Electronics and Computer Science Faculty of Physical and Applied Sciences University of Southampton

Alejandro Saucedo

5th December, 2013

sparse and group regression models

in portfolio optimization

Project Supervisor: Prof. Mahesan Niranjan

Second Examiner: Prof. Vladimiro Sassone

A progress report submitted for the award of

Software Engineering BSc

# Abstract

Current approaches to portfolio optimization only consider financial instruments as individual entities, not taking any advantage of the huge amounts of categorizing information available on the underlying financial instruments (i.e. instrument type, industry, sector, volatility level, etc). In this paper we take a novel approach to portfolio optimization, where our main objective is to show that it is possible to exploit categorizing information on the financial instruments in the portfolio and benefit from group correlations present in the data to obtain richer results.

This paper focuses in a very popular branch of portfolio optimization – namely, market index tracking, where the aim is, chosen a Market Index (i.e. a set of high-performing stocks), we want to find a subset that follows the behaviour of its respective Market Index as close as possible. Our approach to solving this problem consists of feature-level regression models with a cardinality (L0-norm) constraint[[1]](#footnote-1), and sparse-inducing group-level regression models[[2]](#footnote-2). These two approaches will be introduced, analysed and compared in order to provide an insight on the effect group characteristics have when implemented in financial datasets.

Given that the Sparse Group Model analysed in this paper is limited to a single category (i.e. only one category of groups can be taken into consideration) a new regression model was proposed based on our results. This model suggests to take into consideration multiple categories of groups (i.e. type of financial instrument, sector, volatility, etc.) in order to provide more diverse portfolios, and more accurate results.

Finally, the code that was written for the implementation of these models can be found online in a well documented GitHub repository, which has been registered on an Open Source licence and is available for download.

# Contents

Abstract 2

Contents 2

1 Introduction 5

1.1 Background Research 5

1.2 Implementation 6

1.3 Analysis 7

1.4 Expansion 7

1.5 Testing Error! Bookmark not defined.

1.6 Conclusion 8

2 Background and Report of Literature Search 9

2.1 Mathematical Definitions 9

2.1.1 General 9

2.1.2 Group Notation 10

2.2 Financial Definitions 10

2.2.1 Financial Risk **Error! Bookmark not defined.**

2.2.2 GARCH Prediction Model 11

2.2.3 Autoregressive Integrated Moving Average 11

2.2.4 Modern Portfolio Theory 11

2.2.5 Value-at-Risk 12

2.2.6 Expected Shortfall 13

2.2.7 Tracking Error 12

2.3 Simple Regression Models Error! Bookmark not defined.

2.3.1 β-VaR 13

2.3.2 Norm Constrained CVaR 14

2.3.3 L0+L2 Model 14

2.3.4 Lasso Regression 16

2.4 Model Selection Approaches 19

2.4.1 Full Search 19

2.4.2 Greedy Search 19

2.5 Group Regression Approaches 20

2.5.1 Model Selection Approach 20

2.5.2 Group Lasso Regression 21

2.6 Zero-Constrained Group Lasso Application 23

3 Implementation 25

3.1 Working Environment 25

3.1.1 CVX 25

3.1.2 SPAMS 26

3.2 Implementation Design 26

3.2.1 DataSets 26

3.2.2 Data grouping 27

3.2.3 Financial Mathematical Notation 27

3.3 Core Financial Formulations 27

3.4 Simple Regression Models 28

3.5 Sparse Inducing Models 29

3.5.1 Simple Lasso model 29

3.6 Individual-Feature Selection 31

3.6.1 Algorithm 31

3.6.2 Feature-Selection Implementation 32

3.7 Group Models 34

3.7.1 Group Selection 35

3.8 Sparse Group Models 37

3.8.1 Group Lasso 37

3.8.2 Sparse Group Lasso 39

4 Analysis 45

4.1 Model Comparison 45

4.1.1 Accuracy 45

4.1.2 Sparsity **Error! Bookmark not defined.**

4.1.3 Speed 48

4.2 Applications 48

5 Expansion 49

6 Conclusion 53

7 References 55

# 

# Introduction

Portfolio optimization has been a major research topic since it was introduced in Markowitz’s paper, Modern Portfolio Theory [REFERENCE] – one of the most influential research papers in the field of finance. Since its debut, this theory has been dissected, analyzed and expanded thoroughly, and has been expanded into several branches, including index tracking, enhanced indexation, absolute return, market neutral, between others.

Until this date, portfolio optimization models have only considered financial instruments as individual entities of data – that is, using only common regression models (e.g. linear, ridge, weighted regression models). In this paper, we take a completely novel approach on portfolio optimization, where we take into consideration groups of financial instruments as opposed to only the individual instruments. These groupings are achieved through prior knowledge of the financial instruments, such as financial sector, industry of the underlying company, or any characteristic of the instruments that allows for grouping.

A background on al the financial terms, concepts and definitions required to understand and implement these approaches will be given throughout this paper. Likewise, all the machine learning models, formulas and definitions will be dissected and explained in order to provide an intuitive understanding on the application and implementation of these group regression approaches.

This paper is structured in six main sections – namely, Background Research, Implementation, Analysis, Expansion, Testing and Conclusion.

The working code for the Implementation of this paper has been registered in an Open Source public licence, which is being hosted in GitHub at the time of writing, and can be downloaded at http://github.com/axsauze/sparse. Links to all the papers referenced by this paper, as well as graphs and all material related to this paper are also included in this repository.

## Background Research

Given the complex nature of this research topic, there was a lot background research required to obtain an intuitive understanding on the models and concepts related. This section aims to provide the reader with the core knowledge required on the financial, mathematical, statistical and algorithmic concepts, definitions and formulations related to this research topic.

Understanding on several financial concepts will be important to provide knowledge on the application of this model. This section provides the mathematical notations and definitions that will be consistent throughout the paper, as well as financial definitions and formulations, such Volatility, Financial Index, Modern Portfolio Theory, Portfolio Optimization, Index Tracking, Value-at-Risk, Conditional-Value-at-Risk, between others.

It will also be important to cover the machine learning and algorithmic models that are present in this research project. These are broken down and analyzed in order to provide an intuitive and clear knowledge. Models covered in this paper are Conditional Value at Risk minimization, Linear, Ridge, Weighted, Lasso and Lars regression, the L0-L2-norm model, and several group regression approaches. The algorithmic approaches taken are namely full-search and greedy approaches.

## Implementation

The aim of this section is to implement the models and concepts introduced previously in order to discover the effects of group analysis in financial data as opposed to individual stock analysis, and whether using this group knowledge allows for better results.

Each individual regression model is applied to Market Index Data for a specific window of time. Experiments are based completely on Index Tracking, and the results are given through the final Tracking Error. In few words, the index tracking minimization problem consists on finding a subset of stocks from a specific portfolio that behaves as similar as possible to the original portfolio (i.e. has the lowest tracking error).

Each model contains several constants that need to be adjusted to provide reliable and desirable results – some of these constants define level of sparsity to be achieved, as well as other characteristics. Several experiments will be carried out to find efficient values for these variable to provide reliable results.

Although stochastic data will be used initially, the main financial data set used in this paper is the FTSE100 (Pronounced ‘footsie’ 100) – which is a share index of the 100 companies listed on the London Stock Exchange with the highest market capitalization.

The initial approach for group analysis of stocks is applied through a greedy approach similar to the one in [0], however it will be explained that most of the times it is possible to apply full search approaches, as the number of groups tend to be much smaller.

Group regression model applications in financial data are also visited briefly in the implementation section of this paper, however these are expanded in the Analysis and Expansion sections.

## Analysis

In this section, this paper aims to provide a thorough and in-depth analysis of the results obtained in the implementation section.

The objective of this section is to provide the effects of Index Tracking applications when groups are taken into consideration.

This section provides a perspective on the differences between the models being considered, and aims to provide a more qualitative-oriented, intuitive analysis on the results obtained (As opposed to the more quantitative perspective of the previous section).

Several real-life examples as well as applications are considered in this section to provide a clear understanding on these individual and group approaches, and provides several ideas to introduce the next section – Expansion of group approaches in portfolio optimization.

## Expansion

Although in this paper we focus on Market Index and Index Trakcing is certainly possible to expand this into any Market Index, or even custom portfolios containing distinct types of financial instruments (e.g. commodities, options, bonds, etc).

In order to expand the application of it is necessary to obtain further knowledge of the specific financial concepts required. These concepts include commodities, currencies, high frequency and real time data.

The currencies market varies a lot from the stock market, and some of its characteristics would make the application of this model ambiguous – an example would be the constraint that the model requires which does not allow shorting – however, in currencies, although one might go long in one currency (Such as long on EUR/USD), one could say it is shorting on the other (such as short on USD/EUR).

Commodities in the other hand have a seasonality characteristic, in which financial commodities such as specific fruits might have drastic increase or decrease of price due to the time of the year due to numerous environmental factors.

High frequency data might require an implementation in a different programming language and environment due to the considerably larger sets of data that will need to be processed. Real time data in the other hand, will require the model implementation to be revisited, as the local optimal nature of the greedy algorithm would not allow for updates as new data comes in.

## Conclusion

We finalize this paper by providing a brief overview of the results obtained, and the final observations made. This section is very important as it would be ideal that this paper can provide a base, or an addition to current research. Conclusions made in this paper are

# Background and Report of Literature Search

In this paper we aim to discover the effects of feature and group level approaches to Market Index tracking. In order to achieve this, it is required to understand several financial terms, as well as the Machine Learning concepts used in this paper. This section aims to provide the reader with the core knowledge required on definitions, formulas and concepts related to the topics that comprise feature and group level regression approaches in our financial datasets.

This project was initially inspired by [0] Mahesan et al. (2013), where an innovative regression model was proposed – namely the L0+L2-norm model. This model proposed is mainly a ridge regression model (i.e. constrained on an L2-norm) with a cardinality constraint (i.e. constrained on an L0-Norm) – hence the name of the model. The main objective of this paper is to find an optimal subset of stocks from a Market Index with the lowest possible tracking error (Training error for Index Tracking problems), which is approached through a greedy algorithm through an increasing L0-norm constraint. In this paper we expand this concept to various regression models, we compare it to sparse inducing models, and we introduce several very interesting group regression approaches. These terms and regression models will be explained further.

## Mathematical Definitions

### General

We will use σ to refer to the volatility of a financial instrument (i.e. the standard deviation of the daily returns of a specific financial instrument).

Norms will be referred as L–norms, where as .

The statistical and mathematical notations used in this paper are all considered within a specific time window t = 1, ...,T and a number of assets n.

Input data is of the form **R**t∈ **ℝ**n= (**R**t,1, **R**t,2, …, **R**t,n)Τ where **R** is an **ℝ**Txn matrix where each column is a vector of returns of all the assets of the portfolio at time t.

Our target data is of the form  where ***I***is a Market Index (i.e. a share index of the n companies listed on a specific Stock Exchange with the highest market capitalization) and **π** ∈ **ℝ**n = (**π**1, …, **π**n)Τ is the parameter to be learned, and consists of the proportion of the total capital invested on each asset in the portfolio.

This paper will be implicitly limited by the constraints **π** > 0 which ensures that it is not possible to sell assets that are not owned (shorting) and .

### Group Notation

When dealing with groups, we will have groups of size , where .

Our stocks will be grouped in these m groups of size , which means our data **R** is in groups **,** where and .

Although our Index Portfolio would be referenced with the same variable **I**, a new group definition is now introduced as , where are the parameters to learn that belong to the stocks in group .

For the new group formulation proposed we will need to introduce the concept of superscripting (same as in arrays in code) in order to obtain specific elements in matrices and vectors through indexing. In this paper we will use the pseudo-code notation , where **X** is a matrix of any size, **x** and **y** are vectors of the same size containing the positions in the Matrix to be superscripted, and they are of the form . A column can be used to denote that all columns or rows are selected, so would superscript all the row elements for columns contained in the indexes **y**.

For indexing we will use a variable that will denote the indexes of the stocks of each group. For example, if group contains stocks 3, 5, 6, etc, then = {3, 5, 6, …}. This allows to introduce the logical biconditional.

The final notation required for the new proposed formula is the sum of all the columns or rows as where r is the number of rows in **A** and c is the number of columns in **A**.

## Financial Definitions

### Basic financial terms

* **Financial risk** - potential loss or uncertainty from an investment, and can be measured through mathematical and statistical models.
* **Volatility** - A factor that is often used when measuring financial risk previously defined as σ. The volatility of a financial instrument is the standard deviation of the historical returns.
* **Portfolio –** A group of financial assets such as bonds, stocks where an amount of money is invested
* **Market Index** – An aggregate of stocks from a financial market that aim to represent an entire stock market.

### Portfolio Diversification

In finance it is a very important concern to obtain diversification when dealing with investments. What this means is to distribute (diversify) risk through several (preferably uncorrelated) financial instruments. This way, if the price of a group of financial instruments goes down, losses would not be as bad, as a diversified portfolio would have investments in numerous disjoint groups.

### GARCH Prediction Model

### Autoregressive Integrated Moving Average

### Modern Portfolio Theory

In finance, portfolio optimization is an extremely popular and revisited concept since its debut in the original paper by [1] Markowitz(1952), ‘Porfolio Theory’. More widely known as Modern Portfolio Theory, it consists of a minimization on the risk for a given level of expected return through choosing the proportions (weights) of the assets that comprise the portfolio. The formulation introduced in this paper is defined as follows:

Equation – MPT Minimization

Using input data over a range of time **T** for **n** assets, this Markowitz portfolio optimization formula minimizes on the portfolio variance given by, where **π** is the proportion invested on each asset, and **R** ∈ **ℝ**nxn is a covariance matrix of asset is the parameter to be learned. The amount of expected return is given by *γ*, and a vector of expected returns for each asset is given by ***r*** ∈ **ℝ**n. This model revolutionized portfolio theory since its debut, and has been thoroughly revisited since.

### Tracking Error

In this paper, we will base all our calculations in a very commonly used loss function – the Tracking Error. Tracking error is very intuitive, especially when seen graphically – it basically measures how similar a portfolio A behaves to a portfolio B. We will refer to tracking error as ξφ, t(**π**) defined as:

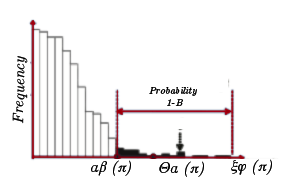
Equation - Tracking Error

This tracking error function will also be used to measure the results of the accuracy of our model compared to other models proposed.

## Background in Regression Applications for Finance

In order to obtain an understanding on how regression models can be effectively implemented in financial datasets, we require to introduce some financial concepts before this, including Value-at-Risk and it’s variations.

