel often yields the best results and is most often used (Burges, 1998) (Nanye Long et al., 2011). The RBF kernel is given in (3-21), where the radial basis function width is determined by γ, the kernel parameter.

$$K(\mathbf{a}_{n_1}, \mathbf{a}_{n_2}) = exp(-\|\mathbf{a}_{n_1} - \mathbf{a}_{n_2}\|_2^2/\gamma^2) \tag{3-21}$$

Tuning
In the Least Squares SVR framework two tuning parameters are present: C and γ. The built-in tuning method of the LS-SVMlab toolbox is used for finding the optimal parameters. First a global optimization is run using Coupled Simulated Annealing (Xavier-De-Souza et al., 2010). The resulting parameters from this global optimization are fine-tuned using a grid-search. Both these optimization steps use 10-fold CV (De Brabanter et al., 2010).

3-5 Feature selection for SVR

A feature selection method for the SVR is sought which is not only accurate, but also has a reasonable computational complexity. The computational complexity is a very important aspect as evaluation of all possible combinations of features is infeasible. When there are P features present, 2^P subsets of the features are possible; this is even for low numbers of P infeasible to calculate. Therefore algorithms have been developed to get reasonable accuracy by calculating few subsets. One of these is the Recursive Feature Elimination (RFE) which has shown to give good results. This method (RFE-SVR) is discussed first, followed by an extension using a statistical test (SRFE-SVR). To be able to have feature selection with few data points, a new algorithm is proposed, the Weighted Recursive Feature Elimination. These methods are compared in a numerical simulation using an artificial data set.

3-5-1 Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is a procedure in which the model is first trained with all features and then the features are removed one at a time based on a ranking criterion. This criterion provides information on which feature is least useful for prediction (Guyon et al., 2002). The most logical choice for this criterion is the MSE, which leads to the Recursive Feature Elimination as proposed by (Li et al., 2009), given in Algorithm 14. The procedure is as follows: first the SVR is tuned for all the features and the corresponding MSE is calculated. Then one by one the features are removed based on which removal gives the lowest MSE. Note that this means that for each removal step for all the remaining features the MSE should be calculated. The trick here is to set the corresponding row in the A-matrix to zero instead of removing it. This way no retuning has to happen. This is referred to as the fast algorithm, which we compared with the exact version; the exact version has a retuning step at line 8, which is written out in Appendix B-7. In the numerical comparison it will become clear that this retuning leads to a marginal increase of accuracy.
The support is found by looking at which removal step resulted in the lowest MSE. Note that in the proposed framework of (Li et al., 2009), the MSEs were calculated based on the training set instead of a 10-fold CV. In simulations this was found to be inferior, so it has been replaced with the cross validated MSE. The notation $|\mathcal{S}^i|$ refers to the cardinality of the set $\mathcal{S}^i$.

3-5-2 Statistical Recursive Feature Elimination (SRFE)

(Yang and Ong, 2011) propose an improved Recursive Feature Elimination method by looking at the probabilistic prediction of the SVR. The output can be viewed as the prediction of the SVR with a noise term as in (3-22). When a probability distribution of ε is assumed, the distribution of $\hat{y}_n$ can be derived. For example a Gaussian distribution with standard deviation σ_ε can be assumed, in which case the probability of finding y_n given A and σ_ε can be derived as in (3-23). The idea of the SRFE-SVR is to compare the probability distribution of $p(y_n|A;\sigma_\varepsilon)$ to the probability density function after removing

Algorithm 14 Recursive Feature Elimination (RFE-SVR) (fast)

```
    **Initialize**
 1: $i = 1$, $\mathcal{S}^1 = \{1, 2, \dots P\}$, $A^1 = A$
 2: Tune SVR using $A^1$: find parameters $C^1 \& \gamma^1$
 3: Calculate 10-fold CV $MSE^1$ using $A^1, C^1 \& \gamma^1$.
 4: while $|\mathcal{S}^i| > 0$ do
 5:     for each $s \in \mathcal{S}^i$ do
 6:         Set $A_t = A^i$
 7:         Replace column $s$ of $A_t$ with zeros
 8:         Calculate 10-fold CV $MSE_t^s$ using $A_t, C^i \& \gamma^i$
 9:     end for
10:     Find the index $s_{min}$ belonging to the minimum of set $\{MSE_t^1, \dots, MSE_t^{|\mathcal{S}^i|}\}$
11:     $S^{i+1} = S^i \setminus s_{min}$
12:     $A^{i+1} = A^i$
13:     Remove row $s_{min}$ from $A^{i+1}$
14:     Tune SVR using $A^{i+1}$: find parameters $C^{i+1} \& \gamma^{i+1}$
15:     Calculate 10-fold CV $MSE^{i+1}$ using $A^{i+1}, C^{i+1} \& \gamma^{i+1}$
16:     i = i+1
17: end while
    **Find support delivering minimal MSE**
18: Find the index $s_{min}$ belonging to the minimum of set $\{MSE^1, \dots, MSE^{|\mathcal{S}^i|}\}$
19: Support is found: $\mathcal{S} = \mathcal{S}^{s_{min}}$
```

a certain feature. The smaller the difference in these distribution functions, the less important that feature is. Therefore a Kullback-Leibler divergence test-statistic S was derived to measure the difference.

$$y_n = \underbrace{\sum_{n_2=1}^{N} \alpha_{n_2} K(\mathbf{a}_n, \mathbf{a}_{n_2}) + b}_{\hat{y}_n} + \varepsilon \tag{3-22}$$

$$p(y_n|A; \sigma_\varepsilon) = \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\left(\frac{-\left(y_n - \hat{y}_n\right)^2}{2\sigma_\varepsilon^2}\right) \tag{3-23}$$

The estimation of σ_ε^2 based on the errors from CV is given in (3-24), it is the MSE of the cross validated errors (Lin and Weng, 2004). The Kullback-Leibler divergence function is given in (3-25), to measure the difference between the distributions $q(z)$ and $r(z)$. The test statistic is given in (3-26) for the removal of features s, where the D_{KL} is averaged over all the values in $\mathbf{y}$. This is written out in Appendix A-5. Here A^s denotes the matrix A, with column s set to all zeros, and σ_ε^s being the corresponding MSE and $\hat{y}_n^s$ the prediction.

$$\sigma_\varepsilon^2 = \frac{\sum_{n=1}^{N} (\hat{y}_n - y_n)^2}{N} \tag{3-24}$$

$$D_{KL}(q(z), r(z)) = \int_{-\infty}^{\infty} q(z) ln\left(\frac{q(z)}{r(z)}\right) dz \tag{3-25}$$

$$\begin{aligned} S^s &= \frac{1}{N}\sum_{n=1}^{N} D_{KL}(p(y_n|A), p(y_n|A^s)) \\ &= \frac{1}{2N}\sum_{n=1}^{N}\left[\frac{(\hat{y}_n^s - \hat{y}_n)^2}{(\sigma_\varepsilon^s)^2} + \frac{(\sigma_\varepsilon)^2}{(\sigma_\varepsilon^s)^2} + 2ln\left(\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon}\right)\right] \end{aligned} \tag{3-26}$$

The SRFE-SVR method is identical to that of the RFE-SVR, only now the choice on which feature to eliminate is based on a minimal S^s, not a minimal MSE. The method is written out in Appendix B-8.

3-5-3 Weighted Recursive Feature Elimination (WRFE) (proposed)

One of the assumptions in the SVR is that all the inputs are equally important, they all carry the same predictive power (Coen et al., 2006). This has as a consequence that, especially in the case of few data points, some inputs with less predictive power will not be selected in the RFE methods. A solution to this problem is to add a weighting in the Kernel function as in (3-27) (Hai-xian Zhao, 2011) (He et al., 2008). Here every feature has a weight between 0 and 1, where at 1 it is completely taken into account and at 0 the feature is neglected.

$$K(\mathbf{a}_{n_1}, \mathbf{a}_{n_2}) = exp\left(\frac{-\sum_{p=1}^{P} w_p(a_{n_1,p} - a_{n_2,p})^2}{\gamma^2}\right) \tag{3-27}$$

This new kernel function is mostly used for allowing more nonlinearity. Here a new RFE method is suggested to use this weighting to better estimate the support in case not all the features are equally important. This new method is written out in Algorithm 15. Here the fact that the RFE-SVR not just estimates the support, but also ranks their importance, is used. For the 25% least important features of the support and an equal amount of features from the non-support the weights are optimized for an improved cross validated MSE. This weight vector is initialized with ones for the elements estimated as support and zeros for those estimated as non-support elements. This optimization is done using a local-optimization technique, a gradient descent based algorithm. This is implemented in Matlab using the `lsqnonlin` function in Matlab.
A disadvantage of this method is overfitting; very small weights can be attributed to the features to have a better fit. This is not desired, as these inputs generally do not contribute to the prediction and is prevented by applying a thresholding on the weight vector after the optimization; small elements are set to zero. If the optimization has improved the MSE the new support is selected based on which weights are now larger than zero.

3-5-4 Simulation

To compare the described feature selection methods a numerical simulation will be run. This is done in the same way as (Yang and Ong, 2011) by comparing the approximation power of the formula (3-28). To capture the capacity to detect also a less important feature, the coefficient of x_5 is set to 0.3.

Algorithm 15 Weighted Recursive Feature Elimination (WRFE-SVR)

```
1: Find estimation of $\mathcal{S}$ using RFE-SVR (Algorithm 14)
2: Fixate the 75% most important features of the support $\rightarrow \mathcal{S}^{fix}$ and corresponding $A^{fix}$
3: Select the 25% least important features of the support $\rightarrow \mathcal{S}^{selec_1}$ and corresponding $A^{selec_1}$
4: Select the $|\mathcal{S}^{selec_1}|$ most important features of the non-support $\rightarrow \mathcal{S}^{selec_2}$ and corresponding $A^{selec_2}$
5: Initialize weight vector $\mathbf{w}^{selec_1} = \mathbf{1}$ and $\mathbf{w}^{selec_2} = \mathbf{0}$
6: Use gradient descent optimization to find optimal $\mathbf{w}$ based on 10-fold CV using the initial $[\mathbf{w}^{selec_1}\mathbf{w}^{selec_2}]$ using $[A^{selec_1}A^{selec_2}]$ and $A^{fix}$. Note that no weights are optimized for the fixated part $A^{fix}$, these are taken into account with a weight of 1.
7: Remove low weights: if $(w_p < 0.05)$ then $w_p = 0$
8: Compare MSE from using $\mathbf{w}$ to the MSE of the RFE-SVR algorithm
9: If the MSE has improved, use $\mathbf{w}$ and $\mathcal{S} = \{\mathcal{S}^{fix}$, and the elements from $\mathcal{S}^{selec_1}$ and $\mathcal{S}^{selec_2}$ for which $w_p > 0\}$
10: Else use $\mathcal{S}$ from the RFE-SVR
```

First data sets are generated using (3-28). Here 5 features are in the support, $a_{n,1}, \ldots, a_{n,5}$ and 5 other features are not in the support, they have no influence on the output $a_{n,6}, \ldots, a_{n,10}$. The values of the inputs $a_{n,p}$ are randomly sampled from a uniform distribution on $[0, 1]$. This way 30 training sets are generated, with varying number of data points: $N = 200, N = 100, N = 70, N = 50$. 30 validation data sets are generated using a fixed number of data points of 200.

On all these data sets the different feature selection methods are compared, of which the results are shown in Table 3-2. Note that in the top part of the table the average MSE is shown and in the bottom part the accuracy of estimating the right features (support). For each method and data set two percentages are present for the support estimation, the first percentage refers to the fraction of support elements are estimated as support. This percentage is the average of the number the number of correctly estimated support elements divided by the support length, 5. The second percentage to the fraction of the non-support elements estimated as non-support.

As reference the MSE values of the SVR run over only the support (correct support) and over all the features (including all) are also shown in the table.

$$y_n = 0.1e^{4a_{n,1}} + 4/(1 + e^{-20(a_{n,2}-0.5)}) + 3a_{n,3} + 2a_{n,4} + 0.3a_{n,5} + \mathcal{N}(0, 0.1^2) \qquad (3\text{-}28)$$

3-5-5 Discussion

From the simulation it is clear that feature selection is important, including all the features has a much higher MSE compared to selecting the correct support. The exact RFE-SVR is only marginally more accurate compared to the fast RFE-SVR and will thus not be considered further. The RFE-SVR has the best performance with many data points and is thus the preferred method for the SVR.

Table 3-2: Comparison of Feature Selection Methods, the best accuracies are formatted in bold.

		N=200	N=100	N=70	N=50
MSE	***Correct support***	**0.29**	0.56	0.75	**0.88**
	Including all	0.74	0.92	1.02	1.16
	RFE-SVR (exact)	**0.29**	0.58	0.81	1.15
	RFE-SVR (fast)	0.30	0.56	0.80	1.23
	SRFE-SVR	0.39	0.59	0.78	1.07
	WRFE-SVR	0.44	**0.54**	**0.65**	1.02
Support	**RFE-SVR (exact)**	**99%** / **99%**	85% / 87%	84% / 82%	89% / 78%
	RFE-SVR (fast)	98% / 98%	85% / 87%	86% / 84%	87% / 80%
	SRFE-SVR	95% / 88%	**91%** / **89%**	84% / 84%	88% / **82%**
	WRFE-SVR	79% / 91%	79% / 85%	**90%** / **85%**	**90%** / 79%

The SRFE-SVR performs very well when there are few data points, but has a worse performance when $N = 200$. The fact that this algorithm performs worse when the number of data points is high can be explained by the fact that the least important feature according to this method is the one which gives the same probabilistic prediction. This is in principle not necessarily true, as the prediction of the SVR improves when features are removed, so it makes no sense that the probabilistic prediction should remain the same. When the number of data points is high it is thus better to look at removing which features gives the biggest drop in MSE.
The WRFE-SVR also has a worse performance when the number of data points is high, this probably due to overfitting. For lower number of data points it has the best performance. The performance is close to the SRFE-SVR, but a distinct difference is that there is no assumption on the noise model. For stock market data it is not known what kind of noise model is present, so this assumption should not be made unless necessary. Therefore the WRFE is the preferred method for the SVR-SH, where the number of data points is low.

3-5-6 Conclusion

Thus, the RFE (fast) will be used for the SVR and the WRFE will be used for the SVR-SH. A new method has been successfully developed for selecting features when the number of data points is low and not all features are equally important.

3-6 Stock market scenario

To compare the results of the simulations to literature, the same data set will be used as in the publication by (Bollen et al., 2010). There the Dow Jones closing value is used between 28 February 2008 and 28 November 2008 for learning and between 1 and 19 December 2008 for testing. The closing price in this period is shown in Figure 3-3. This means that there are 191 data points for training and 19 for testing. In the pre-processing (Bollen et al., 2010) have interpolated the stock market data over weekends. The stock market is only open on weekdays, so for weekends no information is available. In order to have a real estimation of the predictive power, this interpolation is not applied. Only the available data of week days, Monday to Friday, is used in this analysis. The stock market data is acquired from (of St. Louis, 2013).

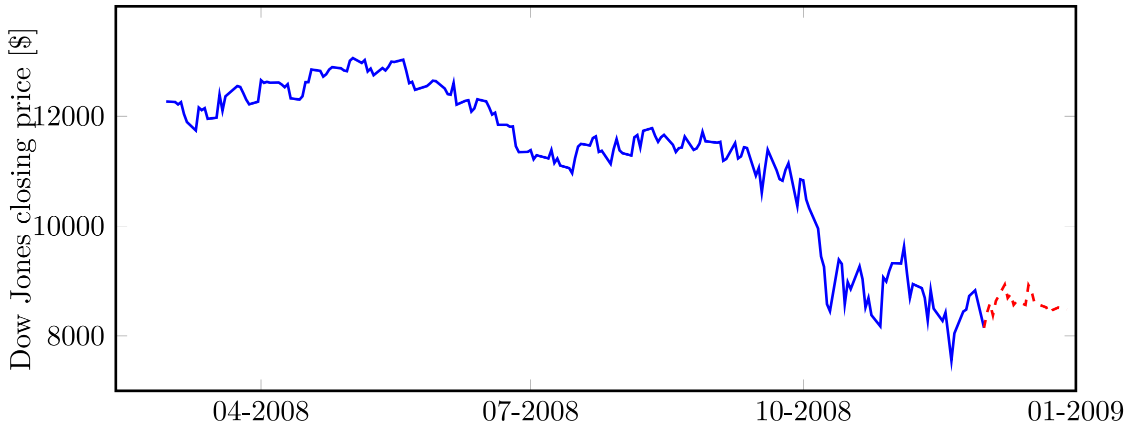


Figure 3-3: Plot of Dow Jones closing value from 28 February to 28 November 19 2008 in blue for learning and from 1 to 19 December 2008 in dashed-red for testing.

