# Improvements to Factor Models with Machine Learning Methods

Can we make factor models smarter?
Optimization Modelling in Finance Course Project

Published by

Qin Lu (ql224)
Yudong Yang(yy723)
Sizhang Zhao(sz459)

*The Cornell University*

May 14, 2017

**Abstract**

Factor models try to describe the return-generating process so that we can get more accurate estimators for mean vector and covariance matrix of underlying assets, which is a good news for Markowitz mean and variance method. And OLS regression is used for original factor models. Although OLS has really good properties, there are other regression methods that may be more suitable for this setting. Thus, in our paper, we change the regression methods to other machine learning regression techniques and test whether the performance of factor models can be improved. And we find that huber regression can consistently beat the traditional OLS regresssion and achieve higher cumulative return and sharpe ratio with lower maximum drawdown for Markowitz back testing.

**Keywords**: Factor models, Machine learning, Markowitz mean and variance method, regularizer, huber regression, support vector regression, neural network

# Content

# Part I

# Introduction

The mean-variance method was firstly brought up by Markowitz[1] in 1952 and it is suitable for one period asset allocation. To implement mean-variance method, we need to get estimators for the mean vector and the convariance matrix for the underlying assets. The original way is to estimate them by the historical data. There are two main drawbacks for this method. Firstly, it would take large computational efforts if we have a large pool of assets. The other one is that a set of historical data provides only a sample of possible outcomes and that is to say, the numbers obtained from historic data are "subject to error". The problem becomes more severe when it comes to optimazation settings, we want to find the optimal values which is more prone to accumulate the errors from the data.

Factor models provide a solution to mitigate the problem. A factor model describes the return-generating process. Once we can get the good factors which can explain the returns pretty well, it is easier for us to get a better estimates for mean vector and covariance matrix for our underlyings. One famous factor model is Fama-French three factor models[2]. They extended their three-factor model to five-factor model in 2016[3].

Although the number of the factors can be much less than the number of the assets (good news for computation), they probabily have lots of overlaps in terms of variances (there are still place where we can do better). PCA (Principal Component Analysis) provides a wonderful solution. PCA was firstly brought up by pearson in 1901 [4] and it is a good way to lower the dimention of the datasets, which has been widely used for the current factor models.

Nowadays, machine learning methods become more and more popular, which provides a wonderful method for learning the relationships between different variables. PCA itself is one of the unsupervised learning methods (there are only independent variables but no denpendent variable under this setting).

For the original factor models, the OLS regression is used to get the relationship between the factors and the return data. There are more other supervised learning methods (there are both independent variables and dependent variable under this setting) coming out from machine learning field. Thus, in this paper, we are interested into applying other regression methods (from machine learning area) to the factor models and test whether the performance of the factor models can be improved by these techniques.

One of the ways to revise the OLS regression is to add regularizers. The general form of a regularized problem is shown below,

$$\text{minimize}(\sum_{i=1}^{n} l(x_i, y_i, w) + r(w))$$

where $l$ is the loss function, $l_2$ norm for OLS setting and $r(w)$ is the regularizer. Regularizers can avoid overfitting problems and may have some other features but would add to bias. Two popular regularized OLS regression is Lasso Regression and Ridge Regression, and thus we try to replace the OLS regression by Lasso and Ridge to see how factor models perform.

The other way to revise the OLS regression is to change the loss function. As mentioned earlier, $l_2$ norm is used as the loss fucntion for OLS. There are other loss functions can be applied. Such as huber loss for huber regression. SVR (support vector regression) and neural network has its own optimazation objective for finding the relationship between the the independent variables and dependent variable. In our paper, we use different regression methods also to try to improve the performance of the factor model.

# Part II

# Theories and Strategy

## 1 Theories

### 1.1 Markowitz Mean and Variance Method

We consider a market system consisting of $d+1$ assets, named $S_t^1, \cdots, S_t^d, B_t$, where $B_t$ is the risk-free treasury. A portfolio consists of a collection of these assets of certain units. We denote the portfolio's weight by a vector

$$\omega = (\omega_1, \omega_2, \cdots, \omega_{d+1})^T, \qquad \sum_{i=1}^{d+1} \omega_i = 1.$$

If the total wealth of the portfolio at time $t$ is $W_t$, then the amount of investment on asset $i$ is $\omega_i W_t$.

There are two types of return. Markowitz uses arithmetic return which is defined as follows,

$$r_t = (r_t^1, \cdots, r_t^d, r_t^f)^T, \qquad r_t^i = \frac{S_t^i - S_{t-1}^i}{S_{t-1}^i},\ i = 1, \cdots, d$$

where the risk-free rate is denoted as $r_t^f$.

The second type of return is the log-return. It is defined as $log(S_t^i) - log(S_{t-1}^i)$. For our project, we mainly deal with log-return. Only when we are going to Markowitz problem, we project our daily log-return to monthly arithmetic return.