### Value-at-Risk

A very important risk management model in finance is the Value-at-Risk model, or VaR [2] Philippe (1996). The most common application of Value-at-Risk is the β%-VaR. Given a percentage β% and a set of historical prices, we can build a frequency table which we assume has normal distribution, and then we can estimate the loss that we are β% sure that will not be exceeded.

Mathematically, given a confidence index α based on β% (i.e. 95% would be 2 standard deviations), a mean μ and a volatility σ, we can compute the model presented in [6] Mina et al. β%-VaR from a portfolio as follows:

Equation - B%-VaR

Value-at-Risk is excellent when dealing with raw calculations from a given volatility, however it lacks of convexity characteristics, making it hard to optimize, and hence unattractive for minimization problems.

### Expected Shortfall

β-Conditional-Value-at-Risk, also known as Expected Shortfall, is the conditional expectation of losses above β%-VaR – in other words, the probability that a specific loss will exceed the β%-VaR. Unlike VaR, CVaR has many desired characteristics such as convexity, that will allow for the definitions required for using this as a minimization problem.

To define this, we base ourselves in the mathematical notation in [Akiko 2010] and in the logic of [4] Rockafellar R. et al. (2000). We initially base ourselves in a regression problem to obtain a linear function approximator y = (w,x) + b from m = 1, ..., M samples (xi, yi). Our variables x ∈ **ℝ**nand y ∈ **ℝ**nare our input and output values respectively and w ∈ **ℝ**nand b∈ **ℝ**are the variables to be learned.

### β-VaR

Once defined our loss function, in this case, Tracking Error, we assume we an get probability of a specific result of by computing ρ( This allows us to define a function Θα(**π**) that gives us the cumulative distribution for our probability loss function ρ( given a given confidence α. This is, in other words, the probability of ξφ(**π**) not exceeding a threshold given by α. The function θα(**π**) is defined as follows:

Finally, now that we can calculate the cumulative distribution of our loss function we can obtain the definition of our β-VaR formula defined as:

Now that we have obtained β-VaR, or the lowest percentage amount that we will be β% sure that will not be exceeded, we can define the CVaR as the average loss **exceeding** VaR. Now that we have reached this definition, we can bring up the proof in [4] Rockafellar R. (2000), which shows the convex nature of CVaR, and allows for a minimization definition as follows:

Where when t > 0 and when .

## Feature Regression Models

Now that the necessary formulations were introduced to provide the reader with a core understanding on some of the minimization functions in financial data, as well as an idea on their potential applications we can proceed to discuss the regression models that will be used for single-feature analysis in this paper.

### CVaR Minimization

It is proven in Akiko [REFERENCE 1] that this CVaR minimization is equivalent to the Support Vector Regression algorithm, implying optimality, and also includes a proof that allows us to introduce the variable **zt,** where **zt** allows us to use the definition of by a simple rearrangement of constraints, hence simplify the implementation of the problem in our algorithm massively. This allows us to get our final definition of our CVaR minimization formula as follows:

s.t.

It is worth bringing up the fact that although the formulation provided in AKIKO [REFERENCE 1], called the Norm Constrained CVaR (NCCVaR) implements a sparsity inducing variable C2 which induces sparsity in the set of weights. Although the NCCVaR formulation won’t be used, a lot of the proofs and concepts present in this paper were of great help in order to be able to use the CVaR minimization formula efficiently.

### L0+L2-norm Model (Ridge Regression)

The L0+L2-norm model as proposed in [REFERENCE MAHESAN], as its name implies is basically a regression model constrained by an L0-norm and an L2-norm. In simple words, this model is basically a Ridge Regression model with a cardinality constraint. When this mode was proposed, the main objective was to find an effective application in Index Tracking – mainly finding a subset of size smaller than C0 (L0-norm constraint limited by ) from an initial portfolio, which behaves as similarly as possible to the initial.

As it can be observed, the norm constrained CVaR provides us with the **L2-norm** component of the L0+L2-norm model, which allows to re-define the L0+L2 model as follows:

s.t.

This allows us to observe the clear division between the L0-norm given by the cardinality constraint and the L2-norm given by the density constraint . These two norms are controlled by the variable . If , the model would behave as a L2-norm constrained model, and = 0 would give use an L-0-norm constrained model only. In [0] Mahesan et al. (2013) the constraint is not taken into account, hence in that case, when , the optimal solution exists when all the values of .

This model is what inspired the main implementation of the feature selection section of this paper. The **L0-norm** component of this model is what makes the time complexity of the algorithm into an NP-Hard problem, and hence why there was a Greedy approach taken in [REFERENCE MAHESAN]. This is the same reason why in this paper we will expand this definition and implement numerous other models with the C0 constraint presented by this model. We will refer to this model as Ridge.

### Absolute Error Model (Abs)

The Abs method is probably the simplest and most effective minimization methods - this is basically a minimization of the absolute sum of errors (L1-Norm). As it can be observed this would be the exact same as a minimization on Index Tracking Error, as the formulations are exactly the same. We will refer to this mode as Abs.

### Least Squares Minimization (Squares)

The least squares minimization is a classic when it comes to Machine Learning. This method is also one of the simplest, most efficient and most straightforward method. This is the same model implemented in Markowitz’ MPT paper [REFERENCE], which is defined as follows:

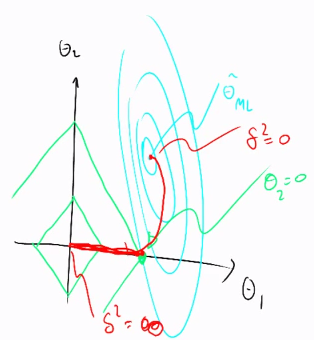
### Lasso Regression

Before proceeding to the introduction of the group regression methods, it is necessary to introduce some core concepts in regards to sparsity model – one crucial one is the Lasso Regression model.

The Lasso regression model is comprised of a sum of squares, plus a scaled sum of the absolute value of the parameters – using our variables for Index, Return portfolio and weights, our model would be defined as follows.

s.t.

At first sight, it may seem very similar to the ridge (L0+L2-norm) model introduced previously, however, what makes this model stand out is the second coefficient – namely the scaled sum of the absolute value of the parameters, which, different from the past model, this coefficient controls the sparsity of our final .

To visualize this we’ll take n=2 as an example, which is pictured in the figure 1 on the right, where the x axis holds the value for our and the y axis holds the value for .

The center of the green diamond represents the point of () when , and the center of the blue circle represents the point when .

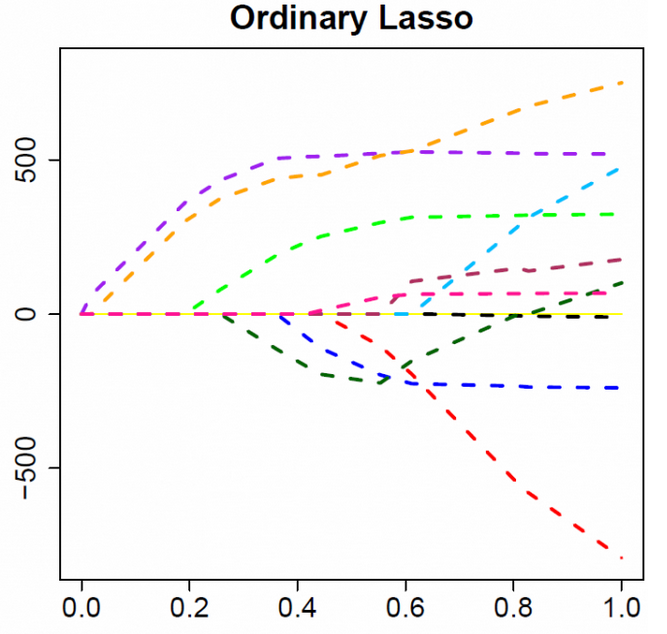
The red path denotes the value and take as changes value. It can be observed that when , will be equal to the maximum likelihood of , denoted in blue by which is the result of the first term of our equation. When , the values of will need to be as small as possible in order to minimize our cost function, which will result in our point in the origin.

Figure 1

By differentiating our cost function and equating it to zero, we find that many of our variables become zero – this is shown in Figure 2

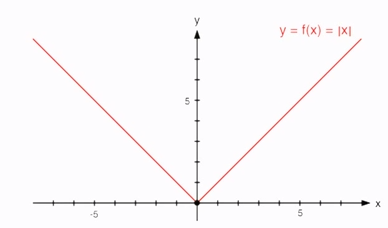
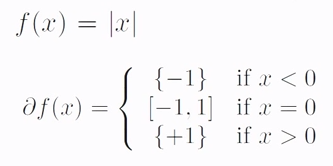
Going back to figure 1 we can observe this behaviour when our blue circle intersects the corner of the green diamond. It is proven that intersections between these two are very likely to happen in the corners – which implies that one of our parameters has taken the value of zero – in this example, our parameter takes the value of zero when the red line intersects the x axis.

Figure

Intuitively, we will build an algorithm that initially takes a guess for all our parameters, then it will proceed to calculate the optimal value of each of the parameters using the (partial?) derivative of our cost function, and repeat this until it converges.

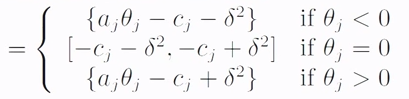
The reason why the last term has not been derived yet is because it consists of an absolute value – for this, subdifferentials will be used. In order to simplify the formulation shown, we will make the following assignments.

This would give us a much simpler formula to expand on its subdifferential, which consist of the following:

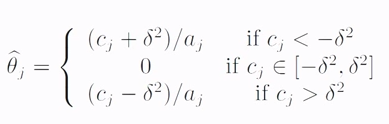
Basically, with a subdifferential, we have a point which cannot be differentiated normally, as we would require a single gradient to explain a subset of gradients – in Figure 3, our point would be the origin, and we would have to define the differential of our formula as:

Figure

So, our formulas would consist of exactly the same format, but instead we would use our variables a, b and as follows:



Now, by setting our values to zero, our estimate can be calculated by setting these differentials to zero for each of the , which gives us the following estimates:



Knowing this estimate now allows us to implement the simple Lasso Algorithm in a very simple manner:

1. **Initialize** (Can be either random, or with something like ridge, etc)
2. **Repeat** until converged
   1. **For** j = 1,2, … , d DO
      1. **if**
      2. **elseif**
      3. **else**

## Model Selection Approaches

### Full Search

### Greedy Search

#### Forward Search

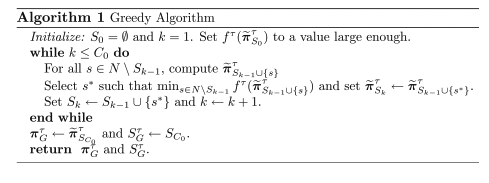
#### Backward Search

#### Forward Search Application

Due to the exponential complexity of the problem in [0] Mahesan et al. (2013) (i.e. to find an optimal subset of stocks from a given Market Index), it was very limited when it comes to even relatively small portfolios. Given that the average Market Index is composed of at least 100 distinct assets, this makes the implementation of a full search is almost impossible.

For this reason, a greedy approach was taken - namely, this was done through a forward model selection algorithm, where one optimal stock is chosen and added to the subset on each step until the subset reaches the desired cardinality.

Formally, the model is formulated as a subset selection, where each step, a locally optimal asset is chosen and added to the portfolio. The algorithm proposed in [0] Mahesan et al. (2013) is converted into a subset selection problem defined as follows:



To apply this algorithm, we begin with an empty set, then for all the available stocks, we add one and run the L0+L1-norm model on each. We then calculate the tracking error for each, and select the asset with the lowest tracking error as our local optimal. We do this until the cardinality of our set reaches the L-0 constraint C0.

## Group Regression Approaches

Now that the most important key points have been introduced in the previous sections of the background research, it is possible to proceed to explain the approaches that have been considered in this paper to implement group selection in sparse and non-sparse regression models.

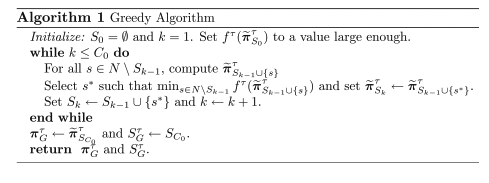
### Model Selection Approach

Before considering complex group regression models, this paper considers an approach based in the approach in [0] Mahesan et al. (2013) that initially inspired this research project – mainly it is being referred to the forward-search approach that was taken to select a subset of stocks from a Market Index with the lowest tracking error.

As it was revisited in the last section, the approach taken in [0] was a greedy, forward-search approach, and this was the choice over a full-search approach due to the NP characteristics of the subset selection problem. Given that stocks are considered individually, and the Market Indexes vary from 100 to 500 stocks, the time complexity of this problem would be exponential on the number of stocks to consider, which would not allow for a full-search approach.

This section proposes to reduce this individual stock subset selection problem reduces to a group subset selection problem, and assuming that our group number does not consists of a large amount of groups (e.g. more than 20), a full-search approach is actually possible, as we could have a full subset selection on our groups.

Similar to the previous algorithm, we can select any regression model and implement a full-search group subset selection as follows:



Similar to results presented in [0], with this full-search algorithm, we would be able to select a subset of financial instruments from a market index based on prior knowledge on the instruments that allowed us to create the groupings.

### Group Lasso Regression

A problem with the approach taken in the previous section is that model selection does not allow for varied weightings on each of the groups – instead it can only either include or exclude them completely.

#### Simple Group Lasso

Now that the reader has obtained a core knowledge on the characteristics and functionality of the Lasso model and its components it is possible to expand the definition into groups. For this, we assume that the variables are clustered in groups, and instead of having a penalization on the sum of absolute values, the values of Euclidean norms of the parameters in each group are used. What this does is drive all values in each group to the same exact value, so with this, we would have a lasso model which induces sparsity within groups.

Group lasso was proposed in a very interesting paper by Yuan & Lin (2007) [REFERENCE]. Recalling the mathematical group notation introduced in the beginning of this paper, this model has been adapted to our financial implementations, which is defined as follows:

Intuitively it can be observed that this model follows the same methodology than the individual Lasso model presented earlier – the only difference is that this formulation acts on data as groups of stocks, as opposed to individual stocks. If the group size for all groups is one, this problem would reduce to a simple individual Lasso model [REFERENCE NOTE ON GROUP LASSO]

Similar to the Lasso formulation, if the value of is too high, group sparsity would be enforced in our variable making all the groups approach to zero.

#### Sparse group lasso

The only problem with the simple group lasso introduced previously is that all the features in each group have equal values, which means that there is no sparsity within the elements inside a group. What this means is that if one group of features is zero, then all the features in the group would also be zero as well.

An innovative model is proposed in [REFERENCE A PARSE GROUP LASO] which not only provides the sparsity between groups, but also provides sparsity within individual levels. This is known as the Sparse Group Lasso, which is defined as follows:

s.t.