3-7 Results

In this section first the correlation between the different data-types is measured using a Granger Causality test. To motivate the time-varying behavior the windowed Granger causality test is performed. Then the measures of accuracy for the prediction results are discussed, followed by the prediction results.

3-7-1 Correlations

The Granger Causality test is applied to Google search terms and the VIX index to test the correlation with the Dow Jones closing price over the training set. In Table 3-3 the resulting p-values are shown for different delays. Here the investigated data is delayed with respect to the Dow Jones closing price. A higher delay thus means that the data predicts farther in the future. The lower the p-value, the higher the correlation is, and the p-values below 0.05 are formatted in bold. As a reference the listed p-values of two indices in the paper by (Bollen et al., 2010) are also included.

Table 3-3: Granger Causality test p-values for the literature scenario. The p-values below 0.05 are formatted in bold.

	n-1	**n-2**	**n-3**
DJIA	0.477	0.143	0.222
Finance	0.668	0.056	0.057
Stock today	0.645	0.098	**0.032**
Stock	0.254	0.187	0.468
Economy	0.289	0.260	0.335
Happy	0.684	0.851	0.996
Interest	0.078	0.157	0.377
Crisis	0.657	0.417	0.298
VIX	$\mathbf{3.72 \cdot 10^{-5}}$	$\mathbf{7.23 \cdot 10^{-7}}$	$\mathbf{9.07 \cdot 10^{-6}}$
Calm index [a]	0.272	**0.013**	**0.022**
Sure index [a]	0.648	0.811	0.35

3-7-2 Time-varying correlation

Stock markets are complex systems, time varying behavior could thus be expected and is also found in our analysis. If a windowed Granger test is done the time-varying behavior becomes visible. This is done for the search term 'Stock market today' in Figure 3-4, with a window length of 100 time samples.

[a]The Calm and Sure index results are from the paper by (Bollen et al., 2010).

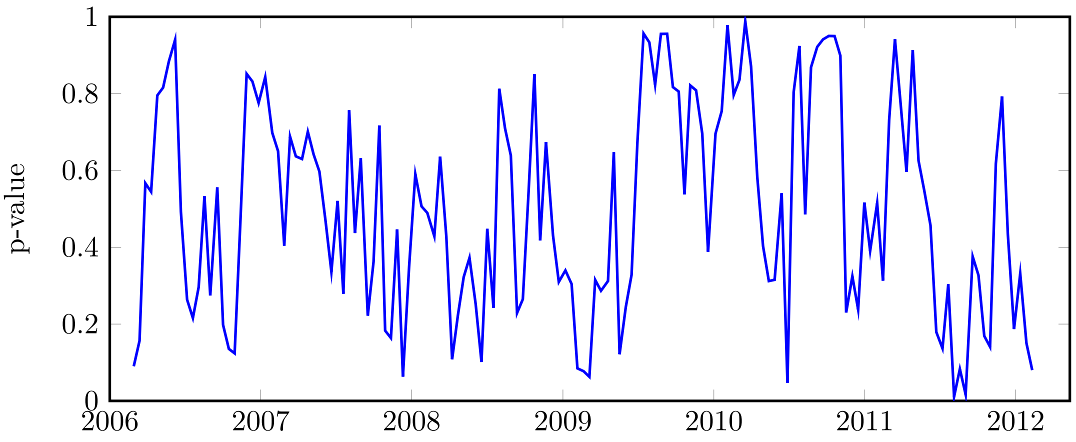


Figure 3-4: Plot of the p-value of the windowed Granger causality test using the Google search term 'Stock market today' with delay 3.

3-7-3 Prediction accuracy measures

Two measures for accuracy are used: the Directional Accuracy (DA) and the Mean Absolute Percentage Error (MAPE). The DA shows the percentage of the times the stock market direction is predicted correctly (3-29). The gamble percentage would be 50%, thus a DA around 50% is horrible and indicates that the model does not capture the stock market behavior. If the prediction is that the stock will increase but in reality it decreases this is a loss of money. The MAPE is the relative accuracy of the prediction (3-30).

$$\begin{aligned} DA_n &= \begin{cases} 1 & \text{if } (\hat{y}_{n+1} - y_n)(y_{n+1} - y_n) > 0 \\ 0 & \text{else} \end{cases} \\ \text{DA} &= 1/N \sum_{n=1}^{N} DA_n \end{aligned} \tag{3-29}$$

$$\text{MAPE} = 1/N \sum_{t=1}^{N} \left| \frac{y_n - \hat{y}_n}{y_n} \right| \tag{3-30}$$

3-7-4 Prediction results

In Table 3-4 the obtained results are listed for the different methods and models. Note that due to the CV, for each data set multiple models can be found using the same method. The CV data sets are randomly selected and the performance of these models differs slightly. To capture the average accuracy of the methods 10 simulations are run and the accuracies averaged. For model 6 (tot) the ranking of features by the SVR and SVR-SH is given in Table 3-5. The results of model 2, adding each Google search term, are given in Table 3-6. Note that based on the correlation results another model 2 using the search term 'Stock Today' is added with three days delay. As a comparison the results of Bollen et al. (2010) are given in Table 3-7.

Table 3-4: Accuracies of the methods.

DA	1 (nom.)	3 (gis.)	4 (vix.)	5 (ind.)	6 (tot.)
OLS	47%	58%	63%	47%	58%
OLS-SH	47%	58%	63%	63%	47%
LASSO	53%	58%	63%	63%	58%
LASSO-SH	47%	47%	47%	47%	47%
SVR	62%	52%	56%	68%	68%
RFE-SVR	66%	42%	59%	68%	51%
SVR-SH	63%	65%	52%	71%	63%
WRFE-SVR-SH	63%	66%	58%	59%	54%
MAPE	**1 (nom.)**	**3 (gis.)**	**4 (vix.)**	**5 (ind.)**	**6 (tot.)**
OLS	1.50%	2.08%	1.34%	2.91%	3.34%
OLS-SH	1.56%	10.83%	1.67%	3.89%	17.09%
LASSO	1.49%	1.93%	1.34%	1.44%	1.76%
LASSO-SH	7.10%	11.92%	6.46%	9.73%	11.90%
SVR	1.28%	7.50%	2.56%	1.53%	1.94%
RFE-SVR	1.27%	8.19%	2.25%	1.35%	8.43%
SVR-SH	1.32%	2.30%	1.71%	1.69%	1.86%
WRFE-SVR-SH	1.30%	2.14%	1.68%	2.33%	3.49%

Table 3-5: Ranking of features by the RFE-SVR and WRFE-SVR-SH and the regression vector of the LASSO.

Features ranked by importance by RFE-SVR		Features ranked by the % times included by WRFE-SVR-SH		$\mathbf{x}$ of the LASSO	
1.	MFI_{t-1}	84%	o_{t-2}	0.65	o_{t-1}
2.	$interest_{t-1}$	84%	o_{t-3}	0.14	o_{t-3}
3.	ROC_{t-1}	84%	$Interest_{t-2}$	0.07	SMA^{20}_{t-1}
4.	$Crisis_{t-3}$	73%	VIX_{t-2}	0.06	MFI_{t-1}
5.	$Stock\ today_{t-3}$	63%	$Crisis_{t-1}$	0.02	$interest_{t-1}$
6.	$Candle_{t-1}$	63%	$Interest_{t-1}$		
7.	$Finance_{t-2}$	63%	$Stock_{t-3}$		

Table 3-6: Results for model 2: nominal + specific GIS term.

DA	DJIA	Finance	Stock Today	Stock Today (lagged 3)	Stock
SVR	55%	45%	60%	63%	55%
RFE-SVR	57%	52%	63%	66%	55%
SVR-SH	54%	50%	61%	64%	60%
WRFE-SVR-SH	50%	44%	60%	65%	55%
MAPE	DJIA	Finance	Stock Today	Stock Today (lagged 3)	Stock
SVR	2.19%	4.86%	1.80%	1.28%	1.86%
RFE-SVR	2.77%	2.39%	1.65%	1.29%	2.11%
SVR-SH	1.81%	1.71%	1.64%	1.32%	1.46%
WRFE-SVR-SH	1.83%	2.08%	1.45%	1.33%	1.73%
DA	Economy	Happy	Interest	Crisis	
SVR	47%	62%	64%	48%	
RFE-SVR	39%	56%	73%	53%	
SVR-SH	57%	49%	63%	56%	
WRFE-SVR-SH	52%	44%	74%	58%	
MAPE	Economy	Happy	Interest	Crisis	
SVR	3.23%	5.43%	1.73%	5.18%	
RFE-SVR	6.98%	6.59%	1.40%	2.92%	
SVR-SH	1.56%	2.37%	1.55%	1.89%	
WRFE-SVR-SH	2.34%	2.72%	1.25%	1.49%	

Table 3-7: Results of (Bollen et al., 2010).

	1. (nom)	1. (nom) + Calm	1. (nom) + Calm + Sure
DA	73.3%	86.7%	46.7%
MAPE	1.95%	1.83%	2.13%

3-8 Discussion

Here the results of the previous section are discussed, starting by the linear models, the LASSO and OLS, followed by the performance of the nonlinear model, the SVR, and its short horizoning variant. Then the feature selection methods are reviewed. Finally, the usefulness of the different data types in the methods is discussed.

Linear models
The accuracies of the linear models, as seen in Table 3-4, are definitely lower compared to the nonlinear models. This is very explicit in the nominal scenario where the DA is around 50%, versus the average 63% for the nonlinear models. This observation backs the decision to investigate the nonlinear SVR for stock market prediction. Looking at the LASSO, it handles the feature-selection reasonably well, it provides higher DA for almost all situations.
This does not hold for the LASSO-SH, which gives a very high MAPE in all scenarios. This indicates that the number of data points included in the window, 75, is too low for the LASSO to do the feature selection well. It can also be observed that the OLS-SH performs well when there are few features, such as in model 1 and 4, but its performance heavily deteriorates when the number of features increases.

SVR
The nonlinear regression by SVR has a higher DA and lower MAPE compared to the linear models, which was the motivation for using them. Comparing the results against those of Bollen in Table 3-7, the DA is lower, but the MAPE is also lower. Bollen used the complex Self Organizing Fuzzy Neural Network, which seems to give a higher DA, compared to the SVR. It is however also very sensitive to selecting the correct input, just as the SVR.

SVR-SH
The time-varying method, SVR-SH, works really well; in almost all models a higher accuracy is achieved. For example including individual search terms, model 2, is better handled using the SVR-SH compared to the SVR. When a lot of Google search terms were added, model 3, the MAPE is also significantly lower. This is probably thanks to the fact that the model only focuses on the most recent correlations, thereby achieving a higher accuracy. The SVR tries to fit with many features over a longer horizon in which the correlations have changed. The SVR does not take into account the change in correlation, therefore it achieves a lower accuracy.

RFE-SVR
The feature selection for the SVR works well for models 1, 2, and 4 for which the number of features are low; here a higher DA and a lower MAPE compared to the SVR are achieved.
It is also observed that the feature selection method is not able to retain the performance of model 1 as a minimum for the other models, often a decrease of performance is visible. This means that the MSE of the testing set does not offer enough information to correctly discard data not useful for prediction. This is probably due to overfitting, the SVR can achieve comparable performance by selecting several features which are not useful.
When there are many features, such as in model 6 where there are 47 features in total, the feature selection method does not work well.

WRFE-SVR-SH
The proposed WRFE-SVR-SH works well when the number of features is low, such as in model 1, 2, and 4. In several of these models an increase in accuracy is present; this is a formidable achievement when comparing it against the LASSO-SH which has a terrible performance due to the low amount of data points. As explained in the previous paragraph, the short horizoning bases the feature selection only on recent history, which helps to increase the prediction accuracy. Furthermore it allows insight which features were selected at what time steps, which is shown in Table 3-5. This is insightful when the number of features is low, but not when the number of features becomes high. Then the selected features differ greatly between each time step. This is partially due to the fact that many features contain the same type of information and no element in the method forces the feature selection to choose the same inputs for each time step, when other features contain the same information.
Feature selection for the SVR performs not too well with a large number of features, such as in the models 3, 5 and 6. In these scenarios the ratio of features to data points makes the feature selection too difficult and using the SVR-SH over all features is a better solution.

Usefulness of the data
The VIX index is highly correlated with the Dow Jones closing price and results in a definite increase of accuracy in the linear methods, but not for the non-linear methods. Most google search terms are not that correlated, only 'Stock Today' has a p-value lower than 5%, very similar to that of the Calm index of Bollen. This index clearly does lead to a higher accuracy as visible in model 2.
The technical indicators are useful, they increase prediction accuracy for almost all the methods.

Chapter 4

Conclusions and Recommendations

In this chapter the conclusions of the two subjects are given in Section 4-1. First the subject of finding the covariance of the LASSO is reviewed, followed by the subject of feature selection for the SVR. The conclusions are given as answers to the research questions posed in the introduction. Finally, several recommendations for future research are given in Section 4-2.

4-1 Conclusions

For both Chapter 2 and 3 the conclusions are listed here by answering the research questions set in the introduction, Chapter 1.

4-1-1 Covariance and confidence interval of the LASSO

Does the theoretical probability density function of the LASSO coefficients correspond with the numerical version?
No. A comparison has been made between the practical distribution of $\mathbf{x}$, obtained by a Monte Carlo method, and the mathematical probability density function. The latter was used in literature to show the error of the bootstrap methods. In this report it was shown that a clear difference around $x = 0$ is visible between the practical and mathematical distribution of $\mathbf{x}$.

Which of the existing methods has the best finite sample behavior for finding the covariance and confidence interval of the regression coefficients of the LASSO?
The performance of the methods is very dependent on the used scenario. For simple situations, with low shrinkage and many data points, analytical formulae suffice and are preferred because of their low computational complexity. The approximation given by Osborne clearly outperforms the approximation of Tibshirani and is preferred.

If the shrinkage effect is high or there are few data points, the modified bootstrap method is the best method to estimate the covariance and confidence interval. Residual resampling also has a good performance but has a larger bias for the non-support. Vector resampling is not able to handle high shrinkage effects well and double bootstrapping did not increase the accuracy of the confidence intervals. Fast resampling methods introduced a large error and should thus not be considered. Two new fast resampling methods have been proposed that outcompete the fast residual resampling methods found in literature. However, they are still inferior to the residual resampling.
Remarkably the simplest method works best in the most complex situation: the approximation of Tibshirani is the most accurate method in the scenario of $P > N$.

Can the Unscented Transform be used to recover the covariance and confidence interval of the regression coefficients of the LASSO?
Yes. The proposed framework in this report successfully applies the Unscented Transform to the LASSO to find the covariance and confidence interval. The UT has, to the best of our knowledge, not been applied for finding the covariance of the LASSO or any other regression method in literature. The framework of the Variance UT works and using more sigma points to capture higher order moments increases the accuracy of finding the confidence intervals.
The minimal amount of sigma points is $2N$, reducing the number of sigma points below $2N$ resulted in unpredictable behavior and large errors. For normal situations the UT offers little improvement in accuracy, but for the scenario $P > N$ the 3-fold Variance UT is more accurate than all the other bootstrap variants. This increase of performance is thanks to the structured way of finding the covariance, instead of randomly sampling, as is done in the bootstrap methods.

4-1-2 Feature selection for nonlinear stock market prediction

What is the most suited wrapper feature selection method for fast and accurate feature selection for the SVR?
The RFE was found a suitable feature selection method. On artificial data it yielded highly accurate results when the number of data points was high. The correctness of identifying the support was almost 100% and the MSE was close to the SVR run over only the correct support. On stock market data the RFE-SVR yielded an increase in accuracy, a higher DA and a lower MAPE, compared to the SVR when the number of features to select from was low.

Does a time-varying SVR yield a higher accuracy in predicting stock markets compared to a time-invarying SVR?
Yes. It yielded an increase in accuracy compared to the SVR. A time-varying SVR was applied by using a short horizoning approach, where the SVR was trained at each time step over the last 75 data points. This worked really well, for most models the prediction accuracy was higher. The SVR-SH only focuses on the most recent correlations, thereby achieving a higher accuracy. The SVR tries to fit with many features over a longer horizon in which the correlations have changed. The SVR does not take these changes into account, therefore it achieves a lower accuracy.

What is the most suited wrapper feature selection method for fast and accurate feature selection for time-varying SVR?
A new method was proposed, the WRFE, which combined the recursive feature selection with weighting of features in the kernel to allow usage of less important features. This feature selection method outcompetes the RFE in the case of few data points on the artificial data set; it recovers the support with a higher accuracy and has a lower MSE. On stock market data it offers some improvement in the situation of few features.

4-2 Recommendations

Based on the results presented in this thesis, recommendations for future research on the discussed subjects can be made:

The 3-fold Variance UT has a higher accuracy compared to bootstrap methods in finding the covariance and confidence interval for the LASSO when $P > N$. Only the analytical approximation by Tibshirani achieves a higher accuracy. It would be interesting to investigate if this observation also holds for more complex sparse linear regression methods, for example the Adaptive LASSO (4-1) or the Octagonal Shrinkage and Clustering Algorithm (OSCAR) (4-2) (Zou, 2006) (Bondell and Reich, 2008). Here analytical approximations are harder to impossible to derive, so the increase of performance compared to bootstrap method is even more valuable. Note that $\mathbf{x}^{OLS}$ refers to the OLS solution of the regression problem.

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \sum_{p=1}^{P} \frac{|x_p|}{|x_p^{OLS}|^{\pi}} \tag{4-1}$$

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \Big[\|\mathbf{x}\|_1 + \pi \sum_{p<p_2} max\{x_p, x_{p_2}\} \Big] \tag{4-2}$$

Another axis to investigate this property on is the assumed distribution of data. In the LASSO the explicit assumption of a Gaussian distribution is made $y \sim \mathcal{N}(A\mathbf{x}, \sigma_\varepsilon^2)$. It would be interesting to investigate if this method can also work for a different distribution. The Gaussian distriubtion model is one in the class of Generalized Linear Models (GLMs), which also contains other distributions. For example an exponential distribution could be investigated: $y \sim p_{exp}(-(A\mathbf{x})^{-1}, \zeta)$. (Thomas et al., 2011) have applied the higher order UT to a Rayleigh distribution. In (Sartori, 2011) several bootstrap methods for penalized GLMs are given to compare the UT with.

The WRFE-SVR-SH is a feature selection method, which selects different features at each different time-step. Ideally one would like to see features selected multiple time-steps after each other. As already mentioned in the discussion, this is unfortunately not the case; the selected features differ greatly between the different time-steps. This hinders the interpretability, as it is difficult to see trends when which features is taken into account by the model. A possible solution would be to link the feature selection methods of the different time-steps to each other using an optimization algorithm. This algorithm could then force a trade-off between a very good fit and keeping more consistent behavior of feature selection. A possible disadvantage is that the total method becomes quite cumbersome.

The current comparison illustrates the usefulness of SVR feature selection methods for stock market prediction. In order to further explore the usefulness of this method it is recommended to expand the simulations. Adding information from different social media, especially Twitter, could further show if the feature selection method can capture when this information is useful. Furthermore, a longer prediction horizon could shed more light on the interpretability of the WRFE-SVR-SH, especially after the adjustment of the previous paragraph.

Appendix A

Proofs

A-1 Approximation of the covariance by Tibshirani

Upon rewriting the LASSO objective function as a ridge regression function as in (A-1), the covariance is given by (A-2).

$$\begin{aligned}\min_{\mathbf{x}} \quad ||A\mathbf{x}-\mathbf{y}||_2^2+\lambda||\mathbf{x}||_1 &= \min_{\mathbf{x}} \quad ||A\mathbf{x}-\mathbf{y}||_2^2+||G\mathbf{x}||_2^2 \\ G &= \begin{bmatrix} \sqrt{\frac{\lambda}{|x_1|}} & 0 & .. & 0 \\ 0 & \sqrt{\frac{\lambda}{|x_2|}} & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & \sqrt{\frac{\lambda}{|x_p|}} \end{bmatrix}\end{aligned} \tag{A-1}$$

$$\begin{aligned}\Sigma_{\mathbf{x}} &= \Big(A^TA+\frac{\lambda}{2}W^-\Big)^{-1}A^TA\Big(A^TA+\frac{\lambda}{2}W^-\Big)^{-1}\hat{\sigma}_\varepsilon^2 \\ W^- &= \text{ Moore-Penrose pseudoinverse of } \begin{bmatrix} |x_1| & 0 & .. & 0 \\ 0 & |x_2| & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & |x_p| \end{bmatrix}\end{aligned} \tag{A-2}$$

Proof.

$$J = \|A\mathbf{x}-\mathbf{y}\|_2^2+||G\mathbf{x}||_2^2 = \mathbf{x}^T\Big(A^TA+G^TG\Big)\mathbf{x}-\mathbf{y}^TA\mathbf{x}-\mathbf{x}^TA^T\mathbf{y}+\mathbf{y}^T\mathbf{y}$$

$$\frac{\delta J}{\delta \mathbf{x}} = 2\Big(A^TA+\frac{1}{2}G^TG\Big)\mathbf{x}-2A^T\mathbf{y}=0 \quad \text{(in solution-point)}$$

Assuming a constant matrix G the expression above is untrue, it would become $\frac{\delta J}{\delta \mathbf{x}} = 2\Big(A^TA+G^TG\Big)\mathbf{x}-2A^T\mathbf{y}=0$. But the matrix G is dependent on $\mathbf{x}$, so the factor $1/2$

must be added to match the derivative to that of the LASSO.

$$
\begin{aligned}
\mathbf{x} &= \Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^T\mathbf{y} \\
E[\mathbf{x}] &= \Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^TA\mathbf{x} \\
\Sigma_\mathbf{x} &= E[\mathbf{x}^2] - E[\mathbf{x}]^2 = E\Big[\Big(\Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^T(A\mathbf{x}+\varepsilon)\Big) \\
&\quad \Big(\Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^T(A\mathbf{x}+\varepsilon)\Big)^T\Big] - \Big(\Big(A^TA + \frac{1}{2}G^TG)^{-1}A^TA\mathbf{x}\Big) \\
&\quad \Big(\Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^TA\mathbf{x}\Big)^T \\
\Sigma_\mathbf{x} &= \Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}A^TA\Big(A^TA + \frac{1}{2}G^TG\Big)^{-1}\hat{\sigma}_\varepsilon^2
\end{aligned}
$$

■

A-2 Approximation of the covariance by Osborne

An alternative expression of the Approximation by Tibshirani is given by:

$$
\begin{aligned}
\Sigma_\mathbf{x} &= (A^TA + W_2)^{-1}A^TA(A^TA + W_2)^{-1}\hat{\sigma}_\varepsilon^2 \\
A^TA + W_2 &= A^T\Big(I_N + \frac{(\mathbf{y}-A\mathbf{x})(\mathbf{y}-A\mathbf{x})^T}{||\mathbf{x}||_1||A^T(\mathbf{y}-A\mathbf{x})||_\infty}\Big)A
\end{aligned}
\tag{A-3}
$$

Proof. This solution is derived from the derivative with respect to $\mathbf{x}$ of the LASSO problem; this derivative should be equal to zero for $\mathbf{x}$ being the solution:

$$
J = \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 \quad \rightarrow \quad \frac{\delta J}{\delta \mathbf{x}} = -2A^T(\mathbf{y} - A\mathbf{x}) + \lambda\mathbf{s} = 0
$$

Here $\mathbf{s}$ is of equal size as $\mathbf{x}$ and it has the values:

$$
\begin{aligned}
s_p &= 1 \quad \text{if} \quad x_p > 0 \\
s_p &= [-1, 1] \quad \text{if} \quad x_p = 0 \\
s_p &= -1 \quad \text{if} \quad x_p < 0
\end{aligned}
$$

From this it can be derived that $\lambda = \|2A^T(\mathbf{y} - A\mathbf{x})\|_\infty$, since $||\mathbf{s}||_\infty = 1$:

$$
\|2A^T(\mathbf{y} - A\mathbf{x})\|_\infty = \lambda\|\mathbf{s}\|_\infty \quad \rightarrow \quad \lambda = \|2A^T(\mathbf{y} - A\mathbf{x})\|_\infty
$$

This yields for $\mathbf{s}$:

$$
\mathbf{s} = 2\frac{A^T(\mathbf{y} - A\mathbf{x})}{\lambda} \quad = \quad \frac{A^T(\mathbf{y} - A\mathbf{x})}{\|A^T(\mathbf{y} - A\mathbf{x})\|_\infty}
$$

λ can now be rewritten as follows, since $\mathbf{s}^T\mathbf{x} = ||\mathbf{x}||_1$:

$$\begin{aligned} 2A^T(\mathbf{y} - A\mathbf{x}) =& \lambda\mathbf{s} \quad \rightarrow 2(\mathbf{y} - A\mathbf{x})^T A\mathbf{x} = \lambda\mathbf{s}^T\mathbf{x} \\ & \rightarrow \lambda = 2(\mathbf{y} - A\mathbf{x})^T A\frac{\mathbf{x}}{||\mathbf{x}||_1} \end{aligned}$$

Substituting these expressions back in the solution point the following expression for $\mathbf{x}$ is obtained:

$$\begin{aligned} 2A^T(\mathbf{y} - A\mathbf{x}) = \lambda\mathbf{s} &= \Big(\frac{2A^T(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^T A}{||\mathbf{x}||_1||A^T(\mathbf{y} - A\mathbf{x})||_\infty}\Big)\mathbf{x} \\ \rightarrow A^T\mathbf{y} &= \Big(A^TA + \frac{A^T(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^T A}{||\mathbf{x}||_1||A^T(\mathbf{y} - A\mathbf{x})||_\infty}\Big)\mathbf{x} \\ &= \underbrace{A^T\Big(I_N + \frac{(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^T}{||\mathbf{x}||_1||A^T(\mathbf{y} - A\mathbf{x})||_\infty}\Big)A}_{A^TA+W_2}\mathbf{x} \\ \mathbf{x} &= (A^TA + W_2)^{-1}A^T\mathbf{y} \end{aligned}$$

With this equation the covariance can be derived:

$$\begin{aligned} E[\mathbf{x}] &= (A^TA + W_2)^{-1}A^TA\mathbf{x} \\ \Sigma_\mathbf{x} &= E[\mathbf{x}^2] - E[\mathbf{x}]^2 = E\Big[((A^TA + W_2)^{-1}A^T(A\mathbf{x} + \varepsilon)) \\ &\quad ((A^TA + W_2)^{-1}A^T(A\mathbf{x} + \varepsilon))^T\Big] - ((A^TA + W_2)^{-1}A^TA\mathbf{x}) \\ &\quad ((A^TA + W_2)^{-1}A^TA\mathbf{x})^T \\ \Sigma_\mathbf{x} &= (A^TA + W_2)^{-1}A^TA(A^TA + W_2)^{-1}\hat{\sigma}_\varepsilon^2 \end{aligned}$$

■

A-3 Sandwich formula equals the approximation of Tibshirani

The Sandwich formula is exactly the same as the formula by Tibshirani, except in only recovers the covariance of the non-zero components.

Proof. The Sandwich formula is derived on a general regression problem given as:

$$\min_\mathbf{x} \quad l(\mathbf{x}) + \lambda\sum_{p=1}^{P}\rho(|x_p|) \quad \rightarrow \quad \min_\mathbf{x} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

Here $l(\mathbf{x})$ is the loss function and $\rho(|x_p|)$ is the general shrinkage operator. For the lasso $l(\mathbf{x}) = \|A\mathbf{x} - \mathbf{y}\|_2^2$ and $\rho(|x_p|) = |x_p|$. For this general formula the sandwich formula is proposed only for the non-zero components of $\mathbf{x}$, $\mathbf{x}_{\mathcal{S}}$:

$$\Sigma_{\mathbf{x}_{\mathcal{S}}} = \left\{\frac{\delta^2 l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}^T} + \lambda W_{\mathcal{S}}^-\right\}^{-1} \Sigma_{\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}} \left\{\frac{\delta^2 l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}^T} + \lambda W_{\mathcal{S}}^-\right\}^{-1}$$
$$W^- = \mathrm{diag}\left\{\frac{\delta\rho(|x_1|)}{\delta x_1}/|x_1| \quad \frac{\delta\rho(|x_2|)}{\delta x_2}/|x_2| \quad \dots \quad \frac{\delta\rho(|x_P|)}{\delta x_P}/|x_P|\right\}$$

Here $\mathcal{S}$ refers to the set of support elements, the non-zero elements of $\mathbf{x}$. $W_{\mathcal{S}}$ is the matrix W^- using only the support elements of $\mathbf{x}_{\mathcal{S}}$.
The general covariance formula is written out in for the LASSO:

$$\begin{aligned}
\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}} &= A_{\mathcal{S}}^T A_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} - A_{\mathcal{S}}\mathbf{y} \\
E\left[\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}\right] &= E\left[2A_{\mathcal{S}}^T A_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} - 2A_{\mathcal{S}}^T A_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} - 2A_{\mathcal{S}}^T\epsilon\right] = A_{\mathcal{S}}^T E[\epsilon] = 0 \\
E\left[\left(\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}\right)^2\right] &= 4A_{\mathcal{S}}^T A_{\mathcal{S}}\sigma_\varepsilon^2 \\
\Sigma_{\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}} &= E\left[\left(\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}\right)^2\right] - E\left[\frac{\delta l(\mathbf{x}_{\mathcal{S}})}{\mathbf{x}_{\mathcal{S}}}\right]^2 = 4A_{\mathcal{S}}^T A_{\mathcal{S}}\sigma_\varepsilon^2 \\
\frac{\delta^2 l(\mathbf{x}_{\mathcal{S}})}{\delta\mathbf{x}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}^T} &= 2A_{\mathcal{S}}^T A_{\mathcal{S}} \\
\Sigma_{\mathbf{x}_{\mathcal{S}}} &= \left((A_{\mathcal{S}}^T A_{\mathcal{S}}) + \frac{\lambda}{2}W_{\mathcal{S}}^-\right)^{-1} A_{\mathcal{S}}^T A_{\mathcal{S}} \left((A_{\mathcal{S}}^T A_{\mathcal{S}}) + \frac{\lambda}{2}W_{\mathcal{S}}^-\right)^{-1}\sigma_\varepsilon^2 \\
W^- &= \mathrm{diag}\{1/|x_1| \quad 1/|x_2| \quad \dots \quad 1/|x_p|\}
\end{aligned}$$

Here $A_{\mathcal{S}}$ refers to the columns of the A-matrix corresponding to the support. The resulting formula is identical to the approximation of Tibshirani (2-2), with the only difference being that it defines the covariance only for the support. ■

A-4 LASSO as a soft-thresholding function

Orthonormal A
In the case of an orthonormal A matrix, the LASSO solution can be found using (A-4).

$$x_p = \mathrm{sign}(\mathbf{f}_p^T\mathbf{y})(|\mathbf{f}_p^T\mathbf{y}| - \lambda/2)_+ \tag{A-4}$$

Proof. The objective function of the LASSO can be rewritten as:.

$$J = \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 = \|(I - AA^T)\mathbf{y}\|_2^2 + \sum_{p=1}^{P}(\mathbf{f}_p^T\mathbf{y} - x_p)^2 + \lambda\sum_{p=1}^{P}|x_p|$$