As it can be observed, the only difference in this formulation is the last term included, which is simply the L1-norm regularizer (sum of absolute values) found in the simple single-feature lasso model. This term, together with our variable which balances the ratio between which this regression model considers group sparsity and individual feature sparsity.

This model is revisited again, and a new approach is given in [REFERENCE A NOTE ON GROUP] where another simple implementation of the Sparse Group Lasso is proposed. In this implementation, the coefficient is dropped, allowing us to introduce and :

It should be mentioned that from our practical experience, the best results are obtained when contains a scaled value of (Hence the value of , however, for the sake of simplicity, and for being able to offer the reader with a clear difference for when we refer to either the Group Sparsity inducing coefficient () or when we refer to the Feature Sparsity inducing coefficient (). We will instead use the definition where will define the scale factor.

## Zero-Constrained Group Lasso Application

The biggest challenge in this paper was to come up with a formula that does not require the inner loops that most implementations require as the machine learning library used for the implementation cannot process the inner loops required for cofactors such as and **.** In order to come up with a solution, a new constraint was implemented which allowed us to solve this equation with only matrix multiplications – this constraint is explained below.

For this section we will require the variable introduced in our mathematical notation where contains the indexes of for the stocks in group g – which are the stocks contained in . With this, the formal notation should also be intuitive **.**

For the sake of simplicity we will be using the function sum(x,i) throughout our implementation. This function was introduced in the first section of this paper, and outputs the sum of the rows (i=1) or columns (i=2). The sum of the columns or rows of any matrix can be represented with a matrix multiplication of a column or row vector of ones respectively, however, this function will be used for making the explanation simpler.

Initially we have to replace the first summation, namely , for a cofactor that does not require this loop. Our solution proposed is , where where . In simple terms, can be seen as an n by m (Number of stocks by number of groups) matrix, where each row contains the weights for a specific group in its respective index position and the rest of the elements are zeros.

To provide a more clear understanding on this, lets assume , we have m=2 groups where which gives us and These definitions allow us to build as:

With this we can relate our new introduced terms to the previous implementations of the group lasso – it should be obvious that . It should also be noted that in this example . We will need this to throughout the execution of the algorithm.

Now that we obtained a way to represent this cofactors with only matrix multiplications, we need a constraint that ensures that always **.** In other words, to make sure that all the elements in our variable are always zero if they do not belong to their respective group. Following this definition, we specify to be the set containing the indices of the elements that are **not** in group g. Formally, we define **.** This allows us to introduce the constraint that makes this implementation possible, which is:

The only thing left now is to make the term suitable for our operations. It can be noticed that the only thing we will require now is a way to calculate the L2-norm of each of our individual vectors (rows of ), as currently our function returns a scalar which represents the L2-Norm of the matrix, however we need the L2 norm for each of the individual groups. This is a very simple thing to do, which can be achieved with the formula . This allows us to transform our second cofactor into:

sum(\*)

With this new constraint, and this new second term, it is possible to introduce the Group Lasso and Sparse Group lasso models that will be used throughout this paper:

s.t.

Which is equivalent to the lasso formulation presented earlier, however, all the computations are all done through matrix multiplications, which makes it suitable for our implementation.

# Implementation

This section assumes that the reader has an appropriate understanding on most topics covered in the background research, and contains the implementation of the models and algorithms presented in the previous section.

This section begins with a description of the developing environment in which the code snippets were ran and tested, as well as information on the mathematical libraries used to obtain such results. This section will then follow with the implementation of the feature regression models, and finally the group regression models.

As mentioned in the introduction, the working code for this section has been registered on a public Open Source licence, which at the time of writing is being hosted in GitHub, and can be downloaded at http://github.com/axsauze/sparse.

## Working Environment

All the experiments in this paper were ran in a MacBook pro with OS X Mountain Lion 10.8.5, Intel Core i7 2.9 GHz processor, with 8 GB 1600MHz DDR3 SDRAM.

All code is written in Matlab R2014a, and two main libraries were used for linear regression computations – namely CVX and SPAMS. Details on compilation and installation of these can be found in the main repository.

### CVX

CVX is a Machine Learning library for Matlab that provides modelling tools that allow the programmer to specify constraints and objectives using standard Matlab code. This library allows great flexibility when it comes to solving simple regression problems such as least-squares, weighted regression, ridge regression, etc.

The CVX library would allow us to translate simple regression problems as follows:

|  |  |
| --- | --- |
|  |  |

### SPAMS

There are very few implementations for accurate group selection models. The Sparse Modelling Software (SPAMS) library released in 2010 is one of the few libraries that provide a very wide range of sparse regression tools. Unfortunately the library is compiled, and has a lot of dependencies, so it’s more of a black box. The implementation of these models in financial data also have not proven to be as accurate as the CVX modelling toolbox, however, it is very worth mentioning this library, as its tools helped on obtaining an insight while working on this paper. These tools are based on a paper released by the same creators of this library [REFERENCE OPTIMIZATION SPARSITY], which contains in-depth detail on all of the sparse regression models included in this library, together with several algorithmic approaches and optimization methods.

Although this library takes several approaches to achieve convergence in its constituent regression models, Lasso related models utilize an algorithm devised in [REFERENCE FISTA] which comes from the ISTA (Iterative Shrinkage-Thresholding Algorithms) family of algorithms – namely the FISTA iterative method. This method is used in signal image processing due to its much greater converging speed.

## Implementation Design

### DataSets

Experiments are carried out on various sets of data in order to provide more robust and reliable results. Sets of data consist of both, real and stochastic financial data.

Both datasets used in this implementation consist of a financial time series containing the returns of a specific Market Index, together with the individual price of its constituent stocks.

#### Market Financial Data

In this paper, the Market Index to be analysed will be the FTSE100, together with the n=100 stocks that comprise it. The time window to be analysed will consist of T=274 working days, starting from 1st of November, 2011, until the 1st of December, 2012. This data is retrieved using the Yahoo finance API.

#### Stochastic Financial Data

Stochastic data will have the same format as the real financial data – namely a set of n=200 stocks, with it’s Index Value for a time window of T=300. Thorough research was undergone to simulate Market Index value behaviour. Three approaches were taken, namely naïve, GARCH and ARIMA – these were discussed briefly in the 2.Background Research section, and will be implemented in this section.

### Data grouping

Groupings of data

#### Market Financial Data

#### Stochastic Financial Data

KNN,

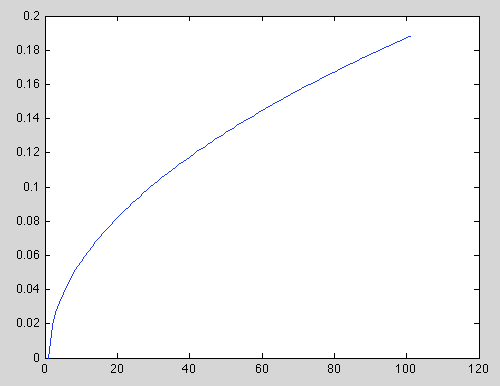
### Financial Mathematical Notation

Expanding our initial mathematical notation towards our implementation in all our computations, we will have an input data **D** consisting of a Matrix containing the historical price for n = 1, …, n assets at time t = 1, …, T, as as well as our constants as required.

The first thing that has to be done is to convert this matrix of historical prices into a matrix of historical returns **R**, which can be done simply by creating a matrix of column length T-1, and assign the value to each column for t = {1, …, T-1}.

In order to obtain our index, we have where is a proportionately distributed percentage of capital invested on each asset. This way, we obtain a proportionately distributed portfolio containing all the assets in the specific Market Index.

## Core Financial Formulations

With our current portfolio it is possible to calculate the Value-at-Risk. To be able to show the results in a graph, we will use the function proposed in [6] Mina J. et al. (2001) computation can be expanded to give us the VaR over time with by adding the variable T to the formulation, where T represents the number of days from today, and the result would be the amount that we would be β% sure it would not be exceeded:

The figure above shows a common application of Value-at-Risk, and how it behaves against time. However, as it was mentioned previously, the lack of convexity of the Value-at-Risk model does not allow for simple optimizations, so this model itself cannot be used for this purposes of portfolio optimization.

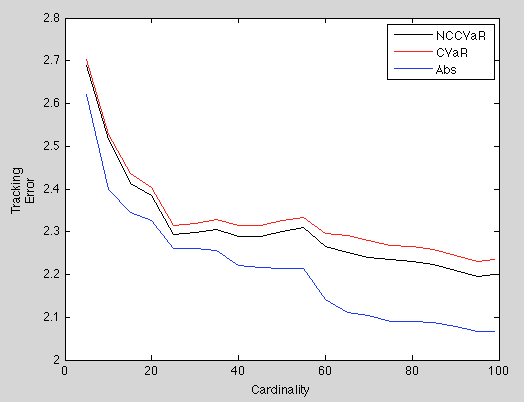
As it was mentioned previously, the accuracy of all the results, which mainly will consist in the parameter **π** to be learned, will be measured against the our index **I** with the Tracking Error formula introduced in the previous section, such as the offset between **I** and **Rπ** will define how accurate the results are.

## Simple Regression Models

It is now possible to test the L2-norm constraint under the same circumstances that were carried out in [5] Takeada A et al. (2000). The main objective of this previously mentioned paper is to show not only that the Norm Constrained Conditional-Value-at-Risk minimization is equal to an SVR minimization, hence ensuring optimal values, but as well showing that the NCCVaR is as reliable as CVaR minimization. For this, the NCCVaR minimization will be compared to the CVaR minimization proposed in Optimization of Conditional Value At Risk and the minimization on the absolute tracking value error, which will be referred to as abs.

To proceed, we will use the NCCVaR, CVaR and Abs to obtain their respective parameters **π** and compare their Tracking Error offset between one another. The lower the Tracking Error offset, the more accurate it is. Our minimization formula requires two input parameters that will need to be tweaked in order to obtain reliable results – these are namely β and C2.

The following task is to find the best value for these two variables in order to obtain reliable results. [Takeda 2010] proposes a value for β between 0.1 and 0.9, and C is chosen from , where k can be between 1 and 5. After carrying out several tests for a wide range of values between these two, it was discovered that the optimal value for β is 0.3 and for k is 5.

We executed each of the models for subsets of assets from the model chosen at random with cardinalities {0, 5, 10, …, 95, 100}. We then calculated the Tracking Error between the Index portfolio and the Returns matrix multiplied by our learned parameters **π**. This gave us the results displayed in our figure to the right, in which we can observe the behavior of the Norm Constrained Conditional Value at Risk in black, the Conditional Value at Risk in red and the Absolute Tracking Error in blue.

As it can be observed, the tracking error is very low in all of the cases, and although Abs is slightly more accurate than the NCCVaR and CVaR, we can see that the Minimization on NCCVaR is extremely reliable, and can be used to expand for numerous applications. To which, one of our next steps is the individual feature selection models.

## Sparse Inducing Models

Given that with normal regression models it’s not possible to obtain subsets of data by simply modifying constraints, sparse inducing models are an attractive choice to force these characteristics. Now that the theory was covered in the background research, it is possible to take a practical perspective.