This can be shown by writing out both objective functions, using the fact that $A^TA = I$:

$$\begin{aligned}
J_1 &= \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 = (A\mathbf{x} - \mathbf{y})^T(A\mathbf{x} - \mathbf{y}) + \lambda\|\mathbf{x}\|_1 \\
&= \mathbf{x}^TA^TA\mathbf{x} + \mathbf{y}^T\mathbf{y} - \mathbf{x}^TA^T\mathbf{y} - \mathbf{y}^TA\mathbf{x} + \lambda\|\mathbf{x}\|_1 \\
&= \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - \mathbf{x}^TA^T\mathbf{y} - \mathbf{y}^TA\mathbf{x} + \lambda\|\mathbf{x}\|_1 \\
J_2 &= \|(I - AA^T)\mathbf{y}\|_2^2 + \sum_{p=1}^{P}(\mathbf{f}_p^T\mathbf{y} - x_p)^2 + \lambda\sum_{p=1}^{P}|x_p| \\
&= \|(I - AA^T)\mathbf{y}\|_2^2 + \|A^T\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \\
&= \mathbf{y}^T(I - AA^T)(I - AA^T)\mathbf{y} + \mathbf{y}^TAA^T\mathbf{y} - \mathbf{x}^TA^T\mathbf{y} - \mathbf{y}^TA\mathbf{x} + \mathbf{x}^T\mathbf{x} + \lambda\|\mathbf{x}\|_1 \\
&= \mathbf{y}^T(I - AA^T - AA^T + AA^T + AA^TAA^T)\mathbf{y} + \mathbf{x}^T\mathbf{x} - \mathbf{x}^TA^T\mathbf{y} - \mathbf{y}^TA\mathbf{x} + \lambda\|\mathbf{x}\|_1 \\
&= \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - \mathbf{x}^TA^T\mathbf{y} - \mathbf{y}^TA\mathbf{x} + \lambda\|\mathbf{x}\|_1 \\
&= J_1
\end{aligned}$$

In this scenario the minimization of the LASSO (A-4) can be done element-wise:

$$\min_{x_p} \quad (\mathbf{f}_p^T\mathbf{y} - x_p)^2 + \lambda|x_p|$$

The solution to this minimization problem results in the soft-thresholding function:

$$\begin{aligned}
J &= (\mathbf{f}_p^T\mathbf{y} - x_p)^2 + \lambda|x_p| \\
\frac{dJ}{dx_p} &= 2x_p - 2\mathbf{f}_p^T\mathbf{y} + \lambda\text{sign}(x_p) = 0 \\
&\text{sign}(x_p)\{|x_p| + \lambda/2\} = \mathbf{f}_p^T\mathbf{y} \\
&\text{sign}(x_p) = \text{sign}(\mathbf{f}_p^T\mathbf{y}) \qquad \text{since } |x_p| + \lambda/2 > 0 \\
&|x_p| = \max(|\mathbf{f}_p^T\mathbf{y}| - \lambda/2, 0) \\
&x_p = \text{sign}(\mathbf{f}_p^T\mathbf{y})\max(|\mathbf{f}_p^T\mathbf{y}| - \lambda/2, 0) \\
&x_p = \text{sign}(\mathbf{f}_p^T\mathbf{y})(|\mathbf{f}_p^T\mathbf{y}| - \lambda/2)_+
\end{aligned}$$

Note that the sign function $\text{sign}(x)$ is defined as -1, 0 and 1 for the situations $x < 0$,$x = 0$ and $x > 0$ respectively. The soft-thresholding is built in because $x_p \neq 0$ only in the case that $|\mathbf{f}_p^T\mathbf{y}| - \lambda/2 > 0$. In the region of $-\lambda/2 \leq |\mathbf{f}_p^T\mathbf{y}| \leq \lambda/2$ the found regressor is shrunk to zero. ■

A consisting of ones

In the case of A consisting of only ones and only one regression coefficient the LASSO problem can be rewritten as in (A-5).

$$x = \text{sign}(\bar{\mathbf{y}})\left(|\bar{\mathbf{y}}| - \frac{\lambda}{2N}\right)_+ \tag{A-5}$$

Proof.

$$
\begin{aligned}
J &= \sum_{n=1}^{N}(x-y_n)^2+\lambda|x| \\
&= Nx^2-2x\sum_{n=1}^{N}y_n+\sum_{n=1}^{N}y_n^2+\lambda|x| \\
\frac{dJ}{dx} &= x-\underbrace{1/N\sum_{n=1}^{N}y_n}_{\bar{y}}+\lambda/(2N)\text{sign}(x)=0 \\
& x+\lambda/(2N)\text{sign}(x)=\bar{y} \\
& \text{sign}(x)(|x|+\lambda/(2N))=\bar{y} \\
& \text{sign}(x)=\text{sign}(\bar{y}) \qquad \text{since } |x|+\lambda/(2N)>0 \\
& |x|=\max(|\bar{y}|-\lambda/(2N),0) \\
& x=\text{sign}(\bar{\mathbf{y}})\left(|\bar{\mathbf{y}}|-\frac{\lambda}{2N}\right)_+
\end{aligned}
$$

■

A-5 Test statistic of SRFE

The test statistic (A-6) can be written out to (A-7) (Yang and Ong, 2011).

$$
S^s=\frac{1}{N}\sum_{n=1}^{N}D_{KL}(p(y_n|A),p(y_n|A^s)) \tag{A-6}
$$

$$
S^s=\frac{1}{2N}\sum_{n=1}^{N}\left[\frac{(\hat{y}_n^s-\hat{y}_n)^2}{(\sigma_\varepsilon^s)^2}+\frac{(\sigma_\varepsilon)^2}{(\sigma_\varepsilon^s)^2}+2ln\left(\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon}\right)\right] \tag{A-7}
$$

Proof. The following derivation is partially done using the analytical solver of Maple:

$$
\begin{aligned}
D_{KL}(p(y_n|A), p(y_n|A^s)) &= \int_{-\infty}^{\infty} p(y_n|A) ln\Big(\frac{p(y_n|A)}{p(y_n|A^s)}\Big) dy_n \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big) ln \left(\frac{\frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big)}{\frac{1}{\sqrt{2\pi}\sigma_\varepsilon^s} exp\Big(\frac{-\big(y_n - \hat{y}_n(A^s)\big)^2}{2(\sigma_\varepsilon^s)^2}\Big)} \right) dy_n \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big) ln\Big[\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2} \\
&\qquad + \frac{\big(y_n - \hat{y}_n(A^s)\big)^2}{2(\sigma_\varepsilon^s)^2}\Big)\Big] dy_n \\
&= \underbrace{\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big) dy_n}_{1} \, ln\Big(\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon}\Big) \\
&\quad + \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big) \frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2} dy_n \\
&\quad + \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} exp\Big(\frac{-\big(y_n - \hat{y}_n(A)\big)^2}{2\sigma_\varepsilon^2}\Big) \frac{\big(y_n - \hat{y}_n(A^s)\big)^2}{2(\sigma_\varepsilon^s)^2} dy_n \\
&= ln\Big(\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon}\Big) - \frac{1}{2} + \frac{\big(\hat{y}_n(A^s) - \hat{y}_n(A)\big)^2 + \sigma_\varepsilon^2}{2(\sigma_\varepsilon^s)^2}
\end{aligned}
$$

$$
\begin{aligned}
S^s &= \frac{1}{N}\sum_{n=1}^{N} D_{KL}(p(y_n|A), p(y_n|A^s)) \\
&= \frac{1}{2N}\sum_{n=1}^{N} \Big[\frac{\big(\hat{y}_n(A^s) - \hat{y}_n(A)\big)^2}{(\sigma_\varepsilon^s)^2} + \frac{\sigma_\varepsilon^2}{(\sigma_\varepsilon^s)^2} + 2ln\Big(\frac{\sigma_\varepsilon^s}{\sigma_\varepsilon}\Big) - 1\Big]
\end{aligned}
$$

Since it is used for optimization the static offset -1 can be removed and the expression of (A-7) is obtained. ■

Appendix B

Algorithms

B-1 Parametric bootstrap

Algorithm 16 Parametric bootstrap

```
    **Calculate LASSO solution**
1:    $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    **Calculate $\sigma_\varepsilon$**
2:    $r_n = y_n - \mathbf{a}_n\mathbf{x} \qquad \text{for} \;\; n = 1 \ldots N$
3:    $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:    $\sigma_\varepsilon^2 = 1/(N-1) \sum_{n=1}^{N} (r_n - \bar{r})^2$
    **Iterate to generate set of solutions**
5:  for $i = 1 \to B$ do
6:      for $n = 1 \to N$ do
7:          $\tilde{r}_n^i \leftarrow \mathcal{N}(0, \sigma_\varepsilon^2)$
8:      end for
9:      $\tilde{\mathbf{y}}^i = A\mathbf{x} + \tilde{\mathbf{r}}^i$
10:     $\tilde{\mathbf{x}}^i = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^i\|_2^2 + \lambda\|\mathbf{x}\|_1$
11: end for
    **Calculate the covariance**
12: $\bar{\mathbf{x}} = \frac{1}{B} \sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
13: $\Sigma_{\mathbf{x}} = \frac{1}{B-1} \sum_{i=1}^{B} (\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
    **Calculate confidence intervals**
14: $\mathbf{x}_{(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(84\%)}$
15: $\mathbf{x}_{(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

B-2 Fast residual resampling

Algorithm 17 Fast residual resampling

```
   Calculate LASSO solution
1:   $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
2:   $W^{-} = \text{Moore-Penrose pseudoinverse of } \begin{bmatrix} |x_1| & 0 & .. & 0 \\ 0 & |x_2| & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & |x_p| \end{bmatrix}$
3:   $\Psi = \left(A^T A + \frac{\lambda}{2} W^{-}\right)^{-1} A^T$
   Process the errors
4:   $r_n = y_n - \mathbf{a}_n \mathbf{x} \qquad \text{for} \;\; n = 1 \ldots N$
5:   $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
6:   $r_n = r_n - \bar{r} \qquad \text{for} \;\; n = 1 \ldots N$
   Iterate to generate set of solutions
7: for $i = 1 \to B$ do
8:     for $n = 1 \to N$ do
9:         $index \leftarrow u_{dist}(N)$
10:        $\tilde{r}_n^i = r_{index}$
11:    end for
12:    $\tilde{\mathbf{y}}^i = A\mathbf{x} + \tilde{\mathbf{r}}^i$
13:    $\tilde{\mathbf{x}}^i = \Psi\tilde{\mathbf{y}}^i$
14: end for
   Calculate the covariance
15:    $\bar{\mathbf{x}} = \frac{1}{B} \sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
16:    $\Sigma_{\mathbf{x}} = \frac{1}{B-1} \sum_{i=1}^{B} (\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
   Calculate confidence intervals
17:    $\mathbf{x}_{(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(84\%)}$
18:    $\mathbf{x}_{(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \ldots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

B-3 Proposed fast residual resampling

Algorithm 18 Proposed fast residual resampling

```
    **Calculate LASSO solution**
1:  $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
2:  $\Psi = \left(A^T\left(I_N + \frac{(\mathbf{y}-A\mathbf{x})(\mathbf{y}-A\mathbf{x})^T}{||\mathbf{x}||_1||A^T(\mathbf{y}-A\mathbf{x})||_\infty}\right)A\right)^{-1} A^T$
    **Process the errors**
3:  $r_n = y_n - \mathbf{a}_n\mathbf{x} \qquad \text{for} \;\; n = 1 \dots N$
4:  $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
5:  $r_n = r_n - \bar{r} \qquad \text{for} \;\; n = 1 \dots N$
    **Iterate to generate set of solutions**
6:  **for** $i = 1 \rightarrow B$ **do**
7:      **for** $n = 1 \rightarrow N$ **do**
8:          $index \leftarrow u_{dist}(N)$
9:          $\tilde{r}_n^i = r_{index}$
10:     **end for**
11:     $\tilde{\mathbf{y}}^i = A\mathbf{x} + \tilde{\mathbf{r}}^i$
12:     $\tilde{\mathbf{x}}^i = \Psi\tilde{\mathbf{y}}^i$
13: **end for**
    **Calculate the covariance**
14:     $\bar{\mathbf{x}} = \frac{1}{B} \sum_{i=1}^{B} \tilde{\mathbf{x}}^i$
15:     $\Sigma_{\mathbf{x}} = \frac{1}{B-1} \sum_{i=1}^{B} (\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^i - \bar{\mathbf{x}})^T$
    **Calculate confidence intervals**
16:     $\mathbf{x}_{(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(84\%)}$
17:     $\mathbf{x}_{(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{(16\%)}$
```

B-4 Fast double bootstrap

Algorithm 19 Fast double bootstrap

```
    **Calculate LASSO solution**
 1:     $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
 2:     $W^- = \text{Moore-Penrose pseudoinverse of} \begin{bmatrix} |x_1| & 0 & .. & 0 \\ 0 & |x_2| & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & |x_p| \end{bmatrix}$
 3:     $\Psi = \left(A^TA + \frac{\lambda}{2}W^-\right)^{-1} A^T$
    **Process the errors**
 4:     $r_n = y_n - \mathbf{a}_n\mathbf{x} \quad \text{for} \quad n = 1 \dots N$
 5:     $\bar{r} = 1/N \sum_{n=1}^N r_n$
 6:     $r_n = r_n - \bar{r} \quad \text{for} \quad n = 1 \dots N$
    **First stage bootstrap: iterate to generate set of solutions**
 7: **for** $b = 1 \to B$ **do**
 8:     **for** $n = 1 \to N$ **do**
 9:         $index \leftarrow u_{dist}(N)$
10:         $\tilde{r}_n^b = r_{index}$
11:     **end for**
12:     $\tilde{\mathbf{y}}^b = A\mathbf{x} + \tilde{\mathbf{r}}^b$
13:     $\tilde{\mathbf{x}}^b = \Psi\tilde{\mathbf{y}}^b$
14: **end for**
    **Calculate covariance and roots of the first stage**
15:     $\bar{\mathbf{x}} = \frac{1}{B}\sum_{b=1}^B \tilde{\mathbf{x}}^b$
16:     $\Sigma_{\mathbf{x}} = \frac{1}{B-1}\sum_{b=1}^B (\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})^T$
17:     $\sigma_{x_p} = \frac{1}{B-1}\sum_{b=1}^B (\tilde{x}_p^b - \bar{x}_p)(\tilde{x}_p^b - \bar{x}_p)^T \quad \text{for} \quad p = 1 \dots P$
18:     $\tilde{q}_p^b = (\tilde{x}_p^b - x_p)/\sigma_{x_p} \quad \text{for} \quad b = 1 \dots B \quad \text{and} \quad p = 1 \dots P$
    **Find roots in the second stage for each sample of the first stage**
19: **for** $b = 1 \to B$ **do**
20:     **for** $d = 1 \to D$ **do**
21:         **for** $n = 1 \to N$ **do**
22:             $index \leftarrow u_{dist}(N)$
23:             $\breve{r}_n^{b,d} = \tilde{r}_{index}^b$
24:         **end for**
25:         $\breve{\mathbf{y}}^{b,d} = \tilde{\mathbf{y}}^b + \breve{\mathbf{r}}^{b,d}$
26:         $\breve{\mathbf{x}}^{b,d} = \Psi\breve{\mathbf{y}}^{b,d}$
27:     **end for**
28:     $\overline{\tilde{\mathbf{x}}^b} = \frac{1}{D}\sum_{d=1}^D \breve{\mathbf{x}}^{b,d}$
29:     $\tilde{\sigma}_{x_p^b} = \frac{1}{D-1}\sum_{d=1}^D (\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})(\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})^T \quad \text{for} \quad p = 1 \dots P$
30:     $\breve{q}_p^{b,d} = (\breve{x}_p^{b,d} - \tilde{x}_p^b)/\tilde{\sigma}_{x_p^b} \quad \text{for} \quad d = 1 \dots D \quad \text{and} \quad p = 1 \dots P$
31:     $\tilde{g}_p^b = \#\{\breve{q}_p^{b,d} \leq \tilde{q}_p^b\}/D \quad \text{for} \quad p = 1 \dots P$
32: **end for**
    **Calculate confidence intervals**
33:     $\tilde{h}_{(84\%)} = \{\tilde{g}_p^1 \dots \tilde{g}_p^B\}_{(84\%)}$
34:     $x_{p(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{(84\%)}}$
35:     $\tilde{h}_{(16\%)} = \{\tilde{g}_p^1 \dots \tilde{g}_p^B\}_{(16\%)}$
36:     $x_{p(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{(16\%)}}$
```

B-5 Proposed fast double bootstrap

Algorithm 20 Proposed fast double bootstrap

```
    **Calculate LASSO solution**
1:  $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
2:  $\Psi = \left(A^T\left(I_N + \frac{(\mathbf{y}-A\mathbf{x})(\mathbf{y}-A\mathbf{x})^T}{||\mathbf{x}||_1||A^T(\mathbf{y}-A\mathbf{x})||_\infty}\right)A\right)^{-1} A^T$
    **Process the errors**
3:  $r_n = y_n - \mathbf{a}_n\mathbf{x}$   for $n = 1 \dots N$
4:  $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
5:  $r_n = r_n - \bar{r}$   for $n = 1 \dots N$
    **First stage bootstrap: iterate to generate set of solutions**
6:  **for** $b = 1 \to B$ **do**
7:      **for** $n = 1 \to N$ **do**
8:          $index \leftarrow u_{dist}(N)$
9:          $\tilde{r}_n^b = r_{index}$
10:     **end for**
11:     $\tilde{\mathbf{y}}^b = A + \tilde{\mathbf{r}}^b$
12:     $\tilde{\mathbf{x}}^b = \Psi\tilde{\mathbf{y}}^b$
13:     $\Psi^b = \left(A^T\left(I_n + \frac{(\tilde{\mathbf{y}}^b-A\tilde{\mathbf{x}}^b)(\tilde{\mathbf{y}}^b-A\tilde{\mathbf{x}}^b)^T}{||\tilde{\mathbf{x}}^b||_1||A^T(\tilde{\mathbf{y}}^b-A\tilde{\mathbf{x}}^b)||_\infty}\right)A\right)^{-1} A^T$
14: **end for**
    **Calculate covariance and roots of the first stage**
15:  $\bar{\mathbf{x}} = \frac{1}{B}\sum_{b=1}^{B}\tilde{\mathbf{x}}^b$
16:  $\Sigma_{\mathbf{x}} = \frac{1}{B-1}\sum_{b=1}^{B}(\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^b - \bar{\mathbf{x}})^T$
17:  $\sigma_{x_p} = \frac{1}{B-1}\sum_{b=1}^{B}(\tilde{x}_p^b - \bar{x}_p)(\tilde{x}_p^b - \bar{x}_p)^T$   for $p = 1 \dots P$
18:  $\tilde{q}_p^b = (\tilde{x}_p^b - x_p)/\sigma_{x_p}$   for $b = 1 \dots B$ and $p = 1 \dots P$
    **Find roots in the second stage for each sample of the first stage**
19: **for** $b = 1 \to B$ **do**
20:     **for** $d = 1 \to D$ **do**
21:         **for** $n = 1 \to N$ **do**
22:             $index \leftarrow u_{dist}(N)$
23:             $\breve{r}_n^{b,d} = \tilde{r}_{index}^b$
24:         **end for**
25:         $\breve{\mathbf{y}}^{b,d} = \tilde{\mathbf{y}}^b + \breve{\mathbf{r}}^{b,d}$
26:         $\breve{\mathbf{x}}^{b,d} = \Psi^b\breve{\mathbf{y}}^{b,d}$
27:     **end for**
28:     $\overline{\breve{\mathbf{x}}^b} = \frac{1}{D}\sum_{d=1}^{D}\breve{\mathbf{x}}^{b,d}$
29:     $\tilde{\sigma}_{x_p^b} = \frac{1}{D-1}\sum_{d=1}^{D}(\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})(\breve{x}_p^{b,d} - \overline{\tilde{x}_p^b})^T$   for $p = 1 \dots P$
30:     $\breve{q}_p^{b,d} = (\breve{x}_p^{b,d} - \tilde{x}_p^b)/\tilde{\sigma}_{x_p^b}$   for $d = 1 \dots D$ and $p = 1 \dots P$
31:     $\tilde{g}_p^b = \#\{\breve{q}_p^{b,d} \leq \tilde{q}_p^b\}/D$   for $p = 1 \dots P$
32: **end for**
    **Calculate confidence intervals:**
33:  $\tilde{h}_{(84\%)} = \{\tilde{g}_p^1 \dots \tilde{g}_p^B\}_{(84\%)}$
34:  $x_{p(16\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{(84\%)}}$
35:  $\tilde{h}_{(16\%)} = \{\tilde{g}_p^1 \dots \tilde{g}_p^B\}_{(16\%)}$
36:  $x_{p(84\%)} = 2\mathbf{x} - \{\tilde{\mathbf{x}}^1 \dots \tilde{\mathbf{x}}^B\}_{\tilde{h}_{(16\%)}}$
```

B-6 3-fold UT Variance

Algorithm 21 3-fold Variance UT

```
    **Calculate LASSO solution**
1:      $\mathbf{x} = \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$
    **Calculate $\Sigma_{\mathbf{y}}$**
2:      $r_n = y_n - \mathbf{a}_n\mathbf{x}$    for $n = 1 \dots N$
3:      $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
4:      $\sigma_\varepsilon^2 = 1/(N-1) \sum_{n=1}^{N}(r_n - \bar{r})^2$
5:      $\Sigma_{\mathbf{y}} = I_N \sigma_\varepsilon^2$
    **Calculate sigma point set and weights**
6:      $\tilde{\mathbf{y}}_0 = \mathbf{y}$    $w_0 = 1 - 2N(5.429 \cdot 10^{-4} + 0.0308 + 0.24012)$
7:      $\tilde{\mathbf{y}}^j = \mathbf{y} + \sqrt{3 \cdot 4.6886}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_j = 5.429 \cdot 10^{-4}$    for $j = 1 \dots N$
8:      $\tilde{\mathbf{y}}^{j+N} = \mathbf{y} - \sqrt{3 \cdot 4.6886}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+N} = 5.429 \cdot 10^{-4}$    for $j = 1 \dots N$
9:      $\tilde{\mathbf{y}}^{j+2N} = \mathbf{y} + \sqrt{3 \cdot 1.8672}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+2N} = 0.0308$    for $j = 1 \dots N$
10:     $\tilde{\mathbf{y}}^{j+3N} = \mathbf{y} - \sqrt{3 \cdot 1.8672}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+3N} = 0.0308$    for $j = 1 \dots N$
11:     $\tilde{\mathbf{y}}^{j+4N} = \mathbf{y} + \sqrt{3 \cdot 0.4442}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+4N} = 0.24012$    for $j = 1 \dots N$
12:     $\tilde{\mathbf{y}}^{j+5N} = \mathbf{y} - \sqrt{3 \cdot 0.4442}\left(\sqrt{\Sigma_{\mathbf{y}}}\right)_j$    $w_{j+5N} = 0.24012$    for $j = 1 \dots N$
    **Transform the sigma points**
13:     $\tilde{\mathbf{x}}^j \leftarrow \text{argmin}_{\mathbf{x}} \|A\mathbf{x} - \tilde{\mathbf{y}}^j\|_2^2 + \lambda\|\mathbf{x}\|_1$    for $j = 0 \dots 6N$
    **Calculate covariance**
14:     $\bar{\mathbf{x}} = \sum_{j=0}^{6N} w_j \, \tilde{\mathbf{x}}^j$
15:     $\Sigma_{\mathbf{x}} = \sum_{j=0}^{6N} w_j(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
16: **if** $\Sigma_{\mathbf{x}} \neq$ positive semi-definite **then**
17:     $\Sigma_{\mathbf{x}} = \sum_{j=1}^{6N} w_j(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})(\tilde{\mathbf{x}}^j - \bar{\mathbf{x}})^T$
18: **end if**
```

B-7 Recursive Feature Elimination (RFE-SVR) (exact)

Algorithm 22 Recursive Feature Elimination (RFE-SVR) (exact)

```
    Initialize
1:  $i = 1$, $\mathcal{S}^1 = \{1, 2, \dots P\}$, $A^1 = A$
2:  Tune SVR using $A^1$: find parameters $C^1 \& \gamma^1$
3:  Calculate 10-fold CV $MSE^1$ using $A_1, C^1 \& \gamma^1$.
4:  while $|\mathcal{S}^i| > 0$ do
5:      for each $s \in \mathcal{S}^i$ do
6:          Set $A_t = A^i$
7:          Remove column $s$ of $A_t$
8:          Tune SVR using $A_t$: find parameters $C_t \& \gamma_t$
9:          Calculate 10-fold CV $MSE_t^s$ using $A_t, C_t \& \gamma_t$
10:     end for
11:     Find the index $s_{min}$ belonging to the minimum of set $\{MSE_t^1, \dots, MSE_t^{|\mathcal{S}^i|}\}$
12:     $S^{i+1} = S^i \setminus s_{min}$
13:     $A^{i+1} = A^i$
14:     Remove row $s_{min}$ from $A^{i+1}$
15:     Tune SVR using $A^{i+1}$: find parameters $C^{i+1} \& \gamma^{i+1}$
16:     Calculate 10-fold CV $MSE^{i+1}$ using $A^{i+1}, C^{i+1} \& \gamma^{i+1}$
17:     i = i+1
18: end while
    Find support delivering minimal MSE
19: Find the index $s_{min}$ belonging to the minimum of set $\{MSE^1, \dots, MSE^{|\mathcal{S}^i|}\}$
20: Support is found: $S^{s_{min}}$
```

B-8 Statistical Recursive Feature Elimination (SRFE-SVR)

Algorithm 23 Statistical Recursive Feature Elimination (SRFE-SVR)

```
    **Initialize**
1:  $i = 1$, $\mathcal{S}^1 = \{1, 2, \dots P\}$, $A^1 = A$
2:  Tune SVR using $A^1$: find parameters $C^1 \& \gamma^1$
3:  Calculate 10-fold CV $MSE^1$ using $A_1, C^1 \& \gamma^1$.
4:  while $|\mathcal{S}^i| > 0$ do
5:      for each $s \in \mathcal{S}^i$ do
6:          Set $A_t = A^i$
7:          Replace column $s$ of $A_t$ with zeros
8:          Calculate the 10-fold CV $MSE_t, \hat{\mathbf{y}}_t, \mathbf{e}_t$ using $A_t, C^i \& \gamma^i$
9:          Calculate $S^s$ (3-26) by comparing $(MSE_t, \hat{\mathbf{y}}_t \& \mathbf{e}_t)$ with $(MSE^{i+1}, \hat{\mathbf{y}}^{i+1} \& \mathbf{e}^{i+1})$
10:     end for
11:     Find the index $s_{min}$ belonging to the minimum of set $\{S^1, \dots, S^{|\mathcal{S}^i|}\}$
12:     $\mathcal{S}^{i+1} = \mathcal{S}^i \setminus s_{min}$
13:     $A^{i+1} = A^i$
14:     Remove row $s_{min}$ from $A^{i+1}$
15:     Tune SVR using $A^{i+1}$: find parameters $C^{i+1} \& \gamma^{i+1}$
16:     Calculate 10-fold CV $MSE^{i+1}, \hat{\mathbf{y}}^{i+1} \& \mathbf{e}^{i+1}$ using $A^{i+1}, C^{i+1} \& \gamma^{i+1}$
17:     i = i+1
18: end while
    **Find support delivering minimal MSE**
19: Find the index $s_{min}$ belonging to the minimum of set $\{MSE^1, \dots, MSE^{|\mathcal{S}^i|}\}$
20: Support is found: $\mathcal{S} = \mathcal{S}^{s_{min}}$
```

Appendix C

Comprehensive results

C-1 Comparison nominal scenario

Table C-1: Approximations and accuracies per regression coefficient for the simulation with parameters of Table 2-7

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$\hat{\mathbf{x}}$	0.75	2.00	0.00	0.00	0.00	10.00	-3.00	1.49	0.00	-0.65
$\boldsymbol{\sigma}_{\mathbf{x}}^{MC}$	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
Approximation of Tibshirani: bias [%]	-4%	-3%	-7%	-7%	-8%	-3%	-3%	-3%	-7%	-4%
ce[%]	-2%	2%	0%	1%	2%	5%	3%	1%	5%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.021	0.025	0.026	0.012	0.012	0.012	0.023	0.012
Approximation of Osborne: bias [%]	-3%	-3%	-1%	-1%	-1%	-3%	-2%	-3%	-1%	-3%
ce[%]	-2%	2%	-1%	0%	0%	5%	3%	0%	4%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.011	0.012
Vector resampling: bias [%]	-1%	-1%	-1%	0%	-1%	-1%	-1%	-1%	-1%	-1%
ce[%]	-3%	2%	-3%	0%	1%	2%	1%	1%	3%	2%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.017	0.016	0.015	0.016	0.015	0.017	0.017	0.016	0.016	0.016
Residual resampling B = 100: bias [%]	-3%	-3%	-2%	-3%	-3%	-3%	-3%	-4%	-2%	-4%
ce[%]	-4%	3%	0%	0%	4%	6%	4%	1%	6%	8%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.019	0.019	0.019	0.020	0.019	0.019	0.020	0.019	0.019	0.019
Residual resampling B = 1000: bias [%]	-3%	-3%	-3%	-2%	-2%	-3%	-3%	-3%	-3%	-3%
ce[%]	-2%	1%	0%	2%	4%	4%	0%	1%	5%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.013	0.012	0.013	0.013	0.013	0.013	0.013	0.013	0.012	0.013
Parametric bootstrap: bias [%]	-3%	-3%	2%	-2%	-3%	-3%	-2%	-3%	-2%	-3%
ce[%]	-3%	1%	1%	0%	2%	5%	2%	-1%	6%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.014	0.013	0.013	0.013	0.013	0.013	0.014	0.012	0.012	0.013
Fast residual bootstrapping: bias [%]	-4%	-2%	4%	6%	1%	-3%	-2%	-3%	0%	-2%
ce[%]	-2%	2%	-3%	-1%	1%	6%	2%	-2%	3%	3%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.014	0.017	0.211	0.140	0.085	0.014	0.015	0.022	0.041	0.013
Proposed fast residual Bootstrapping: bias [%]	-3%	-3%	-1%	-1%	-2%	-3%	-3%	-4%	-1%	-4%
ce[%]	-2%	1%	0%	0%	4%	6%	1%	0%	4%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.013	0.013	0.013	0.013	0.013	0.013	0.014	0.013	0.012	0.013
Double bootstrap: bias [%]	-3%	-2%	-3%	-2%	-3%	-3%	-3%	-4%	-3%	-4%
ce[%]	2%	3%	3%	5%	8%	4%	3%	4%	7%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.022	0.017	0.020	0.020	0.020	0.020	0.019	0.020	0.018	0.019
Fast double bootstrap: bias [%]	-4%	-2%	3%	6%	0%	-3%	-3%	-3%	0%	-2%
ce[%]	-1%	2%	2%	2%	3%	5%	2%	1%	3%	6%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.019	0.023	0.215	0.135	0.084	0.019	0.019	0.026	0.046	0.018
Proposed fast double bootstrap: bias [%]	-3%	-4%	-2%	0%	-2%	-4%	-2%	-4%	-2%	-3%
ce[%]	0%	4%	2%	3%	6%	5%	2%	4%	5%	6%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.020	0.021	0.019	0.019	0.019	0.019	0.019	0.018	0.020	0.020
Modified bootstrap: bias [%]	-3%	-3%	-3%	-2%	-3%	-3%	-2%	-3%	-3%	-3%
ce[%]	-2%	1%	1%	2%	2%	6%	2%	0%	5%	3%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.013	0.013	0.013	0.013	0.013	0.013	0.014	0.012	0.013	0.013
Scalar UT: bias [%]	-23%	-25%	-23%	-20%	-26%	-24%	-26%	-25%	-20%	-18%
ce[%]	18%	19%	18%	19%	26%	21%	19%	20%	17%	18%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.127	0.124	0.126	0.131	0.121	0.124	0.122	0.131	0.131	0.137
2-fold Scalar UT: bias [%]	-23%	-25%	-23%	-20%	-26%	-24%	-26%	-25%	-20%	-18%
ce[%]	18%	19%	19%	19%	26%	21%	19%	20%	18%	18%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.127	0.124	0.125	0.131	0.121	0.124	0.122	0.131	0.131	0.137
3-fold Scalar UT: bias [%]	-23%	-25%	-23%	-20%	-26%	-24%	-26%	-25%	-20%	-18%
ce[%]	18%	19%	19%	19%	26%	21%	19%	20%	18%	18%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.127	0.124	0.126	0.131	0.121	0.124	0.122	0.131	0.131	0.137
Variance UT: bias [%]	-3%	-3%	-3%	-2%	-3%	-3%	-2%	-3%	-2%	-3%
ce[%]	-3%	2%	4%	7%	7%	5%	2%	0%	10%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.017	0.015	0.019	0.012	0.012	0.012	0.015	0.012
2-fold Variance UT: bias [%]	-3%	-3%	-3%	-3%	-3%	-3%	-2%	-3%	-3%	-3%
ce[%]	-3%	2%	4%	7%	7%	5%	2%	0%	10%	5%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.016	0.019	0.015	0.012	0.012	0.012	0.015	0.012
3-fold Variance UT: bias [%]	-3%	-3%	-3%	-3%	-3%	-3%	-2%	-3%	-3%	-3%
ce[%]	-3%	2%	3%	7%	6%	5%	3%	0%	10%	4%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.021	0.017	0.014	0.012	0.012	0.012	0.020	0.012