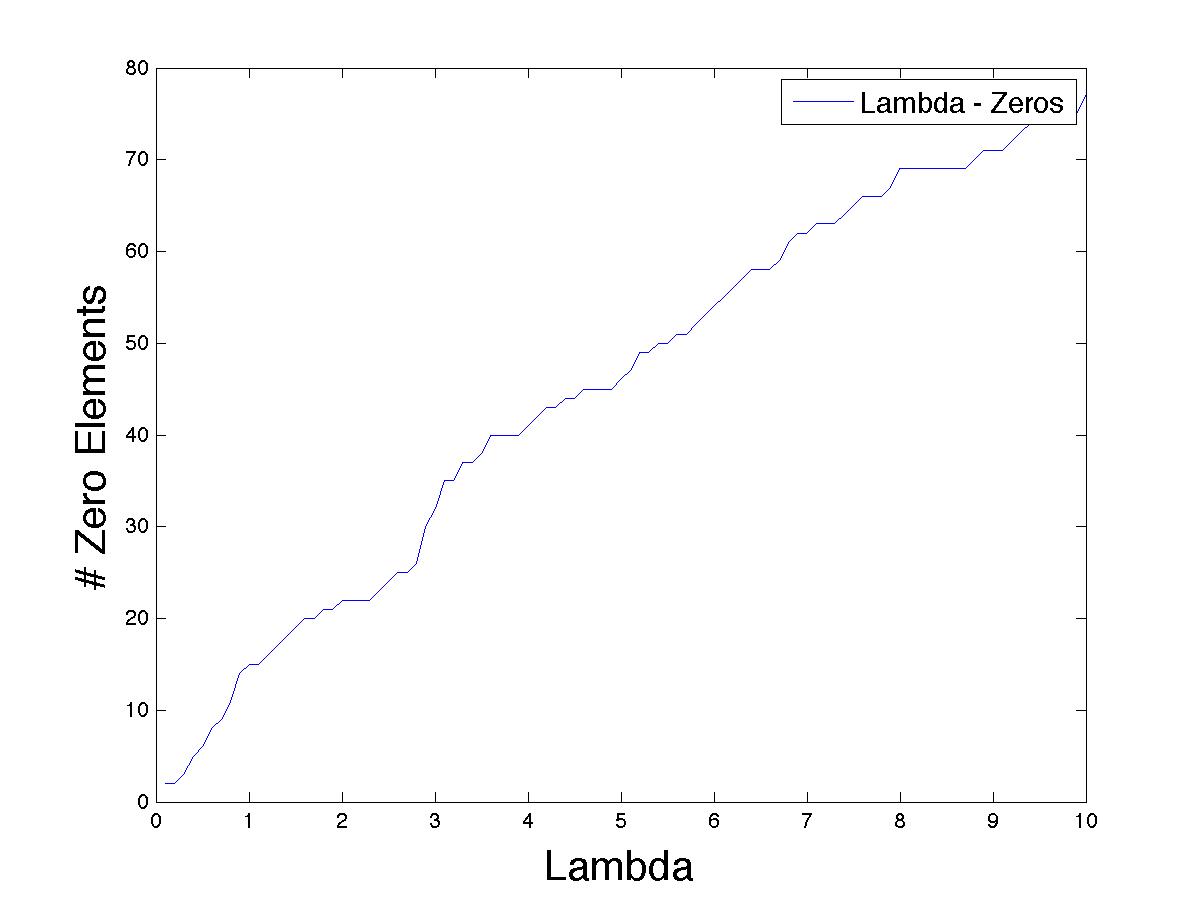
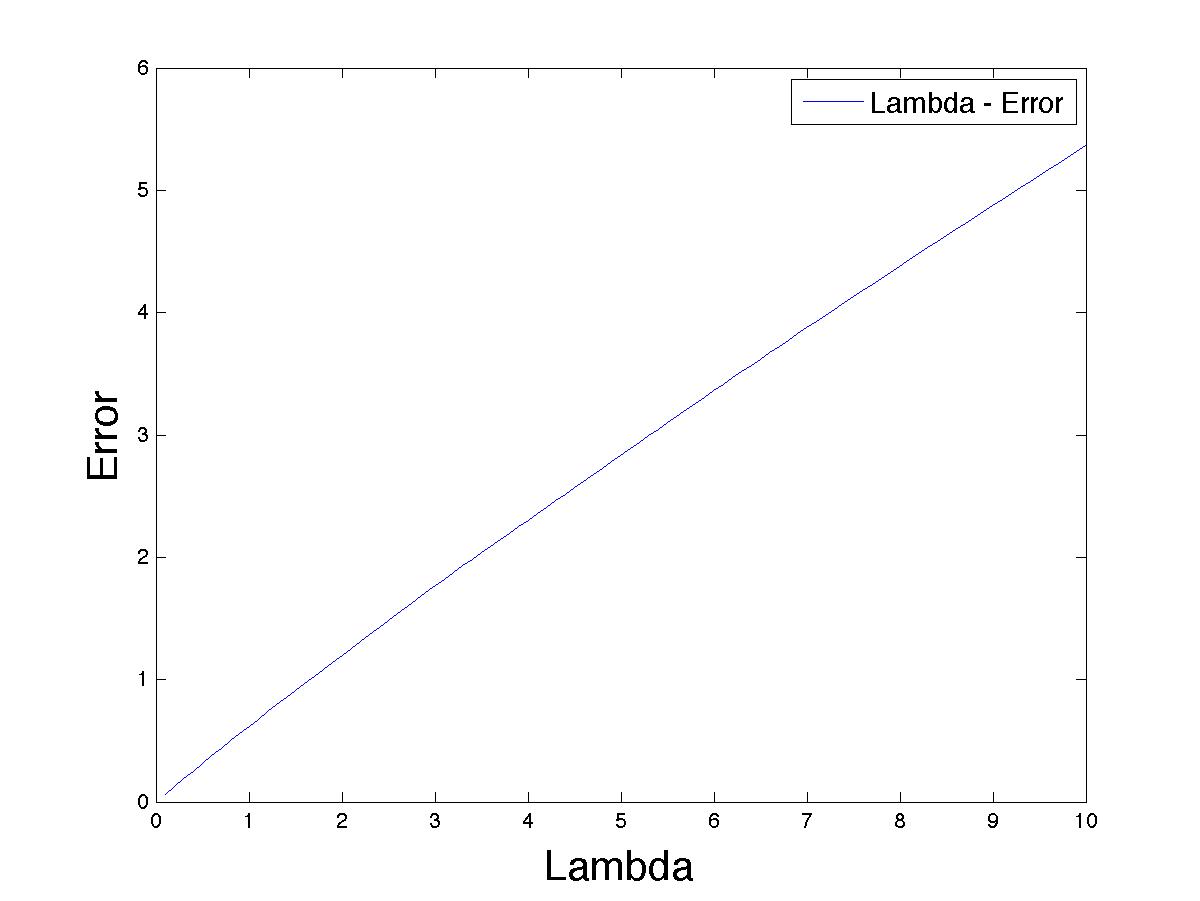
Most of these sparse models induce sparsity through a penalty coefficient consisting of an absolute sum of the weights, which push less important features towards zero values. This penalty is regulated through a coefficient, namely lambda, which controls how the magnitude of this penalization on the sum of weights.

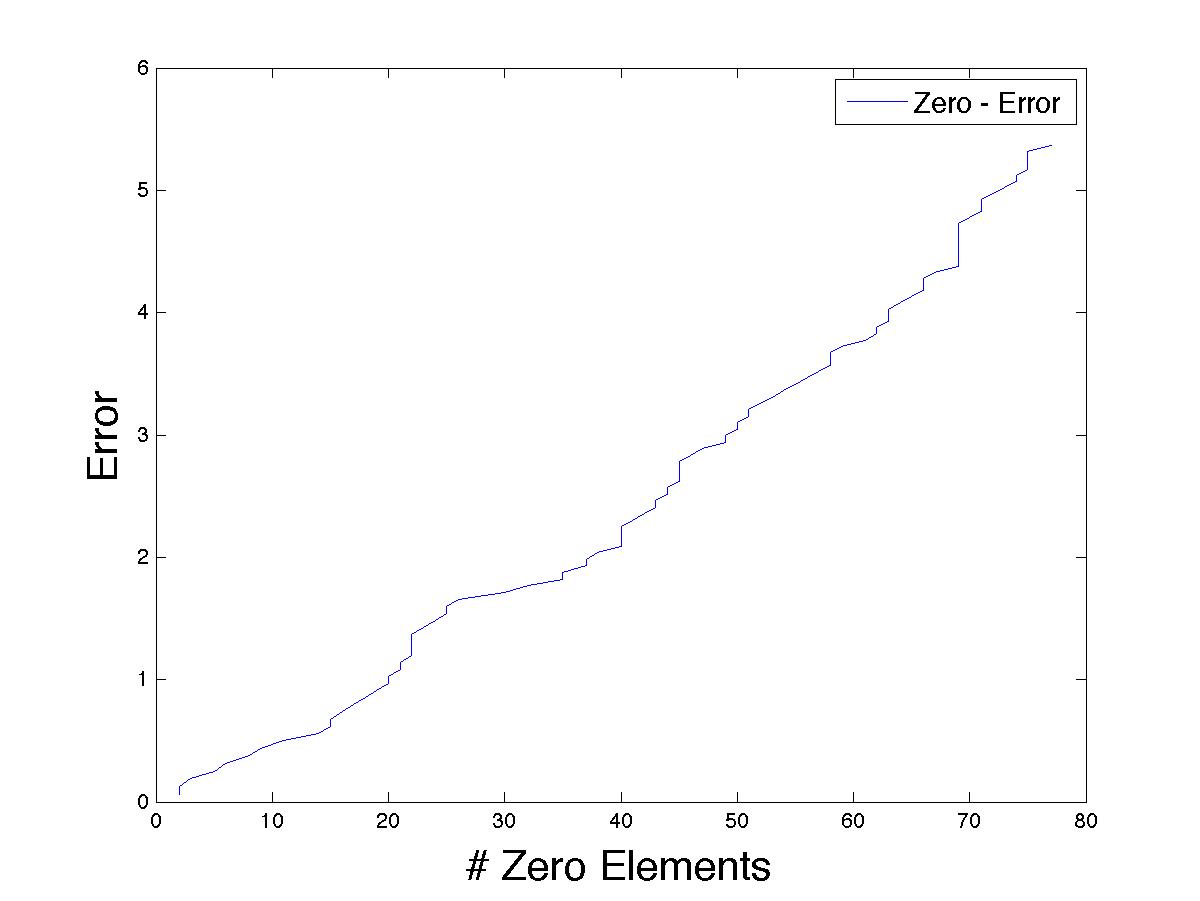
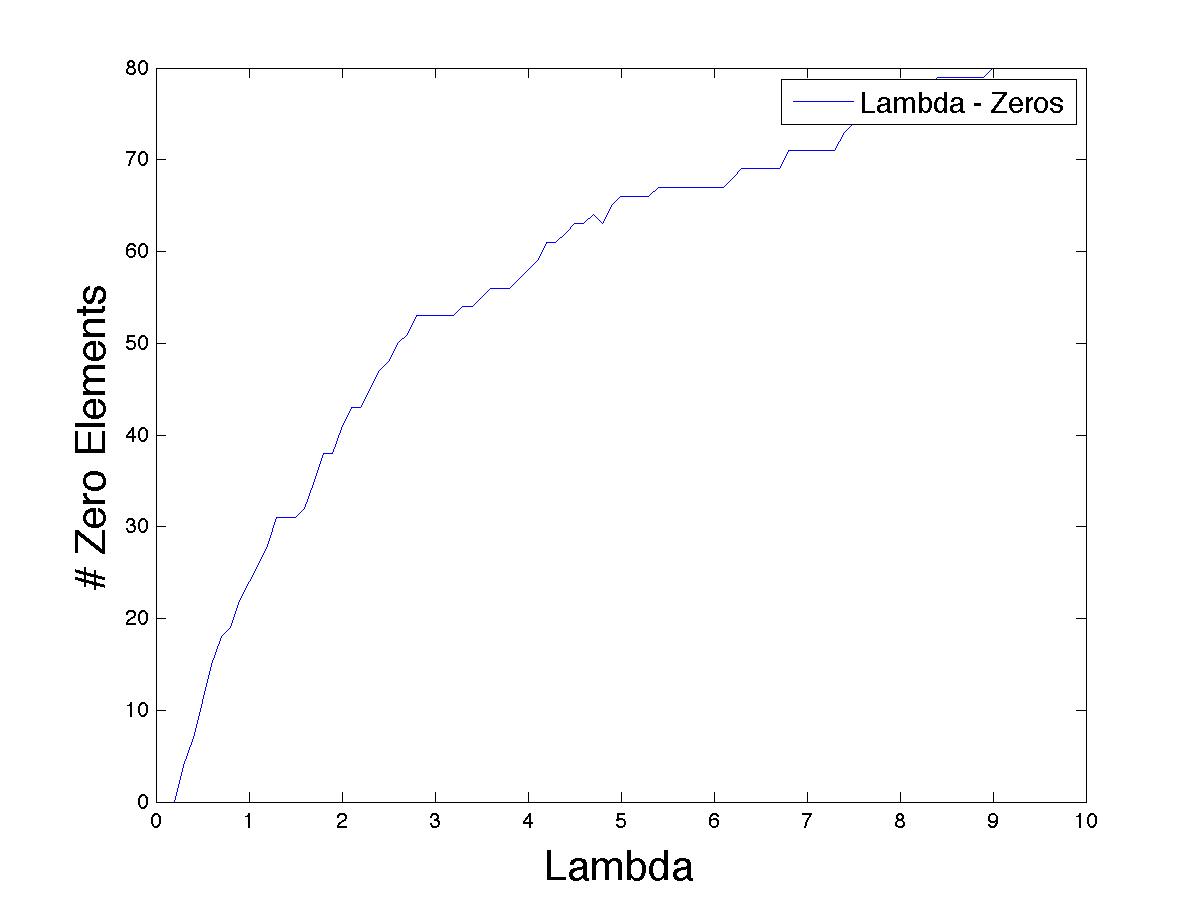
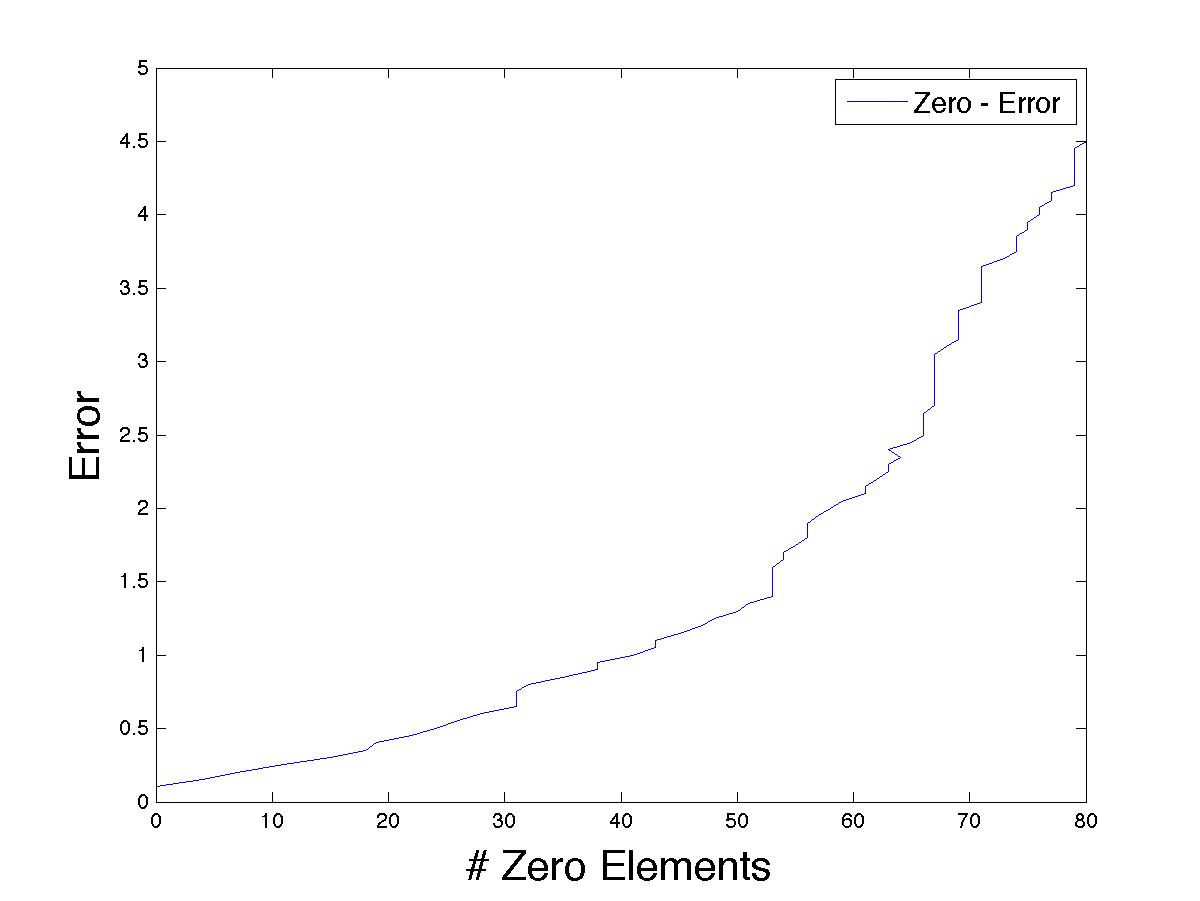
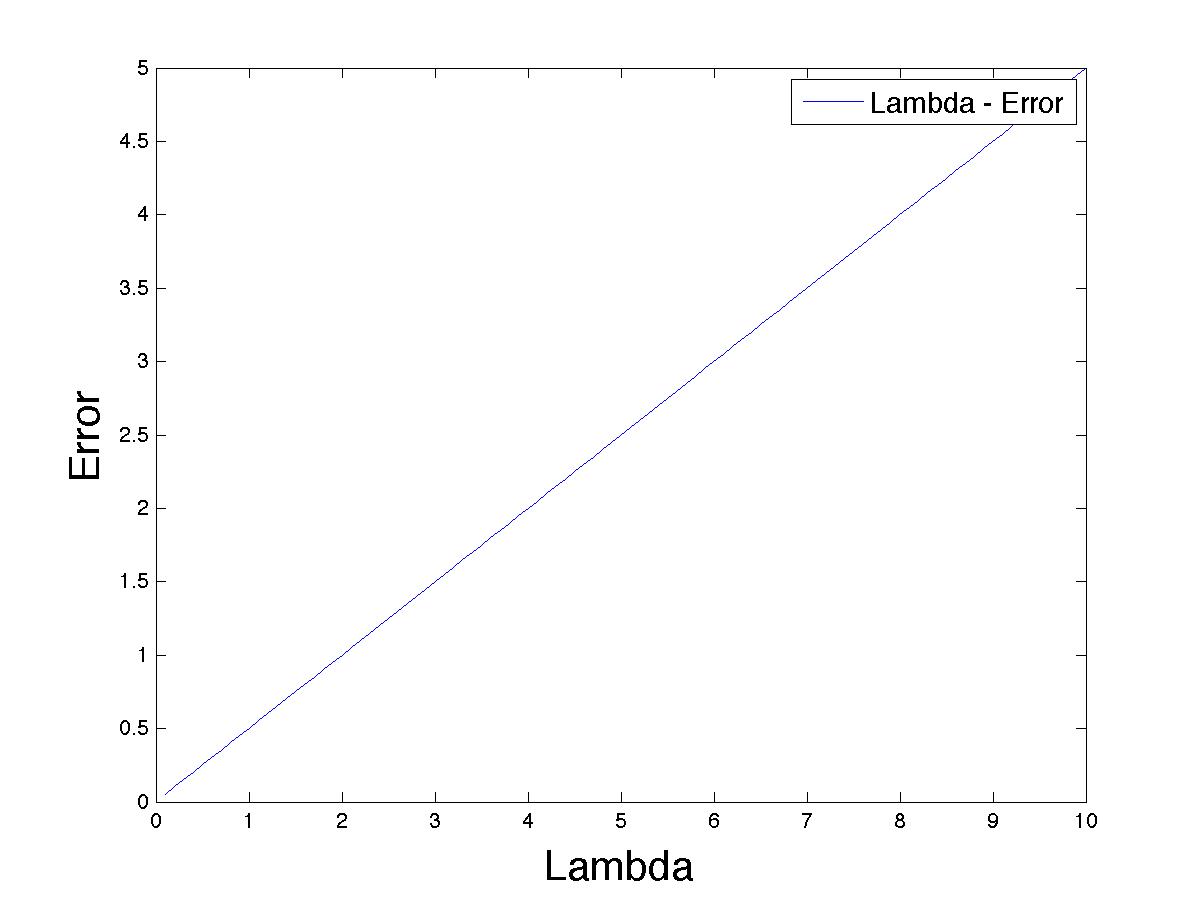
The objective of this section is to collect observations of the effects of different values for this weighting coefficient to induce sparsity in our data, and to observe how efficient this solution is to achieve better goals.