C-2 Comparison high shrinkage scenario

Table C-2: Approximations and accuracies per regression coefficient for the simulation with parameters of Table 2-9

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$\hat{\mathbf{x}}$	0.63	1.88	0.00	0.00	0.00	9.88	-2.88	1.38	0.00	-0.53
$\boldsymbol{\sigma}_{\mathbf{x}}^{MC}$	0.22	0.22	0.14	0.14	0.14	0.22	0.22	0.22	0.14	0.22
Approximation of Tibshirani: bias [%]	-20%	-8%	12%	9%	9%	-3%	-6%	-10%	4%	-21%
ce[%]	11%	10%	3%	-3%	3%	0%	4%	3%	1%	10%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.023	0.011	0.068	0.071	0.069	0.011	0.011	0.011	0.071	0.025
Approximation of Osborne: bias [%]	-1%	-2%	59%	60%	61%	-2%	-2%	-2%	59%	-1%
ce[%]	2%	5%	-17%	-18%	-22%	0%	4%	-1%	-19%	3%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.011	0.012
Vector resampling: bias [%]	-2%	0%	17%	16%	18%	0%	0%	-1%	17%	-3%
ce[%]	3%	4%	-12%	-14%	-16%	-2%	2%	-1%	-12%	1%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.018	0.016	0.028	0.027	0.027	0.015	0.016	0.016	0.026	0.021
Residual resampling B = 100: bias [%]	-6%	-2%	5%	4%	5%	-3%	-3%	-2%	6%	-7%
ce[%]	10%	9%	4%	0%	5%	5%	4%	6%	1%	17%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.024	0.020	0.025	0.026	0.025	0.019	0.021	0.020	0.025	0.028
Residual resampling B = 1000: bias [%]	-6%	-2%	6%	5%	6%	-2%	-2%	-2%	5%	-7%
ce[%]	10%	9%	3%	-3%	5%	4%	6%	7%	1%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.020	0.013	0.023	0.022	0.020	0.013	0.012	0.013	0.021	0.024
Parametric bootstrap: bias [%]	-5%	-2%	6%	5%	6%	-2%	-2%	-2%	5%	-7%
ce[%]	10%	10%	3%	-3%	4%	5%	5%	6%	1%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.021	0.013	0.022	0.021	0.020	0.013	0.012	0.013	0.021	0.024
Fast residual resampling: bias [%]	35%	33%	201%	292%	1227%	37%	32%	47%	569%	326%
ce[%]	12%	7%	-3%	1%	3%	5%	4%	6%	2%	-11%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.883	0.587	0.795	2.847	11.893	0.536	0.413	0.705	5.440	7.075
Proposed fast residual resampling: bias [%]	-1%	-2%	59%	59%	60%	-2%	-2%	-2%	59%	-1%
ce[%]	8%	10%	3%	-3%	3%	4%	5%	6%	0%	10%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.014	0.013	0.013	0.013	0.012	0.013	0.013	0.013	0.013	0.013
Double bootstrap: bias [%]	-5%	-2%	5%	5%	4%	-2%	-2%	-2%	4%	-8%
ce[%]	26%	23%	15%	36%	20%	23%	22%	24%	35%	35%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.026	0.021	0.026	0.027	0.025	0.019	0.020	0.021	0.026	0.027
Fast double bootstrap: bias [%]	36%	32%	194%	288%	1225%	36%	32%	46%	599%	329%
ce[%]	25%	25%	12%	15%	15%	25%	24%	28%	18%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.930	0.578	0.749	2.802	12.175	0.522	0.408	0.705	6.160	7.048
Proposed fast double bootstrap: bias [%]	-2%	-3%	60%	59%	60%	-2%	-2%	-2%	59%	-1%
ce[%]	23%	24%	24%	26%	29%	23%	28%	25%	26%	29%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.019	0.019	0.019	0.019	0.019	0.020	0.021	0.020	0.021	0.018
Modified bootstrap: bias [%]	-6%	-2%	1%	2%	2%	-2%	-2%	-1%	1%	-8%
ce[%]	10%	9%	4%	-2%	3%	6%	6%	6%	0%	18%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.023	0.013	0.020	0.020	0.018	0.012	0.013	0.013	0.020	0.030
Scalar UT: bias [%]	-27%	-14%	-4%	-12%	0%	-20%	-22%	-29%	2%	-18%
ce[%]	22%	18%	11%	8%	9%	16%	20%	22%	8%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.114	0.149	0.121	0.105	0.121	0.134	0.131	0.129	0.119	0.128
2-fold Scalar UT: bias [%]	-26%	-14%	-3%	-11%	2%	-20%	-22%	-29%	3%	-18%
ce[%]	22%	18%	9%	6%	6%	15%	20%	22%	2%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.116	0.149	0.119	0.104	0.119	0.134	0.130	0.128	0.117	0.129
3-fold Scalar UT: bias [%]	-26%	-14%	-3%	-11%	2%	-20%	-22%	-29%	3%	-18%
ce[%]	22%	18%	8%	5%	4%	15%	20%	22%	2%	16%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.115	0.149	0.120	0.104	0.118	0.134	0.130	0.128	0.117	0.128
Variance UT: bias [%]	-1%	-2%	-7%	-6%	-2%	-2%	-2%	-1%	0%	-1%
ce[%]	0%	6%	32%	28%	26%	-1%	3%	-2%	27%	2%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.015	0.012	0.101	0.102	0.099	0.012	0.012	0.012	0.098	0.021
2-fold Variance UT: bias [%]	-1%	-2%	-6%	-6%	-2%	-2%	-2%	-1%	0%	-1%
ce[%]	0%	6%	18%	14%	16%	-2%	4%	-2%	16%	2%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.015	0.012	0.100	0.100	0.098	0.012	0.012	0.012	0.097	0.021
3-fold Variance UT: bias [%]	-1%	-2%	-6%	-6%	-1%	-2%	-1%	-1%	1%	-1%
ce[%]	0%	6%	15%	9%	12%	-2%	4%	-2%	14%	2%
$\sigma(\boldsymbol{\sigma}_{\mathbf{x}})$	0.015	0.012	0.100	0.100	0.098	0.012	0.012	0.012	0.097	0.021

C-3 Comparison $P > N$ scenario

Table C-3: Approximations and accuracies per regression coefficient for the simulation with parameters of Table 2-11

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	$\cdots$	x_{25}
$\hat{\mathbf{x}}$	0.72	1.97	0.00	0.00	0.00	9.97	-2.97	1.47	0.00	-0.62	...	0.00
$\sigma_{\mathbf{x}}^{MC}$	0.05	0.05	0.02	0.02	0.02	0.05	0.04	0.05	0.02	0.05	...	0.02
Approximation of Tibshirani: bias [%]	-26%	-18%	16%	4%	6%	-12%	-18%	-22%	12%	-29%	...	17%
ce[%]	27%	32%	17%	21%	19%	28%	32%	29%	16%	26%	...	17%
$\sigma(\sigma_{\mathbf{x}})$	0.009	0.012	0.023	0.019	0.020	0.015	0.011	0.010	0.020	0.009	...	0.024
Approximation of Osborne: bias [%]	-26%	-28%	48%	48%	46%	-28%	-26%	-26%	49%	-26%	...	41%
ce[%]	30%	37%	-6%	-4%	-8%	35%	36%	31%	-9%	29%	...	-7%
$\sigma(\sigma_{\mathbf{x}})$	0.014	0.012	0.013	0.014	0.013	0.015	0.013	0.015	0.014	0.014	...	0.014
Vector resampling: bias [%]	650%	1201%	872%	864%	891%	1652%	1421%	1010%	898%	617%	...	879%
ce[%]	6%	0%	18%	15%	14%	1%	0%	3%	15%	5%	...	13%
$\sigma(\sigma_{\mathbf{x}})$	0.082	0.123	0.103	0.093	0.095	0.316	0.201	0.092	0.096	0.091	...	0.098
Residual resampling B = 100: bias [%]	-35%	-33%	-38%	-36%	-39%	-34%	-33%	-36%	-42%	-34%	...	-38%
ce[%]	39%	37%	46%	44%	46%	36%	40%	40%	44%	42%	...	42%
$\sigma(\sigma_{\mathbf{x}})$	0.011	0.010	0.011	0.011	0.011	0.010	0.011	0.010	0.009	0.011	...	0.010
Residual resampling B = 1000: bias [%]	-35%	-33%	-37%	-36%	-39%	-34%	-33%	-36%	-41%	-34%	...	-38%
ce[%]	38%	36%	46%	46%	46%	34%	40%	39%	44%	42%	...	42%
$\sigma(\sigma_{\mathbf{x}})$	0.011	0.010	0.010	0.011	0.011	0.010	0.011	0.010	0.009	0.010	...	0.010
Parametric bootstrap: bias [%]	-34%	-31%	-35%	-34%	-37%	-32%	-31%	-35%	-39%	-32%	...	-36%
ce[%]	39%	36%	45%	47%	45%	33%	39%	38%	44%	41%	...	40%
$\sigma(\sigma_{\mathbf{x}})$	0.011	0.010	0.010	0.011	0.011	0.010	0.011	0.010	0.009	0.011	...	0.010
Fast residual resampling: bias [%]	358%	510%	1359%	775%	378%	409%	389%	288%	430%	540%	...	425%
ce[%]	43%	44%	46%	43%	48%	49%	51%	43%	46%	48%	...	46%
$\sigma(\sigma_{\mathbf{x}})$	0.963	2.352	2.906	1.286	0.292	1.487	0.904	0.701	0.343	1.502	...	0.372
Proposed fast residual resampling: bias [%]	-28%	-30%	44%	44%	42%	-30%	-28%	-27%	45%	-28%	...	37%
ce[%]	64%	66%	64%	64%	64%	68%	64%	63%	63%	66%	...	64%
$\sigma(\sigma_{\mathbf{x}})$	0.014	0.012	0.013	0.014	0.013	0.015	0.013	0.015	0.014	0.014	...	0.013
Double bootstrap: bias [%]	-35%	-33%	-37%	-38%	-39%	-34%	-33%	-36%	-41%	-35%	...	-39%
ce[%]	42%	42%	47%	52%	51%	44%	42%	43%	20%	45%	...	25%
$\sigma(\sigma_{\mathbf{x}})$	0.011	0.010	0.011	0.011	0.011	0.010	0.011	0.010	0.009	0.011	...	0.010
Fast double bootstrap: bias [%]	352%	485%	1301%	756%	375%	393%	392%	292%	439%	525%	...	417%
ce[%]	56%	56%	55%	53%	58%	59%	58%	61%	57%	54%	...	54%
$\sigma(\sigma_{\mathbf{x}})$	0.939	2.150	2.672	1.203	0.283	1.365	0.913	0.722	0.354	1.398	...	0.349
Proposed fast double bootstrap: bias [%]	-27%	-30%	45%	44%	41%	-30%	-28%	-28%	45%	-28%	...	37%
ce[%]	67%	66%	67%	65%	67%	68%	68%	67%	66%	66%	...	65%
$\sigma(\sigma_{\mathbf{x}})$	0.015	0.012	0.013	0.014	0.013	0.014	0.013	0.015	0.014	0.014	...	0.014
Modified bootstrap: bias [%]	58%	68%	81%	72%	78%	65%	68%	59%	76%	59%	...	80%
ce[%]	6%	7%	9%	18%	10%	4%	6%	8%	11%	12%	...	5%
$\sigma(\sigma_{\mathbf{x}})$	0.032	0.046	0.029	0.027	0.030	0.040	0.042	0.036	0.028	0.032	...	0.031
Scalar UT: bias [%]	-36%	-39%	-37%	-38%	-35%	-37%	-36%	-41%	-39%	-36%	...	-34%
ce[%]	39%	44%	26%	35%	33%	40%	46%	39%	28%	40%	...	30%
$\sigma(\sigma_{\mathbf{x}})$	0.021	0.020	0.015	0.014	0.017	0.021	0.021	0.019	0.015	0.022	...	0.015
2-fold Scalar UT: bias [%]	-33%	-35%	-29%	-31%	-28%	-32%	-31%	-37%	-32%	-32%	...	-27%
ce[%]	36%	43%	16%	24%	20%	38%	45%	37%	18%	37%	...	20%
$\sigma(\sigma_{\mathbf{x}})$	0.021	0.020	0.016	0.015	0.018	0.021	0.021	0.019	0.016	0.022	...	0.016
3-fold Scalar UT: bias [%]	-32%	-34%	-29%	-31%	-28%	-32%	-31%	-37%	-32%	-32%	...	-27%
ce[%]	36%	43%	14%	24%	18%	38%	44%	37%	15%	36%	...	15%
$\sigma(\sigma_{\mathbf{x}})$	0.021	0.020	0.016	0.015	0.018	0.021	0.021	0.018	0.016	0.022	...	0.016
Variance UT: bias [%]	-24%	-23%	-24%	-19%	-24%	-22%	-22%	-26%	-32%	-22%	...	-26%
ce[%]	34%	36%	35%	24%	35%	35%	39%	34%	30%	29%	...	29%
$\sigma(\sigma_{\mathbf{x}})$	0.014	0.012	0.017	0.018	0.016	0.012	0.013	0.012	0.015	0.014	...	0.017
2-fold Variance UT: bias [%]	-24%	-22%	-23%	-18%	-23%	-22%	-22%	-26%	-31%	-22%	...	-25%
ce[%]	34%	37%	23%	11%	23%	34%	40%	34%	16%	29%	...	20%
$\sigma(\sigma_{\mathbf{x}})$	0.014	0.013	0.017	0.018	0.016	0.012	0.013	0.012	0.015	0.015	...	0.017
3-fold Variance UT: bias [%]	-24%	-22%	-23%	-17%	-23%	-22%	-22%	-26%	-31%	-22%	...	-25%
ce[%]	35%	38%	15%	6%	17%	34%	41%	33%	11%	30%	...	15%
$\sigma(\sigma_{\mathbf{x}})$	0.014	0.013	0.017	0.018	0.016	0.012	0.013	0.012	0.015	0.015	...	0.017

C-4 Comparison few data points scenario

Table C-4: Approximations and accuracies per regression coefficient for the simulation with parameters of Table 2-13