### Simple Lasso model

The simple Lasso model in this paper is solved through two different iterative approaches – the LARS, FISTA and several primal and dual approaches, which are implemented by the CVX library that is used for these calculations.

In order to obtain a specific amount of zero values in the vector of weights it necessary to find the exact values for lambda in the Lasso regression formula. In this section, the Lasso model is implemented to both, our stochastic and market data for increasing values of lambda to observe the tracking error as the number of zero values increases.

Stochastic Data:



FTSE100 Data:

The results shown in figures [FIGURE] through [FIGURE] previous graphs were as expected in terms of their behavior (i.e. a logarithmic increase in the number of zeros as delta increases, as well as an exponential increase of the tracking error as the number of zeros increases), however what should be noticed, and will be revisited in more detail in the following section is the high tracking error values, which seem to increase much faster than in our previous section.

Likewise, numerical results emphasize these characteristics that were just mentioned. The numerical values in figure [FIGURE] have been reduced to only 10 iterations only to provide a practical perspective on results of sparse model applications, however, the experiments, including the previous graphs were ran with 100 iterations.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stochastic Data | | | |  | Market Data | | | |
|  |  |  |  |  |  |  |  |  |
| **Lambda** | **# Zeros** | **Tracking Error** | **Elapsed** |  | **Lambda** | **# Zeros** | **Tracking Error** | **Elapsed** |
| 0 | 0 | 0.0003 | 15.3751 |  | 0 | 0 | 0.0003 | 2.8217 |
| 1 | 15 | 0.6186 | 9.3624 |  | 1 | 24 | 0.5025 | 1.7852 |
| 2 | 22 | 1.1998 | 10.1146 |  | 2 | 41 | 1.0007 | 1.9100 |
| 3 | 32 | 1.7666 | 8.6064 |  | 3 | 53 | 1.5004 | 2.1008 |
| 4 | 41 | 2.3039 | 8.6202 |  | 4 | 58 | 2.0003 | 1.8818 |
| 5 | 46 | 2.8342 | 8.4283 |  | 5 | 66 | 2.4998 | 1.7930 |
| 6 | 54 | 3.3626 | 9.1818 |  | 6 | 67 | 2.9997 | 1.8571 |
| 7 | 62 | 3.8769 | 8.4577 |  | 7 | 71 | 3.4996 | 1.8690 |
| 8 | 69 | 4.3807 | 8.4173 |  | 8 | 76 | 3.9995 | 1.8775 |
| 9 | 71 | 4.8779 | 8.4079 |  | 9 | 80 | 4.4993 | 1.8985 |

## Individual-Feature Selection

Now that several regression models have been applied to several subsets of data, it is possible to expand their application in our individual feature selection approach, where a forward selection model is implemented through a greedy approach to select a subset of stocks form a Market Index that minimizes Tracking Error – this is, that behaves as close as possible to the original market index.

The reason why a greedy approach is used is due to the exponential nature of the problem – it is necessary to select an ‘optimal’ combination of stocks from a set of at least 100 stocks (The average minimum for a financial Market Index). After this approach was implemented, it was confirmed that even small subsets require great amounts of time and computation.

### Algorithm

The algorithm used in this section is discussed in the 2.Background Research section, and the implementation can be found online in the repository. Briefly, the algorithm consist of a forward subset selection, where we initially begin with an empty subset of chosen stocks, we consider every single stock that is still available by adding them one by one and calculating the tracking error by using any regression model. The local-optimum will be chosen by selecting the stock that generated the smallest tracking error, and the algorithm iterates until there are no more available stocks to chose from, or the objective cardinality of our subset has been reached.

From the previous section, it was observed that in any situation, whether with large or small Market Index subsets, the most precise regression model (i.e. generated the lowest tracking error) is the ABS method. Although the ABS method was selected as the main model for individual-feature selection, the code implementation of this paper was written so that it is easy to choose the main regression model from a different range of models, which means that any regression model can be used.

It’s worth pointing out that the results in this section were limited to the time complexity of the problem, as although the greedy approach taken makes this problem possible to solve, it still shows extremely long execution times even when dealing with small sets of data.

### Feature-Selection Implementation

The algorithm was implemented by using the four main regression models in this paper. The greedy approach allowed us to choose subsets of stocks through local optimal decisions based in tracking error. In average, the time taken to select the locally optimal stock was 29.4115 seconds for our Market Data (100 features and 274 observations) and [REFERENCE (not reference, but just in case :P)] seconds for stochastic Market Data (200 features and 1000 observations).

Although it’s not possible to achieve a full search algorithm, and the execution time of this algorithm is extremely slow, the results were impressive in terms of the accuracy of the Index Tracking. As it can be observed in the results from stochastic data (figure [FIGURE]) and FPSE100 data (figure [FIGURE]), the values for Tracking Error are extremely low relative to all previous results.

|  |  |
| --- | --- |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:feature_selection:stochastic_feature_selection_70.jpgStochastic Data | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:feature_selection:ftse100_feature_seleciton_zoom.jpgMarket Data |

Once again it’s reassured that the abs regression model provides the best Index Tracking results, followed by least squares, CVaR and finally Ridge Regression. Our numerical results shown below in figures [FIGURE] and [FIGURE] also show a very low tracking error compared to our other implementations. This is also shown in the summary of data in the tables below.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Tracking Error against Non-Zero Elements.** | | | | | | | | |  |
| Market Data (n = 100) | | | | |  | Stochastic Data (n = 200) | | | | |
| **Non-Zero** | **Abs** | **Squares** | **Ridge** | **CVaR** |  | **Non-Zero** | **Abs** | **Squares** | **Ridge** | **CVaR** |
| 1 | 1.6318 | 1.6318 | 1.6318 | 1.6318 |  | 1 | 5.3229 | 5.3229 | 5.3229 | 5.3229 |
| 7 | 0.5626 | 0.5856 | 0.5889 | 0.5922 |  | 12 | 4.1574 | 4.1659 | 4.6134 | 4.2446 |
| 13 | 0.3698 | 0.3968 | 0.4001 | 0.3897 |  | 23 | 3.4500 | 3.4780 | 4.0713 | 3.5702 |
| 19 | 0.2869 | 0.3087 | 0.3258 | 0.3042 |  | 34 | 2.8956 | 2.9539 | 3.6773 | 3.0781 |
| 25 | 0.2297 | 0.2589 | 0.2716 | 0.2515 |  | 45 | 2.5234 | 2.5609 | 3.2113 | 2.6830 |
| 31 | 0.1862 | 0.2134 | 0.2360 | 0.2222 |  | 56 | 2.1621 | 2.1958 | 2.8352 | 2.3756 |
| 37 | 0.1577 | 0.1839 | 0.2028 | 0.1894 |  | 67 | 1.8357 | 1.8781 | 2.5363 | 1.9910 |
| 43 | 0.1368 | 0.1548 | 0.1736 | 0.1731 |  | 78 | 1.5455 | 1.5954 | 2.2396 | 1.6544 |
| 49 | 0.1178 | 0.1338 | 0.1513 | 0.1574 |  | 89 | 1.2690 | 1.3086 | 1.9591 | 1.3769 |
| 55 | 0.0992 | 0.1156 | 0.1336 | 0.1517 |  | 100 | 1.0461 | 1.0838 | 1.6627 | 1.1712 |

In this previous two tables we can observe the tracking error when the first half of the elements in our market index are chosen. The tracking error shows to be extremely low compared to our other approaches, which proves to be a great method – this, without taking into account the huge time required for the computations, which will be discussed below

Before mentioning time complexity, it was very interesting to observe the behavior of each approach in terms of the stocks chosen, and the order in which they were chosen throughout the execution. For this we calculated the size of the union of each of the subsets of the four methods at each iteration of the algorithm. To make this clear, if on our 3rd iteration the stocks chosen are Abs={1,2,3}, Squres={2,3,4}, Ridge={2,3,5}, CVaR={2,3,6}, then the union of this subsets is {2,3}, which means that there are two stocks that have been chosen in all subsets. This is shown in figure [FIGURE], where it is possible to observe that for all our iterations, at least half of all the subsets consisted of the same stocks, and given that we are considering the four sets at the same time (rather than just comparing pairs), this truly reassures that some stocks really have much more powerful tracking potential than others, and there really is an optimal subset that can be found from subset selection models.

|  |  |
| --- | --- |
| **Percentage of Same Elements in Subsets** | |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:feature_selection:stochastic_zeros_same.jpgStochastic Data | Market Data  Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:feature_selection:ftse_zeros_same.jpg |

Something that really needs to be mentioned is the time taken for computing these results. Although the greedy approach taken made the computations of this section possible, it still does not account fully for the huge time complexity of the problem. The Market Data only consisted of 100 elements, and it took over one hour to compute – the stochastic data, in the other hand, with just a number of elements twice as large as our market data, took over 2 days to compute. Given that 100 is the minimum number for a market index, this approach would be infeasible in the real financial world when speed is a real concern.

## Group Models

Group models is one of the core topics in this paper, as it aims to find the effects of using underlying knowledge of the financial instruments to consider these as groups. There are several approaches in this section to test the effectiveness of this approach, and whether this is a better approach than the ones taken previously.

Similar to our previous section, a model selection approach will be taken, where optimal solutions are believed to be reachable assuming that grouped data does not exceed a number greater than 20 groups. The reason for this is because a full-search approach is taken, rather than a greedy one, as a full-path search is reachable when the number of features to select from are less.

This section is approached in several different ways. Groups and clusters of data are selected in various ways. Market Industry and Market Sector of the individual stocks are important choices to consider in the implementation, however it is also important to consider other approaches that can allow us to group the stocks without prior knowledge or information on these. We assume that groupings in data contain correlations in some way – for this, we use a several clustering methods to find such groupings – our main approach is through spectral clustering based in K-nearest neighbor and K-means. The spectral clustering implementation gives us the ability to find clusters of data based on a correlation matrix computed for each dataset. These groups are then taken as inputs, and data is then processed with these assumptions.

### Group Selection

The method proposed for the individual feature selection in the previous section is expanded into group selection, where instead of selecting individual features, groups are selected based on the training error of each chosen subset.

Initially, stochastic and market data are compared side by side – groups are chosen through our spectral clustering approach with a number of 5 and 8 groups, which would simulate the groupings that would be generated through prior knowledge (e.g. industry type, sector, etc).

This approach is somewhat limited due to the exponential nature of the full-search approach taken to find the optimum solution for this group selection problem. This limitation however should not be a problem in this implementation, as there should not be cases where we encounter numbers of groups greater than 15 (as greater numbers would result in an impossible number of iterations required). If this is the case, other approaches must be considered.

|  |  |
| --- | --- |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_selection:stochastic_zero_error.jpgStochastic Data (5 groups) | Market Data (5 groups)  Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_selection:ftse100_zero_error.jpg |

|  |  |
| --- | --- |
| Stochastic Data (8 groups)  Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_selection:stochastic_zero_error_7groups.jpg | Market Data (8 groups)  Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_selection:ftse100_zero_error_7groups.jpg |

Like in previous sections, the numerical data displayed was limited to only a subset of the full results, as this is enough to show the most important details, which include the Tracking Error dependent on sparsity of the subset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stochastic Data (8 Groups) | | |  | | Market Data (8 Groups) | |
|  |  |  | |  |  |  |
| **# Zero Elements** | **Error** |  | |  | **# Zero Elements** | **Error** |
| 0 | 0.0000 |  | |  | 0 | 0.0000 |
| 21 | 1.2303 |  | |  | 11 | 0.0615 |
| 42 | 1.7481 |  | |  | 20 | 0.1071 |
| 66 | 2.1601 |  | |  | 30 | 0.1293 |
| 90 | 2.6477 |  | |  | 40 | 0.1745 |
| 105 | 3.0955 |  | |  | 53 | 0.4213 |
| 110 | 3.2069 |  | |  | 60 | 0.4910 |
| 134 | 3.5557 |  | |  | 70 | 0.4716 |
| 152 | 4.2498 |  | |  | 80 | 0.7315 |
| 179 | 4.6964 |  | |  | 90 | 1.1174 |
| 197 | 6.3528 |  | |  | 97 | 1.7549 |

What should be noticed from this graphs is the aggressive variations in Tracking Error, which seems to sometimes decrease even when a higher number of zeros is present. Given that this algorithm takes full-search approach rather than a greedy approach, it evaluates every possible combination of groups. Each jump in the data line represents a specific combination of the groups available.

Even though the tracking error is slightly worse, the faster execution times of this algorithm compared to the feature selection approach might make this model an interesting choice to consider for portfolio tracking.

## Sparse Group Models

This following sections contain the implementation of sparse group models in financial data.

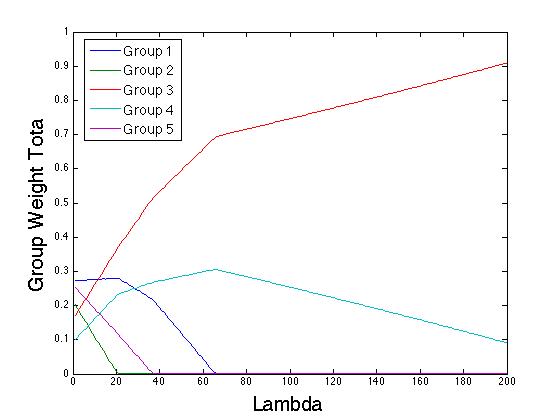
### Group Lasso

The group lasso provides exactly the same sparsity effects than its equivalent single-feature counterpart – the difference is that the sparsity of this model is applied to the groups as a whole, rather than to each individual feature.

The algorithm implemented is described in the Background Research section, which was perhaps the most challenging section in this project. The algorithm was implemented with some help from the CVX toolbox which provides the iterative algorithm to solve the equation provided.

The solution proposed proved to be more accurate than the closed-source solution provided by the SPAMS toolbox – however, results from this library are also displayed further.

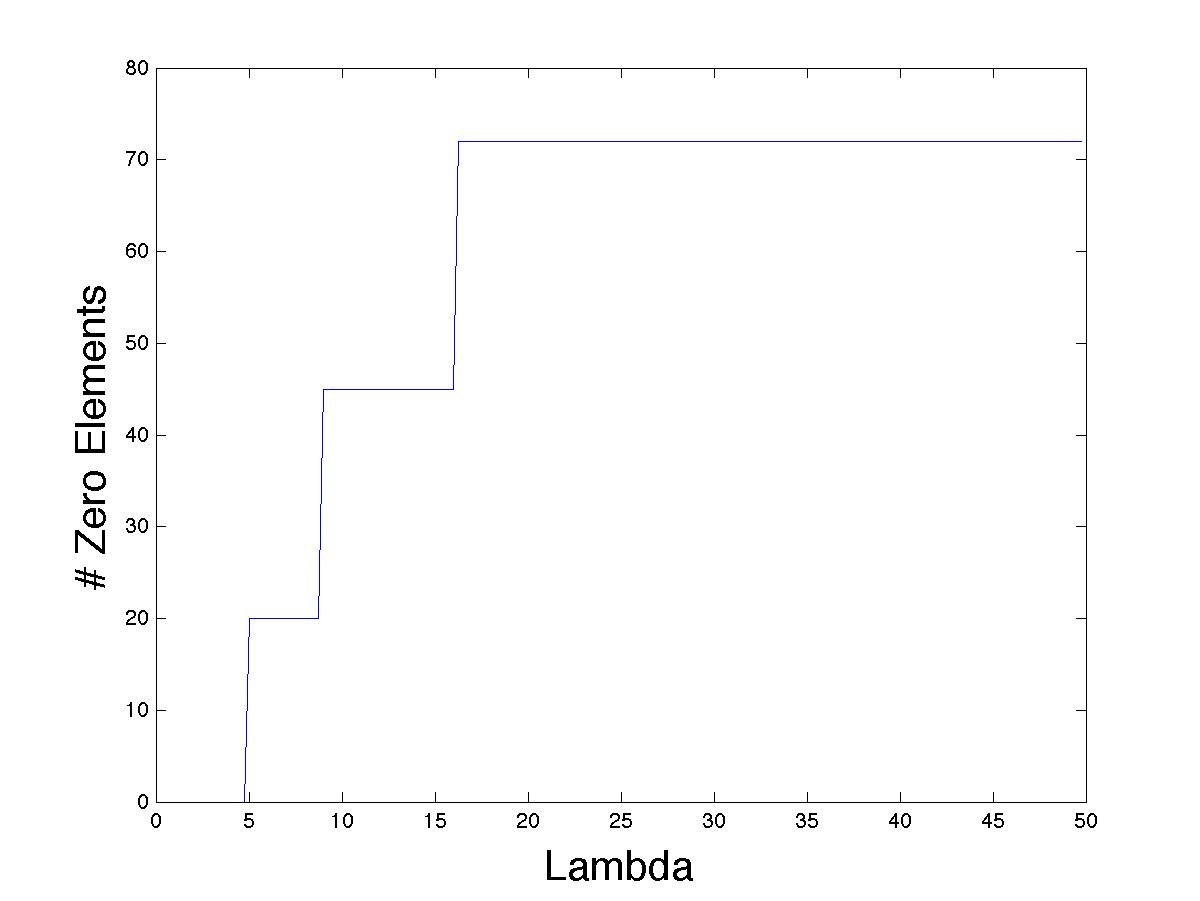
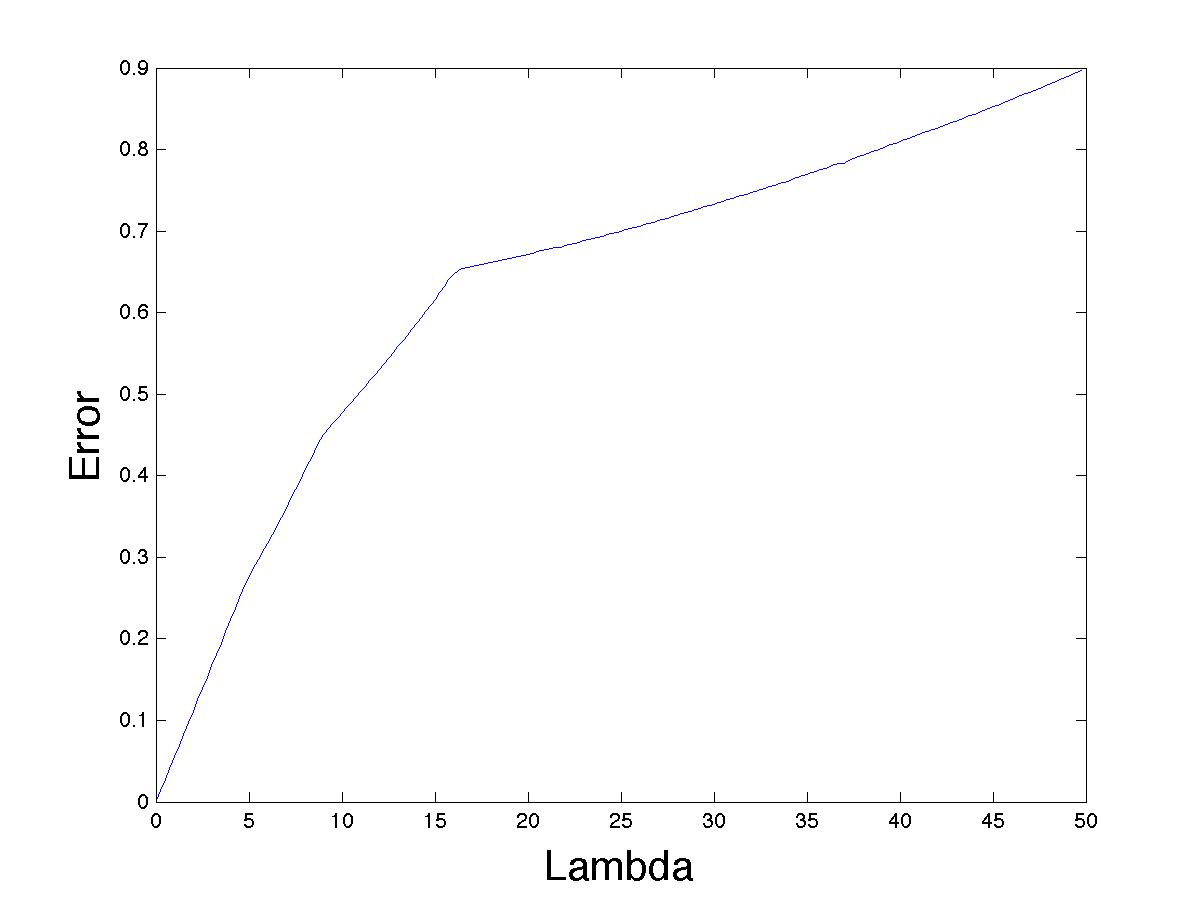
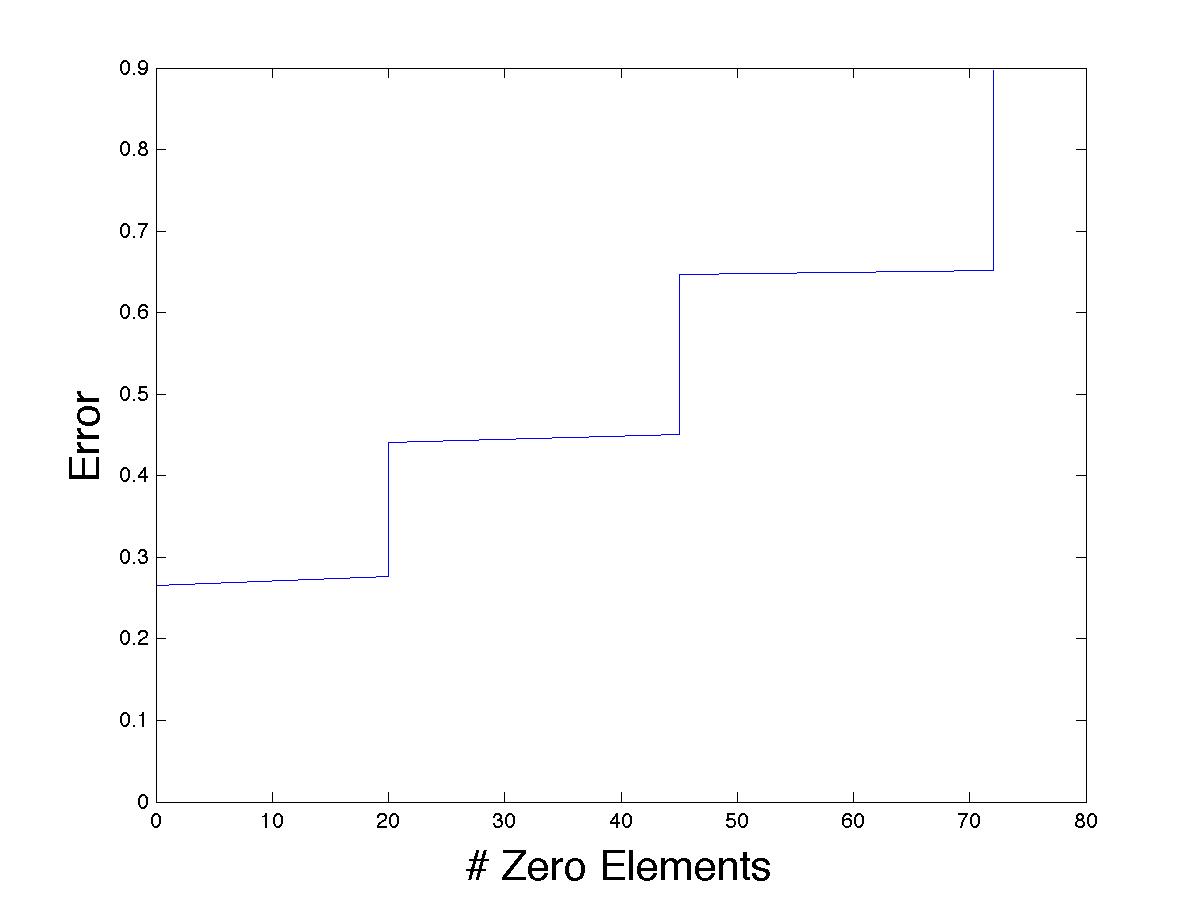
Similar to our implementation with the individual-feature Lasso regression model, we are interested in the behavior of the tracking error as the number of zeros increases – that is, as the value of lambda increases.



It is very worth mentioning that similar to the individual version of this model – as the value of lambda increases, the weights tend to grow towards the group with the highest correlation to our target data as shown in figure [FIGURE] which was drawn from our data. As the value of lambda decreases, our results become more similar to the ones from our Abs and Squares models.