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	...	x_{25}
$\hat{\mathbf{x}}$	0.717	1.967	0.000	0.000	0.000	9.967	-2.967	1.467	0.000	-0.617	...	0.000
$\sigma_{\mathbf{x}}^{MC}$	0.045	0.045	0.023	0.023	0.023	0.045	0.045	0.045	0.023	0.045	...	0.023
Approximation of Tibshirani: bias [%]	-32%	-28%	1%	-1%	-5%	-28%	-27%	-30%	0%	-32%	...	-2%
ce[%]	18%	11%	11%	13%	17%	12%	12%	11%	12%	16%	...	13%
$\sigma(\sigma_{\mathbf{x}})$	0.006	0.007	0.013	0.014	0.015	0.007	0.007	0.007	0.013	0.007	...	0.013
Approximation of Osborne: bias [%]	27%	25%	149%	145%	147%	24%	28%	21%	140%	25%	...	146%
ce[%]	-10%	-10%	-25%	-29%	-27%	-8%	-11%	-9%	-25%	-9%	...	-26%
$\sigma(\sigma_{\mathbf{x}})$	0.026	0.023	0.024	0.021	0.020	0.022	0.021	0.025	0.018	0.023	...	0.023
Vector resampling: bias [%]	175%	232%	137%	143%	143%	251%	249%	218%	142%	162%	...	144%
ce[%]	-2%	-1%	16%	12%	14%	-1%	-1%	-5%	11%	2%	...	13%
$\sigma(\sigma_{\mathbf{x}})$	0.045	0.075	0.026	0.030	0.032	0.082	0.080	0.065	0.032	0.040	...	0.033
Residual resampling B = 100: bias [%]	-39%	-38%	-42%	-41%	-40%	-39%	-38%	-40%	-39%	-39%	...	-40%
ce[%]	26%	26%	38%	34%	36%	28%	33%	28%	30%	30%	...	33%
$\sigma(\sigma_{\mathbf{x}})$	0.007	0.006	0.007	0.006	0.007	0.006	0.006	0.006	0.006	0.006	...	0.007
Residual resampling B = 1000: bias [%]	-38%	-38%	-41%	-41%	-39%	-39%	-37%	-40%	-38%	-39%	...	-40%
ce[%]	24%	26%	37%	32%	36%	32%	28%	27%	32%	31%	...	32%
$\sigma(\sigma_{\mathbf{x}})$	0.006	0.006	0.006	0.006	0.007	0.006	0.006	0.006	0.006	0.006	...	0.006
Parametric bootstrap: bias [%]	-37%	-37%	-40%	-39%	-38%	-37%	-36%	-39%	-37%	-38%	...	-38%
ce[%]	24%	25%	35%	32%	35%	32%	30%	27%	29%	30%	...	33%
$\sigma(\sigma_{\mathbf{x}})$	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	...	0.006
Fast residual resampling: bias [%]	115%	166%	545%	627%	242%	194%	121%	163%	506%	119%	...	329%
ce[%]	29%	23%	24%	25%	23%	30%	32%	26%	25%	24%	...	31%
$\sigma(\sigma_{\mathbf{x}})$	0.269	0.517	1.187	1.207	0.159	0.687	0.215	0.679	1.009	0.259	...	0.507
Proposed fast residual resampling: bias [%]	25%	23%	144%	141%	143%	22%	26%	18%	137%	23%	...	142%
ce[%]	21%	21%	20%	17%	18%	19%	23%	25%	17%	23%	...	14%
$\sigma(\sigma_{\mathbf{x}})$	0.025	0.022	0.023	0.021	0.020	0.022	0.021	0.024	0.018	0.023	...	0.023
Double bootstrap: bias [%]	-38%	-38%	-41%	-42%	-40%	-39%	-37%	-40%	-38%	-39%	...	-41%
ce[%]	32%	40%	50%	48%	48%	35%	43%	41%	14%	42%	...	12%
$\sigma(\sigma_{\mathbf{x}})$	0.007	0.007	0.007	0.006	0.007	0.006	0.006	0.006	0.006	0.006	...	0.007
Fast double bootstrap: bias [%]	109%	156%	505%	581%	236%	179%	116%	149%	471%	115%	...	312%
ce[%]	50%	43%	47%	40%	42%	47%	49%	44%	42%	46%	...	43%
$\sigma(\sigma_{\mathbf{x}})$	0.252	0.465	1.061	1.081	0.151	0.613	0.202	0.607	0.902	0.237	...	0.457
Proposed fast double bootstrap: bias [%]	25%	22%	144%	141%	141%	22%	26%	18%	138%	22%	...	142%
ce[%]	46%	40%	46%	42%	41%	39%	40%	42%	41%	43%	...	41%
$\sigma(\sigma_{\mathbf{x}})$	0.026	0.023	0.023	0.021	0.020	0.022	0.021	0.025	0.019	0.023	...	0.024
Modified bootstrap: bias [%]	-13%	-13%	-12%	-11%	-11%	-13%	-12%	-15%	-11%	-14%	...	-11%
ce[%]	14%	16%	17%	12%	16%	18%	17%	15%	12%	19%	...	14%
$\sigma(\sigma_{\mathbf{x}})$	0.009	0.008	0.006	0.006	0.006	0.008	0.008	0.008	0.006	0.009	...	0.006
Scalar UT: bias [%]	-44%	-44%	-31%	-36%	-33%	-41%	-35%	-48%	-30%	-41%	...	-36%
ce[%]	32%	24%	14%	13%	12%	28%	24%	29%	10%	24%	...	15%
$\sigma(\sigma_{\mathbf{x}})$	0.018	0.016	0.015	0.014	0.014	0.020	0.019	0.017	0.015	0.018	...	0.015
2-fold Scalar UT: bias [%]	-42%	-42%	-27%	-31%	-28%	-39%	-33%	-46%	-25%	-39%	...	-32%
ce[%]	29%	23%	7%	8%	2%	28%	22%	26%	5%	23%	...	8%
$\sigma(\sigma_{\mathbf{x}})$	0.017	0.016	0.015	0.015	0.015	0.020	0.018	0.017	0.015	0.018	...	0.015
3-fold Scalar UT: bias [%]	-42%	-42%	-27%	-31%	-28%	-39%	-33%	-45%	-25%	-39%	...	-31%
ce[%]	29%	25%	3%	5%	-1%	28%	21%	26%	2%	22%	...	5%
$\sigma(\sigma_{\mathbf{x}})$	0.017	0.016	0.015	0.015	0.015	0.020	0.018	0.017	0.015	0.018	...	0.015
Variance UT: bias [%]	-32%	-32%	-35%	-33%	-35%	-33%	-31%	-34%	-36%	-33%	...	-34%
ce[%]	15%	18%	25%	20%	24%	18%	16%	18%	29%	17%	...	32%
$\sigma(\sigma_{\mathbf{x}})$	0.007	0.007	0.014	0.014	0.013	0.007	0.007	0.006	0.013	0.007	...	0.013
2-fold Variance UT: bias [%]	-32%	-32%	-34%	-32%	-34%	-32%	-31%	-34%	-34%	-33%	...	-33%
ce[%]	15%	19%	10%	3%	5%	17%	16%	20%	8%	18%	...	10%
$\sigma(\sigma_{\mathbf{x}})$	0.007	0.007	0.013	0.013	0.013	0.007	0.007	0.006	0.013	0.007	...	0.013
3-fold Variance UT: bias [%]	-32%	-32%	-34%	-32%	-33%	-32%	-31%	-34%	-34%	-33%	...	-33%
ce[%]	16%	19%	4%	-2%	2%	17%	17%	20%	4%	18%	...	8%
$\sigma(\sigma_{\mathbf{x}})$	0.007	0.007	0.013	0.013	0.013	0.007	0.007	0.006	0.013	0.007	...	0.013

Bibliography

Jasper Anderluh. Elk klimaat eigen strategie. *TradersClub Magazine*, pages 20–21, 2011.

Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 492–499, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4191-4. doi: 10.1109/WI-IAT.2010.63. URL http://dx.doi.org/10.1109/WI-IAT.2010.63.

Francisca Beer, Fabrice Hervé, and Mohamed Zouaoui. Is big brother watching us? google, investor sentiment and the stock market. *Economics Bulletin*, 33(1):454–466, 2013. URL http://ideas.repec.org/a/ebl/ecbull/eb-13-00050.html.

Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

Howard D. Bondell and Brian J. Reich. Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR. *Biometrics*, 64(1): 115–123, March 2008.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3. 2011. ISBN 978-1-60198-461-6. doi: 10.1561/2200000016. URL http://dx.doi.org/10.1561/2200000016.

Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

CBOE. Cboe volatility index (vix), 2014. URL http://www.cboe.com/micro/VIX/vixintro.aspx.

Lahari S.N. Chatterjee A. Asymptotic properties of the residual bootstrap for lasso estimators. In *American Control Conference, 2003. Proceedings of the 2003*, pages 4497–4509, 2010.

A. Chatterjeea and S.N. Lahiria. Bootstrapping lasso estimators. *Journal of the American Statistical Association*, 106:608–625, 2011.

T. Coen, W. Saeys, H. Ramon, and J. De Baerdemaeker. Optimizing the tuning parameters of least squares support vector machines regression for nir spectra. *Journal of Chemometrics*, 20(5):184–192, 2006. ISSN 1099-128X. doi: 10.1002/cem.989. URL http://dx.doi.org/10.1002/cem.989.

Wensheng Dai and Chi-Jie Lu. Financial time series forecasting using a compound model based on wavelet frame and support vector regression. *2013 International Conference on Computing, Networking and Communications (ICNC)*, 5:328–332, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/ICNC.2008.455.

K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, and J.A.K. Suykens. Ls-svmlab toolbox user's guide version 1.8. 2010. URL http://www.esat.kuleuven.be/sista/lssvmlab/.

F.M. Dekking, C. Kraaikamp, and H.P. Lopuhaa. *A modern introduction to probability and statistics : understanding why and how.* Springer, New York, 2005. ISBN 1-85233-896-2.

D. Stauffer E. Samanidou, E. Zschischang and T Lux. Agent-based models of financial markets. *Reports on Progress in Physics*, 70(3):409, 2007. URL http://stacks.iop.org/0034-4885/70/i=3/a=R03.

S.G. Elmer. *Modern Statistical Methods Applied to Economic Time Series.* KOF dissertation series. KOF Swiss Economic Institute, ETH Zurich, 2011. URL http://books.google.nl/books?id=QS1OygAACAAJ.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties, 2001.

D.A. Freedman. Bootstrapping regression models. *The Annals of Statistics*, 9(6):1218–1228, 1981.

Eric Gilbert and Karrie Karahalios. Widespread worry and the stock market, 2010. URL https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1513.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944968.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002. ISSN 0885-6125. doi: 10.1023/A:1012487302797. URL http://dx.doi.org/10.1023/A%3A1012487302797.

Frederic Magoules Hai-xian Zhao. Feature selection for support vector regression in the application of building energy prediction. In *Applied Machine Intelligence and Informatics, 2011. Proceedings. 2011 IEEE International Symposium on*, volume 9, pages 219–223, 2011.

Wenwu He, Zhizhong Wang, and Hui Jiang. Model optimizing and feature selecting for support vector regression in time series forecasting. *Neurocomput.*, 72(1-3):600–611, December 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.11.010. URL http://dx.doi.org/10.1016/j.neucom.2007.11.010.

CuiQing Jiang, Kun Liang, Hsinchun Chen, and Yong Ding. Analyzing market performance via social media: a case study of a banking industry crisis. *Science China Information Sciences*, pages 1–18, 2013. ISSN 1674-733X. doi: 10.1007/s11432-013-4860-3. URL "http://dx.doi.org/10.1007/s11432-013-4860-3".

Simon Julier and Jeffrey K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, 1996.

Ling-Jing Kao, Chih-Chou Chiu, Chi-Jie Lu, and Jung-Li Yang. Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. *Neurocomput.*, 99:534–542, January 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.06.037. URL http://dx.doi.org/10.1016/j.neucom.2012.06.037.

Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *The Annals of Statistics*, 28(5):1356–1378, 10 2000. doi: 10.1214/aos/1015957397. URL http://dx.doi.org/10.1214/aos/1015957397.

Minjung Kyung and George Casella. Penalized regression, standard errors, and bayesian lassos. *Bayesian Analysis*, 5(2):369–412, 2010.

J.I. Larsen. Predicting stock prices using technical analysis and machine learning. Master's thesis, Norwegian University of Science and Technology, Norway, 2010.

Blake LeBaron. Chapter 24 agent-based computational finance. volume 2 of *Handbook of Computational Economics*, pages 1187 – 1233. Elsevier, 2006. doi: http://dx.doi.org/10.1016/S1574-0021(05)02024-1. URL http://www.sciencedirect.com/science/article/pii/S1574002105020241.

Hannes Leeb and Benedikt M. Potscher. Model selection and inference: Facts and fiction. *Econometric Theory*, null:21–59, 2 2005. ISSN 1469-4360. doi: 10.1017/S0266466605050036. URL http://journals.cambridge.org/article_S0266466605050036.

Guo-Zheng Li, Hao-Hua Mi, MaryQu Yang, and Jack Y. Yang. Combining support vector regression with feature selection for multivariate calibration. *Neural Computing and Applications*, 18(7):813–820, 2009. ISSN 0941-0643. doi: 10.1007/s00521-008-0202-6. URL http://dx.doi.org/10.1007/s00521-008-0202-6.

Naishan Li and Cuiling Liu. Application of svm to the prediction of water content in crude oil. In *Control, Automation and Systems Engineering (CASE), 2011 International Conference on*, pages 1–4, 2011. doi: 10.1109/ICCASE.2011.5997528.

Tim Futing Liao. *Statistical Group Comparison*. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN 9781118150610. URL http://books.google.nl/books?id=2OgongEACAAJ.

C. J. Lin and R. C. Weng. Simple probabilistic predictions for support vector regression. *Technical report, Department of Cmputer Science, National Taiwan University*, 2004.

Huina Mao, Scott Counts, and Johan Bollen. Predicting financial markets: Comparing survey,news, twitter and search engine data. *CoRR*, abs/1112.1051, 2011. URL http://dblp.uni-trier.de/db/journals/corr/corr1112.html#abs-1112-1051.

McCullough and Vinod. Implementing the double bootstrap. *Computational Economics*, 12(1):79–95, 1998. ISSN 0927-7099. doi: 10.1023/A:1008637230094. URL http://dx.doi.org/10.1023/A%3A1008637230094.

Moulton and Zeger. Bootstrapping generalized linear models. *Computational data and statistics analysis*, 11:53–63, 1991.

Guilherme Rosa J.M. Nanye Long, Daniel Gianola, Weigel, and A. Kent. Application of support vector regression to genome-assisted prediction of quantitative traits. *Theoretical and Applied Genetics*, 123(7):1065–1074, 2011. ISSN 0040-5752. doi: 10.1007/s00122-011-1648-y. URL http://dx.doi.org/10.1007/s00122-011-1648-y.

Federal Research Bank of St. Louis. Dow jones industrial average (djia), 2013. URL http://research.stlouisfed.org/fred2/series/DJIA/downloaddata.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the LASSO and Its Dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000. ISSN 10618600. doi: 10.2307/1390657. URL http://dx.doi.org/10.2307/1390657.

Benedikt M. Pötscher and Hannes Leeb. On the distribution of penalized maximum likelihood estimators: The lasso, scad, and thresholding. *Journal of Multivariate Analysis*, pages 2065–2082, 2009.

Tobias Preis, Helen S. Moat, and H. Eugene Stanley. Quantifying Trading Behavior in Financial Markets Using Google Trends. *Scientific Reports*, 3, April 2013. ISSN 2045-2322. doi: 10.1038/srep01684. URL http://dx.doi.org/10.1038/srep01684.

Tushar Rao and Saket Srivastava. Modeling movements in oil, gold, forex and market indices using search volume index and twitter sentiments. *CoRR*, abs/1212.1037, 2012.

S. Sartori. Penalized regression: Bootstrap confidence intervals and variable selection for high-dimensional data sets. *Milano: Università degli studi di Milano. Universita'degli Studi di Milano*, 2011.

SAS. Bivariate granger causality test, 2014. URL http://support.sas.com/rnd/app/examples/ets/granger/.

Mark Schmidt. Lasso, matlab code for solving lasso problems, 2005. URL http://www.cs.ubc.ca/~schmidtm/Software/lasso.html.

Jasmina Smailovic, Miha Grcar, Nada Lavrac, and Martin Znidarsic. Predictive sentiment analysis of tweets: A stock market application. In Andreas Holzinger and Gabriella Pasi, editors, *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, volume 7947 of *Lecture Notes in Computer Science*, pages 77–88. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39145-3. doi: 10.1007/978-3-642-39146-0_8. URL http://dx.doi.org/10.1007/978-3-642-39146-0_8.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004. ISSN 0960-3174. doi: 10.1023/B:STCO.0000035301.49549.88. URL http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

Doug Steigerwald. Double bootstrap, improve the accuracy of the bootstrap coverage, 2010. URL http://econ.ucsb.edu/~doug/245a/Lectures/DoubleBootstrap.pdf.