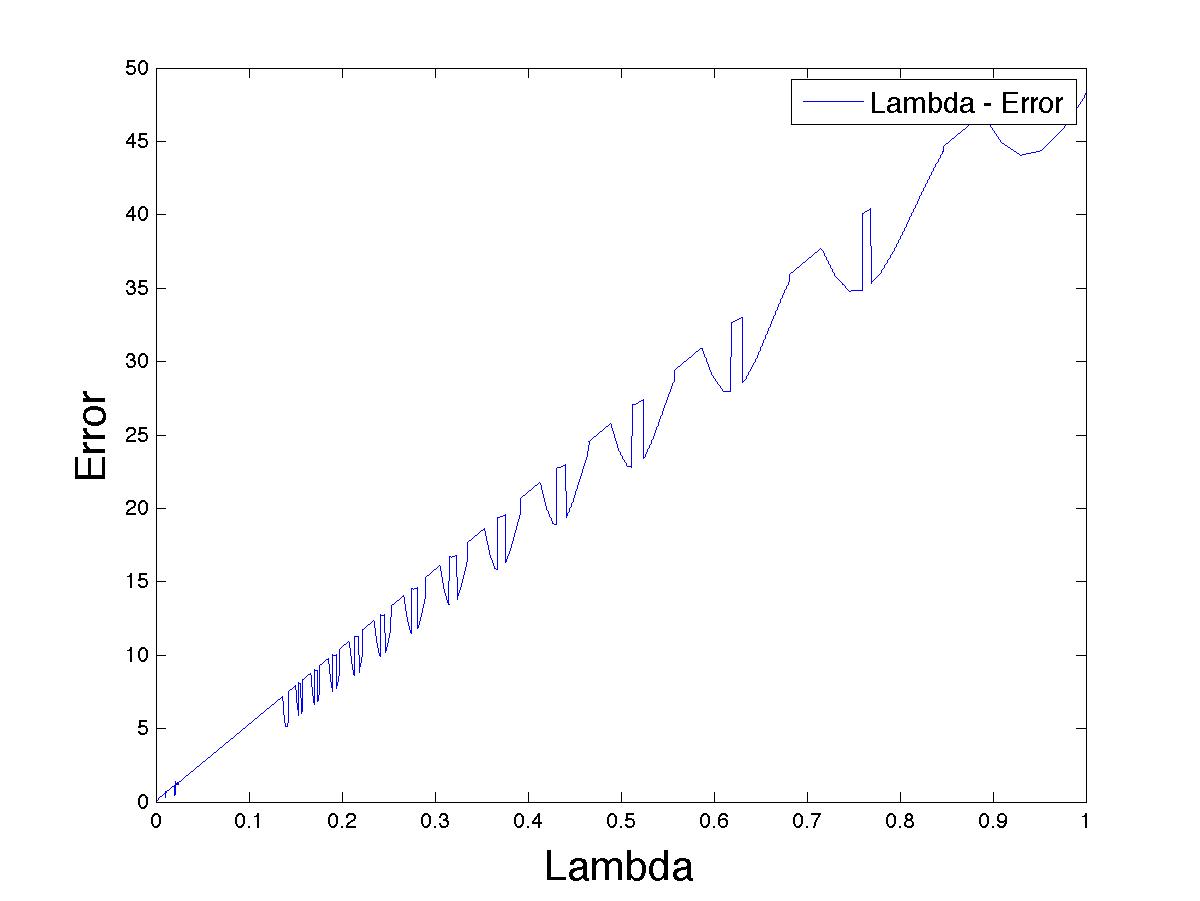
The initial results, shown below in figures [FIGURE] to [FIGURE], include the behavior of the Tracking Error against our Lambda and Number of Zeros. For both, Stochastic and Market data.

|  |  |  |
| --- | --- | --- |
| **Lambda** | **Zeros** | **Error** |
| 0 | 0 | 0.026787871 |
| 0.05 | 47 | 2.646495744 |
| 0.1 | 47 | 5.267187419 |
| 0.15 | 47 | 7.895967048 |
| 0.2 | 47 | 10.53292391 |
| 0.25 | 72 | 10.96675929 |
| 0.3 | 47 | 15.80435922 |
| 0.35 | 47 | 18.44012701 |
| 0.4 | 47 | 21.07673251 |
| 0.45 | 72 | 20.79837242 |
| 0.5 | 72 | 23.67492887 |
| 0.55 | 72 | 27.46705371 |
| 0.6 | 72 | 28.88447056 |
| 0.65 | 72 | 30.85360365 |
| 0.7 | 47 | 36.89241989 |
| 0.75 | 72 | 34.81851777 |
| 0.8 | 72 | 38.38139935 |
| 0.85 | 47 | 44.8010006 |
| 0.9 | 72 | 45.77971577 |
| 0.95 | 72 | 44.32525003 |
| 1 | 72 | 48.23389835 |



A big inconvenience with this model is that the L2-norm constraint imposed in the groups force all the weights to be the virtually same within each group – and when one of the group tends to zero, all it’s values will become zero immediately as well. This disadvantage can be overcome with the sparse group model in the next section.

Our second approach was tested with the black box approach provided by the SPAMS library. The results achieved had a much lower accuracy, but nevertheless, they showed the same behavior.



The irregular pattern in figure [FIGURE] might seem confusing on first sight, but the behavior present is very similar to the one in the previous implementation. This can be observed in the table of summarized data. The algorithm that the SPAMS library provides causes more violent changes in group data, which cause the big jumps in the data line. It can also be noticed that there are drastic changes in the number of zeros and error.

### Sparse Group Lasso

The problem with the normal group lasso model is that the L2 constraint induces sparsity only between groups, but not within groups. What this means is that if one element in a group becomes zero, all the elements in that group would become zero. The Sparse Group Lasso algorithm comes in handy, as it introduces again the L1-norm constraint found in the single-feature Lasso model which induces sparsity within the groups.

It should be noted that and needed to be tweaked a lot in order to obtain the best results. The behaviour of the model varies a lot depending on the value of these variables. This is why a good understanding should be acquired on these values, as the optimum values found were very specific, but these will also provided a great insight on the potential of this model when implemented in financial datasets.

To begin, we observe the effect of variable values of the (group sparsity) coefficient over a fixed (feature sparsity) coefficient. The initial lambda-2 coefficient is set to a value large enough to show how the Simple-Lasso sparsity is affected as the group lasso coefficient increases. Our implementation was executed initially with a fixed (of value 24), with a variable starting at 0. It should be obvious that the expected behaviour of the first iteration should be exactly the same to one from a simple Lasso regression model, and we’ll observe changes as the value of increases.

increasing towards fixed

|  |
| --- |
|  |
| Tracking Error | Number of Zeros per group | Total Number of Zero |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_lambda_error.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_group_zeros.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_lambda_zeros.jpg |

On the first iteration, the model provides the same results as a Lasso regression model, as a value of as it does not take groups into consideration. However, as the value of increases, namely the coefficient that controls the group sparsity characteristic, we can observe that the feature level sparsity decreases as the increase as increases, as it pushes values towards a balance within groups. Given that the value of does not increase, a proportionate amount of sparsity on feature level is not induced, and the weights end up without any sparsity at all, hence pushing all values towards a balanced weight of . This observation is shown more explicitly in figures [FIGURE] and [FIGURE].

increasing towards fixed

|  |  |
| --- | --- |
| Weights of Individual Features | Sum of Weights of all Features |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_lambda_features.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_lambda_group_sums.jpg |

In both figures, it can be observed that, as with a normal Lasso model (i.e. when = 0), many elements start with a zero value, and only values with higher correlation to our target data have a value larger than zero. However, as increases towards , all these values are pulled up from zero, and higher values are pulled down towards group balance. Figure [Figure] on the right gives more insight on this behaviour. Initially, the sum of weights of groups are not controlled, however, as increases, it is possible to observe that the individual weights will tend towards 1/n, and hence, the sums of the weights in the groups tend towards a It can be noticed that if there isn’t a proportionate increase of both ’s – as it will be shown below, this is the feature that makes this model so powerful.

The full power of this model comes when both values grow in proportionate linear amounts to each other. To begin our discussion on the huge power this model obtains when and grow proportionately, it is important to know it’s behaviour as they approach infinity. Intuitively, a proportionate increase of these values would be comprised of an increase of feature-level sparsity induced by , together with a proportionate increase of group sparsity being induced by As both values approach to infinity, we can observe that values are drawn towards as follows:

|  |  |
| --- | --- |
| Weights of Individual Features as and Increase | Sum of Weights of all Features in each group as and Increase |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:ftse_lambdas_features.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:ftse_lambdas_sumgroups.jpg |

Figures [FIGURE] to the left shows how the weights of the individual features tend towards []. [FIGURE] on the right shows the same behaviour for the sums of weight. It should be intuitive that as the individual feature values tend towards [], then the values of the groups will tend towards .

With this insights on the behaviour of the model we can approach our portfolio optimization problem more effectively, where in this case of portfolio index tracking, our objective is like to select the stocks that are most correlated to the Market Index – both in group and feature level – this of course, while keeping as much sparsity as possible. From our previous definitions, we can intuitively see that this can be achieved with a with a proportionate linearly increasling ’s to induce feature sparsity, while inducing sparsity as a group level in order to spread the risk in different groups.

This section contains one of the most important findings in this paper, and consists of three different experiments where we observe the effects of the values for = [0.01,20] with = /Ω and Ω = { 500, 1000, 5,000, 10,000 }.

In order to introduce the great potential applications of this model, we would like to initially recall both our previous models, the Lasso model in figure [FIGURE] on the left and the Group Lasso model in figure [FIGURE] on the right. What is great about this model is that it allows us to reach both of these models without tweaking anything but the Lambdas. Figures [FIGURE] and [FIGURE] below show the effect of their respective variable Lambda in the weights of each individual feature.

|  |  |
| --- | --- |
| Lasso  (=0) | Group Lasso  (=0) |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:LassoComplete:even newer:FeatureWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_lasso:new:FeatureWeights.jpg |

It is possible to notice in figure [FIGURE] that the Lasso model acts on each feature individually without taking into consideration groupings at all. Similarly, our Group Lasso model in figure [FIGURE] does not allow for any sparsity within groups whatsoever. With this in mind, we can now introduce the behaviour of the Sparse Group Lasso model in the figures below.

|  |  |
| --- | --- |
| Ω =10,000) | Ω =5,000 |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:5000:FeatureWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:10000:FeatureWeights.jpg |
| Ω =1,000 | Ω =500 |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:1000:FeatureWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:500:FeatureWeights.jpg |

Figures [FIGURE] to [FIGURE] are pretty much self explanatory – not only they show the full power of the Sparse Group Lasso model, but they also show the effect of the Ω in the way sparsity is achieved, both, within and between groups. Large values of Ω would provide behaviour almost exactly like the one of the simple Lasso model, and as they become smaller, these will impose a behaviour similar to the Group Lasso model.

What is even more interesting is the behaviour of the group weights as Ω increases or decreases. Figures [FIGURE] to [FIGURE] provide a very concise perspective on what effects this variable has on groups, and it shows the true potential in this model.

|  |  |
| --- | --- |
| Ω =10,000 | Ω =5,000 |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:10000:GroupWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:5000:GroupWeights.jpg |
| Ω =1,000 | Ω =500 |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:1000:GroupWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:Relatives:500:GroupWeights.jpg |

At a glance, we can observe the presence of group-level sparsity almost immediately as Ω decreases – whole sets of groups are drawn to zero almost immediately when Ω is smaller. Likewise, the higher our value of Ω, the less our model takes groups into consideration. However, going more in depth, if we observe Figure [FIGURE Ω=5,000] and figure [FIGURE Ω=5,000], for lower values of (i.e. ) we can observe that groups with lower correlation to the target data decrease much less than with smaller values of Ω, but that is because only the ‘less important’ features within the group that decrease towards zero. This is a very important characteristic when it comes to finance, as a financial trader will require diversifying his portfolio – this very interesting application will be covered more in depth in the Analysis section.

Another very important point we should consider in regards to this regression model is the effect of Ω in the tracking error. Figure [FIGURE] contains a table with the tracking error values for each of the values of Ω.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ω = 500 | | Ω = 1,000 | | Ω = 5,000 | | Ω = 10,000 | |
| **# Zeros** | **Group Lasso** | **# Zeros** | **Error** | **# Zeros** | **Error** | **# Zeros** | **Error** |
| 0 | 0.0000 | 0 | 0.0000 | 0 | 0.0000 | 0 | 0.0000 |
| 0 | 0.1614 | 0 | 0.1880 | 31 | 0.2822 | 38 | 0.3119 |
| 0 | 0.2919 | 6 | 0.3153 | 47 | 0.4132 | 55 | 0.4483 |
| 20 | 0.3966 | 24 | 0.4225 | 54 | 0.5098 | 62 | 0.5507 |
| 45 | 0.4870 | 25 | 0.5167 | 58 | 0.5969 | 68 | 0.6414 |
| 45 | 0.5311 | 49 | 0.5782 | 64 | 0.6728 | 73 | 0.7127 |
| 45 | 0.5733 | 50 | 0.6171 | 66 | 0.7405 | 74 | 0.7808 |
| 45 | 0.6176 | 51 | 0.6515 | 74 | 0.8004 | 76 | 0.8447 |
| 45 | 0.6645 | 52 | 0.6844 | 75 | 0.8465 | 77 | 0.9065 |
| 45 | 0.7115 | 52 | 0.7166 | 75 | 0.8920 | 77 | 0.9686 |
| 72 | 0.7469 | 52 | 0.7486 | 77 | 0.9342 | 81 | 1.0270 |
| 72 | 0.7687 | 52 | 0.7802 | 77 | 0.9745 | 83 | 1.0673 |
| 72 | 0.7911 | 52 | 0.8114 | 78 | 1.0155 | 85 | 1.1019 |
| 72 | 0.8137 | 52 | 0.8431 | 80 | 1.0555 | 86 | 1.1355 |
| 72 | 0.8365 | 53 | 0.8777 | 80 | 1.0948 | 87 | 1.1660 |
| 72 | 0.8605 | 53 | 0.9127 | 84 | 1.1323 | 88 | 1.1954 |
| 72 | 0.8849 | 53 | 0.9483 | 84 | 1.1630 | 88 | 1.2260 |
| 72 | 0.9098 | 54 | 0.9843 | 84 | 1.1938 | 88 | 1.2583 |
| 72 | 0.9352 | 54 | 1.0210 | 85 | 1.2236 | 88 | 1.2928 |
| 72 | 0.9614 | 66 | 1.0576 | 85 | 1.2514 | 88 | 1.3288 |

The data also reflects the group and individual feature behaviour induced by both our variables. It can be observed that when the value of Ω is lower, the number of zeros tend to stay constant for larger number of iterations. Likewise, as the value of Ω increases, sparsity increases faster in the feature level.

# Analysis

Very detailed results have been obtained in the previous section, however it is not until now that these come together to provide a great insight on their application to the financial markets, which will allow us to see how efficient these proposed models really are.

Once these models have been thoroughly compared, an application of these will be presented with the objective of showing the potential uses in the financial markets to provide traders and investors with new efficient tools to perform their day-to-day tasks.

Before starting this section it also important to mention that the aim of this paper is to find efficient ways to index tracking, as the results will show, there will also be room for suggestions to apply these modes in portfolio selection, as characteristics of these models include diversification, which is essential for an efficient financial portfolio.

## Model Comparison

Throughout the implementation it was observed that each of the regression models used had a characteristic in which they excelled – speed, accuracy, sparsity, efficiency, etc. In this section we review and analyze the results obtained and the potential applications of these results.

### Accuracy

When it comes to portfolio optimization, and in specific index tracking, accuracy is a very important measurement, as we want to be able to follow the behavior of the Market Index when having the lowest amount of stocks possible in order to reduce transaction costs.

Throughout this paper, it was evident that the most accurate regression model in regards to minimizing tracking error was without any doubts the Abs model. This was followed by the Square, Lasso, CVaR and finally Ridge.