Stockcharts. Money flow index (mfi), 2014a. URL http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:money_flow_index_mfi.

Stockcharts. Rate of change (roc), 2014b. URL http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:money_flow_index_mfi.

D. Tenne and T. Singh. The higher order unscented filter. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2441–2446 vol.3, 2003. doi: 10.1109/ACC.2003.1243441.

D.W.P. Thomas, O.A. Oke, L.R.A.X. de Menezes, and C. Christopoulos. The use of unscented transforms in modeling the statistical response of nonlinear scatterer in a reverberation chamber. In *General Assembly and Scientific Symposium, 2011 XXXth URSI*, pages 1–4, Aug 2011. doi: 10.1109/URSIGASS.2011.6050745.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

R. Van der Merwe and E.A. Wan. The square-root unscented kalman filter for state and parameter-estimation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 6, pages 3461–3464 vol.6, 2001. doi: 10.1109/ICASSP.2001.940586.

Samuel Xavier-De-Souza, Johan A. K. Suykens, Joos Vandewalle, and Désiré Bollé. Coupled simulated annealing. *Trans. Sys. Man Cyber. Part B*, 40(2):320–335, April 2010. ISSN 1083-4419. doi: 10.1109/TSMCB.2009.2020435. URL http://dx.doi.org/10.1109/TSMCB.2009.2020435.

Jian-Bo Yang and Chong-Jin Ong. Feature selection using probabilistic prediction of support vector regression. *Neural Networks, IEEE Transactions on*, 22(6):954–962, June 2011. ISSN 1045-9227. doi: 10.1109/TNN.2011.2128342.

Chi-Yuan Yeh, Chi-Wei Huang, and Shie-Jue Lee. A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3):2177 – 2186, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.08.004. URL http://www.sciencedirect.com/science/article/pii/S0957417410007876.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, December 2006.

Glossary

List of symbols

α_n	Lagrangian multiplier in the SVR
$\boldsymbol{\alpha}$	Vector of Lagrangian multipliers in the SVR, length N
γ	Kernel parameter in the RBF of the SVR
γ^i	Kernel parameter at step i in the RFE methods
δ_0	The Dirac impulse function at 0
ε	Stochastic variable, Gaussian noise
κ	Tuning parameter of the UT
λ	Tuning parameter of the LASSO
μ_p^A	Mean values for the feature p of A
$\boldsymbol{\mu}^A$	Vector containing mean values per feature of A
$\rho(\dots)$	General shrinkage function
$\mathcal{S}$	The set of support elements, the non-zero elements of $\mathbf{x}$, or the selected features in the SVR framework
σ_p^A	Standard deviation for the feature p of A
$\bar{\sigma}_{x_p}$	Average estimated standard deviation σ_{x_p}
σ_ε^s	Estimation of σ_ε without using feature s
$\sigma(\sigma_{x_p})$	The standard deviation of the estimated standard deviation σ_{x_p}
$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})}$	The standard deviation of $\boldsymbol{\sigma}_\mathbf{x}$ averaged over the different regression coefficients
$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})_\mathcal{S}}$	The standard deviation of $\boldsymbol{\sigma}_\mathbf{x}$ averaged over the different regression coefficients in the support $\mathcal{S}$
$\overline{\boldsymbol{\sigma}(\boldsymbol{\sigma}_\mathbf{x})_{\not\mathcal{S}}}$	The standard deviation of $\boldsymbol{\sigma}_\mathbf{x}$ averaged over the different regression coefficients not in the support $\mathcal{S}$
$\boldsymbol{\sigma}^A$	Vector containing the standard deviations per feature of A
$\boldsymbol{\sigma}_\mathbf{x}$	Vector containing the standard deviations of $\mathbf{x}$
σ_x	Standard deviation of x

$\Sigma_{\mathbf{x}}$	Covariance matrix of $\mathbf{x}$
$\Sigma_{\mathbf{x}}^{MC}$	Found covariance matrix of $\mathbf{x}$ using the Monte Carlo method
$\Sigma_{\mathbf{x}}(p,p)$	The (p,p) entry of the matrix $\Sigma_{\mathbf{x}}$
τ_i	Sigma point i in the Higher Order Unscented Transform
$\phi(\dots)$	Probability density function of the standard normal distribution
$\Phi(\dots)$	Cumulative density function of the standard normal distribution
Ψ	Matrix used to approximate the solution of $\mathbf{x}$, $\mathbf{x} \approx \Psi\mathbf{y}$
Ψ^b	Matrix Ψ calculated at the second step b in the double bootstrap algorithm
$\psi(\dots)$	Nonlinear transformation in the SVR
$(\dots)_+$	Soft thresholding function $(\dots)_+ = max(\dots, 0)$
$\{\dots\}_{(16\%)}$	16% percentile value of the set $\{\dots\}$
$\mathbf{0}$	Vector containing zeros
$\mathbf{1}$	Vector containing ones
$\mathbf{1}(\dots)$	Indicator function, resulting in 1 if the statement is true and 0 if it is untrue
$a_{n,p}$	Entry (n,p) in the data matrix A
$\tilde{\mathbf{a}}_n^i$	Generated $\mathbf{a}_n$ sample in bootstrapping
A	Data matrix, dimension $N \times P$
A^{fix}	A matrix containing the columns belonging to the features in $\mathcal{S}^{fix}$
A^i	Generated A sample in the Monte Carlo method or the A matrix in the RFE method at iteration i
$A_{\mathcal{S}}$	The data matrix A only containing the columns / features of the support $\mathcal{S}$
A^s	A matrix A without feature s
A^{selec_1}	A matrix containing the columns belonging to the features in $\mathcal{S}^{selec_1}$
A^{selec_2}	A matrix containing the columns belonging to the features in $\mathcal{S}^{selec_2}$
A_t	Temporary A matrix in an iteration in the RFE methods
$\tilde{A}^i$	Generated A sample in bootstrapping
$\mathbf{a}_n$	Row of the data matrix, data point n
b	Offset parameter in the SVR
$b(\sigma_{x_p})$	The bias of the average estimated standard deviation $\bar{\sigma}_{x_p}$
B	Number of bootstrap iterations
$ceil(..)$	Function to round its argument up to the nearest integer
ce_p	Confidence interval error for regression coefficient p
c_N	Threshold parameter in the Modified bootstrap method
c_n	Closing price at time index n
co_p^i	Entry p from $\mathbf{co}^i$, indicating if x_p^{true} is contained in the confidence interval
$\mathbf{co}^i$	Coverage vector, indicating if $\mathbf{x}^{true}$ is contained in the confidence interval
C	Tuning parameter of the SVR determining the allowance for nonlinearity
$Candle_{1,n}$	Candlestick Range at time step n
$Candle_{2,n}$	Candlestick Body at time step n
$Candle_{3,n}$	Candlestick Upper Shadow at time step n

$Candle_{4,n}$	Candlestick Lower Shadow at time step n
C^i	Tuning parameter at step i in the RFE methods
$Crisis_n$	Google Insights for Search term 'Crisis' at time step n
$diag(\dots)$	Short notation for a square matrix with $\dots$ on its diagonal
D	Number of iterations in the second step of the double bootstrap
DA_n	The directional accuracy of prediction n
D_n^{fast}	Stochastic %D^{fast} line at time step n
$DJIA_n$	Google Insights for Search term 'DJIA' at time step n
$D_{KL}(\dots)$	Kullback-Leibler divergence function
D_n^{slow}	Stochastic %D^{slow} line at time step n
D^y	Number of delays of time-series y used in the AR(X) model in the Granger causality test
e_n	Error value for data point n of the SVR
$\mathbf{e}$	Error values vector of the SVR, length N
$exp(\dots)$	Exponential function: $e^{\cdots}$
$E[\dots]$	Expected Value
$Economy_n$	Google Insights for Search term 'Economy' at time step n
EMA_n^i	Exponential Moving Average over i days at time step n
$f_{coverage}(\dots)$	Function used to determine if the true regression vector is contained in a confidence interval
$f(\dots)$	Nonlinear transformation
$f_{generating}(\dots)$	generating function for the data matrix A
$Finance_n$	Google Insights for Search term 'Finance' at time step n
$floor(..)$	Function to round its argument down to the nearest integer
$\mathbf{f}_p$	Column of the data matrix, feature p
$F(\dots)$	F-distribution function
$\breve{g}_p$	Index representing the proportion of samples $\breve{q}_p^{b,d}$ smaller or equal to $\tilde{q}_p^b$
$google_n$	Google Insights for Search term at time step n
G	Generalized matrix in the Ridge regression
h_n	Highest day price at time index n
$\tilde{h}_{p(\dots)}$	Correcting value for the improved confidence interval value in the double bootstrap method
$Happy_n$	Google Insights for Search term 'Happy' at time step n
H_n	Highest price over the last 14 days at time index n
$index$	Generated index value in bootstrap methods
I_N	Identity matrix of size $N \times N$
$Interest_n$	Google Insights for Search term 'Interest' at time step n
J	Objective function of an optimization problem
$\boldsymbol{K}$	Kernel matrix of the SVR
$K(\mathbf{a}_{n_1}, \mathbf{a}_{n_2})$	Kernel function of the SVR
K_n	Stochastic %K line at time step n

$l(\dots)$	Loss function
l_n	Lowest day price at time index n
l_{window}	Windows length of the Windowed Granger Causality test
$\mathcal{L}$	Lagrange objective function
L_n	Lowest price over the last 14 days at time index n
$max(\dots)$	Function to determine maximum of the arguments
M	Number of Monte Carlo iterations
$MFI_{1,n}$	Money flow index, typical price at time step n
$MFI_{2,n}$	Money flow index, raw money flow at time step n
$MFI_{3,n}$	Money flow index, ratio at time step n
$MFI_{4,n}$	Money flow index at time step n
M_o	The o-th moment of the distribution of y
MSE^i	MSE value at step i in the RFE methods
MSE^s_t	Temporary MSE value at step i in the RFE methods without using feature s
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and standard deviation σ
N	The number of data points, the length of $\mathbf{y}$
o_n	Opening price at time index n
$p(\dots)$	Probability density function
$p_{exp}(\dots)$	Exponential distribution
P	The number of features, the length of $\mathbf{x}$
$Pr(\dots)$	Probability function
$\breve{q}^{b,d}_p$	The root value of the double bootstrap sample $\breve{x}^{b,d}_p$
$\tilde{q}^b_p$	The root value of the bootstrap sample $\tilde{x}^b_p$
$\bar{r}$	The average estimation error, $\bar{r} = 1/N \sum_{n=1}^{N} r_n$
$\breve{r}^{b,d}_n$	Selected estimated error value in the second step b of the double bootstrap
r_n	The estimation error for data point n, $r_n = y_n - \mathbf{a}_n\mathbf{x}$
$\tilde{r}^b_n$	A selected estimation error in the residual resampling method
ROC_n	Rate of Change at time step n
RSS_{AR}	Residual Sum of Squares of an AR model
RSS_{ARX}	Residual Sum of Squares of an ARX model
$sign(\dots)$	Function to determine the sign value: -1 if the value is negative and 1 if the value is positive
s_{min}	Feature which yielded the lowest MSE in an iteration in the RFE methods
s_p	Sign of the value x_p
$\mathbf{s}$	Vector containing the sign values of the vector $\mathbf{x}$
$\mathcal{S}^i$	Support at step i in the RFE methods
S	Test statistic of the SRFE method
$\mathcal{S}^{fix}$	Set of the features belonging to Support which are not retuned in the second step of the WRFE
$\mathcal{S}^i$	Support at step i in the RFE methods

SMA_n^i	Simple Moving Average over i days at time step n
S^s	Test statistic of the SRFE method, without using feature s
$\mathcal{S}^{selec_1}$	Set of the features belonging to Support which are retuned in the second step of the WRFE
$\mathcal{S}^{selec_2}$	Set of the features not belonging to Support which are retuned in the second step of the WRFE
$Stock_n$	Google Insights for Search term 'Stock' at time step n
$StockToday_n$	Google Insights for Search term 'Stock Market Today' at time step n
T	The distribution of $\sqrt{N}(\mathbf{x} - \mathbf{x}^{true})$
$\tilde{T}$	The estimated distribution of T, $\tilde{T} = \sqrt{N}(\tilde{\mathbf{x}} - \mathbf{x})$
$u_{dist}(N)$	uniform discrete distribution with range [1 .. N]
U	Test statistic in the Granger causality test
v_n	Volume at time index n
$\mathbf{v}$	Weight vector in the SVR, length N
VIX_n	Market Volatility Index at time step n
w_j	Weight value used in the UT and the WRFE
$W_{\mathcal{S}}^-$	The matrix W^- containing only the values of $\mathbf{x}_{\mathcal{S}}$
$\mathbf{w}$	Weight vector used in the UT and the WRFE
$\mathbf{w}^{selec_1}$	Weight vector in WRFE for $\mathcal{S}^{selec_1}$
$\mathbf{w}^{selec_2}$	Weight vector in WRFE for $\mathcal{S}^{selec_2}$
W_2	Weight matrix used in the analytical approximation of Osborne
x_p	Regression coefficient p
x_p^*	Thresholded value of x_p in the Modified bootstrap method
$x_{p(16\%)}$	Estimated 16% percentile value of x_p
x_p^{*m}	Sorted value of the set $x_p^1 \dots x_p^M$
$\mathbf{x}$	The estimated regression coefficient vector, dimension P
$\mathbf{x}_{(16\%)}$	Estimated 16% percentile values of $\mathbf{x}$
$\mathbf{x}_{(16\%)}^{MC}$	16% percentile value of the set $\{\mathbf{x}^1 \dots \mathbf{x}^M\}$
$\bar{\mathbf{x}}$	Average $\mathbf{x}$ of a set of $\mathbf{x}$s
$\breve{\mathbf{x}}^{b,d}$	Generated $\mathbf{x}$ sample in the second step b of the double bootstrap
$\mathbf{x}^i$	Generated $\mathbf{x}$ sample in the Monte Carlo method
$\mathbf{x}_{\mathcal{S}}$	The regression vector $\mathbf{x}$ containing only the features of the support $\mathcal{S}$
$\tilde{\mathbf{x}}^i$	Generated $\mathbf{x}$ sample in bootstrapping
$\overline{\tilde{\mathbf{x}}^b}$	Average $\mathbf{x}$ of the samples in the second step b of the double bootstrap
$\mathbf{x}^{true}$	The true regression coefficient vector, dimension P
$x_{y,i}$	Regression coefficient of time series y at time instance i used in the AR(X) model in the Granger causality test
$\hat{y}_n$	Predicted / estimated output y_n
$\hat{y}_n^s$	Estimation of y_n without using feature s
$\tilde{y}_n^i$	Generated y_n sample in bootstrapping
$\mathbf{y}$	Output vector, dimension N

$\breve{\mathbf{y}}^{b,d}$	Generated $\mathbf{y}$ sample in the second step b of the double bootstrap
$\mathbf{y}^i$	Generated $\mathbf{y}$ sample in the Monte Carlo method
$\tilde{\mathbf{y}}^i$	Generated $\mathbf{y}$ sample in bootstrapping
$z(\dots)$	Z-score, pre-processing used for the Granger causality test

List of acronyms

CV	Cross Validation
DA	Directional Accuracy
GIS	Google Insights for Search
GLM	Generalized Linear Model
LASSO	Least Angular Shrinkage and Selection Operation
MAPE	Mean Absolute Percentage Error
MC	Monte Carlo
MSE	Mean Squared Error
OLS	Ordinary Least Squares
Q-Q	Quantile-Quantile
RBF	Radial Basis Function
RFE	Recursive Feature Elimination
RSS	Residual Sum of Squares
SH	Short Horizoning
SOFNN	Self Organizing Fuzzy Neural network
SRFE	Statistical Recursive Feature Elimination
SVR	Support Vector Regression
UT	Unscented Transform
WRFE	Weighted Recursive Feature Elimination