The tracking errors have been compiled into a table, which can be found below, in figure [FIGURE]. This table contains the values for the tracking error relevant to the number of zeros for each model.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lasso** |  | **Group Selection** | | **Group Lasso** | | **Sparse Group Lasso** | | **L1=L2/500** | | | **L1=L2/1000** | | **L1=L2/5000** | | **L1=L2/10000** |
| **# Zeros** | **Error** | **# Zeros** | **Error** | **# Zeros** | **Group Lasso** | | **# Zeros** | | **Group Lasso** | **# Zeros** | **Error** | **# Zeros** | **Error** | **# Zeros** | **Error** |
| 0 | 0.0000 | 0 | 0.0000 | 0 | 0.0000 | | 0 | | 0.0000 | 0 | 0.0000 | 0 | 0.0000 | 0 | 0.0000 |
| 3 | 0.0659 | 4 | 0.0355 | 0 | 0.1391 | | 0 | | 0.1614 | 0 | 0.1880 | 31 | 0.2822 | 38 | 0.3119 |
| 11 | 0.1154 | 7 | 0.0607 | 20 | 0.2775 | | 0 | | 0.2919 | 6 | 0.3153 | 47 | 0.4132 | 55 | 0.4483 |
| 20 | 0.1538 | 9 | 0.0707 | 20 | 0.3829 | | 20 | | 0.3966 | 24 | 0.4225 | 54 | 0.5098 | 62 | 0.5507 |
| 26 | 0.1878 | 12 | 0.0989 | 45 | 0.4766 | | 45 | | 0.4870 | 25 | 0.5167 | 58 | 0.5969 | 68 | 0.6414 |
| 31 | 0.2201 | 14 | 0.0857 | 45 | 0.5437 | | 45 | | 0.5311 | 49 | 0.5782 | 64 | 0.6728 | 73 | 0.7127 |
| 35 | 0.2501 | 15 | 0.0758 | 45 | 0.6172 | | 45 | | 0.5733 | 50 | 0.6171 | 66 | 0.7405 | 74 | 0.7808 |
| 41 | 0.2771 | 16 | 0.1058 | 72 | 0.6592 | | 45 | | 0.6176 | 51 | 0.6515 | 74 | 0.8004 | 76 | 0.8447 |
| 44 | 0.3008 | 17 | 0.0829 | 72 | 0.6718 | | 45 | | 0.6645 | 52 | 0.6844 | 75 | 0.8465 | 77 | 0.9065 |
| 46 | 0.3244 | 18 | 0.1176 | 72 | 0.6851 | | 45 | | 0.7115 | 52 | 0.7166 | 75 | 0.8920 | 77 | 0.9686 |
| 49 | 0.3455 | 20 | 0.1071 | 72 | 0.6998 | | 72 | | 0.7469 | 52 | 0.7486 | 77 | 0.9342 | 81 | 1.0270 |
| 54 | 0.3641 | 20 | 0.1224 | 72 | 0.7159 | | 72 | | 0.7687 | 52 | 0.7802 | 77 | 0.9745 | 83 | 1.0673 |
| 55 | 0.3798 | 22 | 0.1121 | 72 | 0.7331 | | 72 | | 0.7911 | 52 | 0.8114 | 78 | 1.0155 | 85 | 1.1019 |
| 56 | 0.3955 | 24 | 0.1183 | 72 | 0.7502 | | 72 | | 0.8137 | 52 | 0.8431 | 80 | 1.0555 | 86 | 1.1355 |
| 57 | 0.4116 | 25 | 0.1344 | 72 | 0.7695 | | 72 | | 0.8365 | 53 | 0.8777 | 80 | 1.0948 | 87 | 1.1660 |
| 59 | 0.4276 | 26 | 0.1235 | 72 | 0.7891 | | 72 | | 0.8605 | 53 | 0.9127 | 84 | 1.1323 | 88 | 1.1954 |
| 59 | 0.4440 | 27 | 0.1381 | 72 | 0.8097 | | 72 | | 0.8849 | 53 | 0.9483 | 84 | 1.1630 | 88 | 1.2260 |
| 60 | 0.4609 | 28 | 0.1318 | 72 | 0.8306 | | 72 | | 0.9098 | 54 | 0.9843 | 84 | 1.1938 | 88 | 1.2583 |
| 62 | 0.4765 | 29 | 0.1426 | 72 | 0.8522 | | 72 | | 0.9352 | 54 | 1.0210 | 85 | 1.2236 | 88 | 1.2928 |
| 63 | 0.4917 | 31 | 0.1452 | 72 | 0.8753 | | 72 | | 0.9614 | 66 | 1.0576 | 85 | 1.2514 | 88 | 1.3288 |

### Cardinality Constraint

One of the main concerns in this paper was to provide the most efficient way to provide sparsity to an index tracking portfolio. This is, to be able to follow a Market Index as close as possible with the smallest amount of stocks possible.

Sparsity was thoroughly observed and compared between our models during the implementation, so there is not much need for focus on this area. Instead, a very interesting point to analyze is the effect and behavior of non-sparse models in group-aware environments.

By this, we mean to observe and analyze what behavior models like our feature and group selection models have when analyzed from the same perspective as with sparse models like the group Lasso. Figures [FIGURE] to [FIGURE] provide this perspective by breaking down the results of the feature selection, group selection, and individual lasso regression models into the groups that the group lasso and sparse group lasso were divided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Abs | Lasso | Group Lasso | Simple Lasso | Total Number of Zero Elements as Lambda 1 increases |
| Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:ABS_C0_GROUPSUMS.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:LassoComplete:even newer:GroupWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:group_lasso:ftse100_group_lasso_lambda_weight_5.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:LassoComplete:even newer:GroupWeights.jpg | Macintosh HD:Users:axsauze:IdeaProjects:matlab:SPAMS:Graphs:sparse_group_lasso:fixed_l2:ftse_fixl2_lambda_zeros.jpg |

By observing this graphs, we can see that as expected, the Abs model does not take into account groups at all – if anything can be noticed is that the model keeps the groups weights to a value proportionate to their size. This can be a good thing as it provides diversity by distributing risk evenly in different assets, however, this does not help highlight any higher correlations between stocks, as most of the weights are kept to a relatively equal level.

The Lasso model in the other hand follows a pattern similar to the group lasso, where, although there is no knowledge by the model about the groupings, the weights are actually arranged accordingly to give a higher weighting to groups with higher correlations.

### Speed

Speed in finance is always one of the main concerns – financial institutions are constantly striving for improvement in regards to speed. When it comes to financial markets, very small fractions of time really matter – whether it’s financial automated systems, or software for traders, it is always a main concern to provide as much speed as possible.

For this reason it is crucial that we analyze the execution times for each of the models and approaches in this paper.

## Applications

# Expansion

After observing the great potential the group lasso formulation had on financial data, and the great positive effects group data had on our regression models, a new model will now be proposed for the expansion of this project.

Before proposing the new model it is very important to emphasize that groupings have proven to be a great source of insight in financial data, as they not only show how correlated groups interact with the data, but also allows us to use groupings as an extra dimension of data that allows for more effective regression. Specific to finance, it’s very important to recall that these models provide great financial characteristics – portfolio diversification is a main concern, and it can be seen that in models like Abs it is only possible to obtain these results through a greedy selection process, which still does not account for between or within group sparsity as it was observed from figure [FIGURE SHOWING GROUP WEIGHTS OF ABS].

## Multiple Sparse Group Regression (Proposed Model)

In this section we would like to briefly propose a model. The current sparse group model, as it might be observed, can consider groupings in the regression, however it is limited to only one category/type of groupings, so it would not allow us to consider multiple categories/types of groups. This section is very brief as the model to be proposed is just an encouragement for further research in this area of Machine Learning, as the knowledge and time in hands at the moment did not allow us to prove the correctness of this formula. Briefly this model aims to expand the concept of sparse group regression into more than one single category of groupings, for data that might consist of multiple correlated groups.

To provide context, we would like to give an example of a situation where we would encounter data with multiple categories of groupings in finance. A perfect example would be a typical portfolio consisting of commodities, options currencies, stocks, etc. The financial instruments comprising this portfolio can be grouped not only by their nature (i.e. stocks, currencies, bonds, etc), but also by industry, sector, or based on historical return or volatility.

## Notation

In order to introduce this formula, we first need to introduce few concepts. We would like to recall the constraint proposed, .

A problem with introducing multiple categories of groups is that our constraint of would be violated, as there would be overlaps between groups. An example of this would be groups of stocks by type of instrument (stocks, currencies, etc), which would overlap if these are grouped further into sectors or industry.

A simple solution for this would be to subdivide overlapping groups, which would keep the constraint imposed earlier, and would just give us a larger number of groups These stocks would all be considered as completely different groups, but to avoid this, a vector coefficient **w** can be introduced which would scale features from each group according to their groups. There should however exist a more elegant approach to this problem that could allow us to consider whole groups without having to break them into subsets of groups. This vector **w** will however be actually used in order to provide a scale to each category as required.

This model is based on the new notation that was presented when the group lasso model was introduced - Namely the formulation proposed introduced a constraint that allowed us to compute the formulation with only matrix multiplications – together with the model proposed by [REFERENCE YUE LIN] defined as follows:

To begin with, we would like to briefly introduce very briefly a new notation. We will have K categories containing groups of size **,** our indexing notation will be expanded to a matrix consisting of rows containing the logical indexing for all groups in all categories, this can be defined as:

Our new set denotes the indexes for the stocks contained in group g of category k. With this, we would be able to introduce our variable where would be the subset containing the stocks for group g in category k. Now, although our constraint will be removed, we can see form our previous observations that . What this tells us is that the weightings in our original variable to be learned, , will never suffer any change in terms of logic structure even when groups are introduced. Before introducing the model we would like to add a placeholder to our second coefficient as . This would really simplify our formulation, and will allow us to revisit this coefficient further on, which is the most important one. This allows us to define and expand the following model:

However, from what we mentioned earlier, the weights for each stock will be the same on any state of the execution. What this means is that if the stocks from group x in category I are the same than the stocks from group y in category j. What this proves is that will always hold. Likewise, as the sum of absolute values should not vary on any category, as all categories will be composed of the same stocks. We can also omit the K value as we can assume will be able to scale as required. This would leave us with:

Similar to our absolute sum of values, the first coefficient would always be the same, we could replace the summation simply with .

This now leaves us with the main and only coefficient that induces the Group sparsity effect desired – namely:

This is basically the scaled sum of the norms of each of our groups, which in our

situation would expand to:

Now can see that that our problem has reduced to a simple sum of L2-norm of all our individual groups for each category. In order to adapt this to our previous formulation, we would need to build a vector , and a function norms(**v**,) where it returns the L2-norm for a matrix of a set of indexes.

## MSGR Model definition

This would allow us to define our model as:

Assuming that we could also superscript logical indexes, this formulation can be defined as our previously introduced formulation:

We would also like to recall the **w** vector mentioned previously that would provide a scaling to each specific category as required – this would allow for specific categories to have a higher effect than others:

This new formulation now takes into account overlapping groups which are scaled by our constant **w** vector in order to take group regression characteristics to the next level.

It should be noted that if only one category of the weights of **w** is set to 1 and the rest to zero, this would be the same effect than the normal Sparse Group Lasso model.

Once again, it is necessary to mention that for as of the time of writing, time did not allow for a mathematical proof on the correctness or convexity of this model. This model was only proposed from what was learned from the sparse models, and from the observations and results obtained in the implementation.

# Conclusion

The application of this models provided an insight on the effects of sparse and group regression approaches in financial data. Several models were analysed and implemented in a feature and group model selection approach inspired by [REFERENCE MAHESAN] – These regression models included Abs, Least Squares, NCCVaR, CVaR and Ridge Regression. Although these models provided different approaches to optimal stock subset selection, they all proven to have a similar behaviour, and only varied between few stocks, holding at least 50% of the same stocks between each iteration. This reassures us that when taking a feature selection approach to stock subset selection, most models will prove to be very accurate. In regards to group selection, although a relatively low tracking error was achieved, the sparsity of these models was very limited, as it could only either include or exclude groups completely.

Great insights were obtained in regards to sparse and sparse group models, as well as in group approaches to financial data. Initially it is worth noting that a a huge point is that stocks within groups in financial datasets contain a great correlation. Industries for example, any effects in the group as a whole would show a variation in a big percentage of its constituent stocks. This is what brought the biggest attention when implementing sparse group models, as it was observed that these models have great impact when implemented with correlated data, as they induces what in finance would be known as diversification – and this diversification can be tweaked accordingly. The observations made were that accuracy of these models were definitely not as high as the results obtained through our feature selection model, however, it was in regards to diversification where these models excelled, as they provided very in depth customization when it comes to how sparsity is induced when the regression model is applied.

In the financial world, sparse regression models, both in feature and group level show great potential, as they can provide huge amounts of information regarding data and its correlations between its individual features and their respective groups. This also provides great insight on the paths that can be taken after obtaining this information.

Speed was also great concern when analysing these models, and this is another characteristic in which sparse models excelled. The execution time of sparse models, which was not more than 30 seconds long, was only a tiny fraction compared to the time taken for the feature and even group selection approaches, which with datasets of only 200 stocks, took mode than 3 days to compute.

Finally, a new model was proposed based on the observations in both the implementation and analysis of our approaches. Although time did not allow for proving the correctness and convexity of this approach, what is proposed is an insight for potential situations where multiple groupings are to be considered – in financial markets, this would arise when a trader wishes to induce sparsity in a portfolio containing financial instruments of different natures, such as stocks, bonds, currencies, etc.

# References

|  |  |
| --- | --- |
| [0] | Takeda, A., Niranjan, M., Gotoh, J. Y., & Kawahara, Y. (2013). Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. *Computational Management Science*, *10*(1), 21-49. |
| [1] | Markowitz, H. (1952). Portfolio selection\*. *The journal of finance*, *7*(1), 77-91. |
| [2] | Philippe J. (1996) Risk2: Measuring the Risk in Value at Risk. Financial Analysts Journal , Vol. 52, No. 6 (Nov. - Dec., 1996), pp. 47-56 |
| [3] | Rockafellar, R. T. (1997). *Convex analysis* (Vol. 28). Princeton university press. |
| [4] | Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of risk*, *2*, 21-42. |
| [5] | Takeda, A., Gotoh, J. Y., & Sugiama, M. (2010, August). Support vector regression as conditional value-at-risk minimization with application to financial time-series analysis. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on* (pp. 118-123). IEEE. |
| [6] | sss, J., & Xiao, J. Y. (2001). Return to RiskMetrics: the evolution of a standard. *RiskMetrics Group*. |
| [7] | Brodie, J., Daubechies, I., De Mol, C., Giannone, D., & Loris, I. (2009). Sparse and stable Markowitz portfolios. *Proceedings of the National Academy of Sciences*, *106*(30), 12267-12272. |

1. Simple Regression Models: Sum of Absolute Values, Sum of Squares, Ridge Regression, Conditional-Value-at-Risk and Lasso. [↑](#footnote-ref-1)
2. Group Lasso and Sparse Group Lasso [↑](#footnote-ref-2)