Another important input in the Markowitz model is the covariance matrix of stocks $S_t^1, \cdots, S_t^d$. Traditional way to generate the covariance matrix is calculating the historical covariance from the return data. Later, we will discuss different ways to compute covariance, as well as expected returns. The covariance estimation at time $t$ is denoted by $\Sigma_t$.

We use the Cholesky Decomposition of $\Sigma_t$, which reduces complexity in calculation. Denote $\Sigma_t = U_t^T U_t$. The Markowitz Model being used is,

$$\begin{aligned}
\text{maximize}_\omega \quad & \omega^T r_t \\
s.t. \qquad & \Sigma_{i=1}^{d+1} \omega_i = 1 \\
& \omega_i \geq 0 \qquad \text{for all } i = 1, \cdots, d+1 \\
& \omega_i \leq 0.5 \qquad \text{for all } i = 0, \cdots, d+1 \\
& \left\| U_t \omega^{1:d} \right\| \leq \sigma \qquad \text{where } \omega^{1:d} = (\omega_t^1, \cdots, \omega_t^d)^T
\end{aligned}$$

Short selling is not allowed, and the largest allocation cannot exceed half of the total wealth.

We set a 1-month rebalancing frequency, which means that Markowitz is applied every month during the back-testing period.

### 1.2 Factor Model

As mentioned earlier, factor model gives a rather reliable decomposition of market risk and return source. It consists of a linear regression model of historical returns on different factors. Denoting the factors at time $t$ as $F_t^1, \cdots, F_t^m$, the regression model for asset $i$ can be written as,

$$r_t^i = \beta_0 + \beta_1^i F_t^1 + \cdots + \beta_m^i F_t^m + \epsilon_t^i, \qquad t = 1, \cdots, T$$

The factors should be chosen carefully to cover as much market information as possible. A description on selected factors for this report can be referred to Section III.1.

The $\beta_i$-s are traditionally determined by OLS estimator.

### 1.3 PCA

While factors are chosen to depict as much market information as possible, it is unavoidable to discover that the factors are correlated among each other. Principal Component Analysis (PCA), is a dimension reduction technique, which linearly combines factors into new factors that are pairwise orthogonal, and meanwhile packing a fair amount of information into the first several new factors.

To start with, denote the factor matrix as,

$$F = \begin{bmatrix} F^1_{t=1} & F^2_{t=1} & \cdots & F^m_{t=1} \\ F^1_{t=2} & F^2_{t=2} & \cdots & F^m_{t=2} \\ \cdots & \cdots & \cdots & \cdots \\ F^1_{t=T} & F^2_{t=T} & \cdots & F^m_{t=T} \end{bmatrix}$$

The historical information of factor data can be expressed as a sample covariance matrix,

$$\hat{\Sigma} = \frac{1}{T-1} \sum_{t=1}^{T} \left(F_t - \bar{F}\right) \left(F_t - \bar{F}\right)^T$$

where $F_t = (F^1_t, F^2_t, \cdots, F^m_t)$, and $\bar{F} = \left(\frac{1}{T}\sum_t F^1_t, \frac{1}{T}\sum_t F^2_t, \cdots, \frac{1}{T}\sum_t F^m_t\right)$.

Denote the eigenvalues of $\hat{\Sigma}$ are $\lambda_1 > \lambda_2 > \cdots > \lambda_m$, with no ties. Denote $o_1, o_2, \cdots, o_m$ are their respective eigenvectors, and $O = (o_1, o_2, \cdots, o_m)$. $O$ is a normalized orthogonal matrix.

By linear algebra,

$$\hat{\Sigma} = O\text{diag}(\lambda_1, \cdots, \lambda_m)O^T.$$

Taking the orthogonal eigenvectors with the biggest eigenvalue to the smallest as the new factors, we also achieved our goal in maximizing information (variance) contained in the first few factors. For instance, $o_1$ is exactly the vector who maximizes the variance of a linear combination of $F^1, \cdots, F^m$.

Meanwhile, the proportion of total variance explained by the $i$-th eigenvector is exactly

$$\frac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_m}$$

The prove can be found in [4].

PCA works fairly well with factor data. For the chosen factors, the variance explained by the first few eigenvectors quickly accumulates to 95%.

Table 1: PCA Analysis on existing factors

| Number of new factors | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Proportion of variance explained | 88.68% | 94.84% | 99.68% | 99.80% | 99.89% |

The same proportion may be lower during back-test, since the historical periods are shorter (only a month) and many short-term effects may come into place. However, it is enough to choose the first four components as our new factors, as the cumulative variance of them remains above 90% throughout the testing period.

One thing to mention is that after we conduct PCA to choose the principal components, we normalize the new factors so that they are all in the same scale.

### 1.4 Lasso Regression

The goal of OLS(ordinary least squares) regression is minimizing the sum of the squares of the differences between the observed values in the giving data set and those predicted by a linear function of a set of explanatory variables. Besides the easy implementation and some great statistical properties, the model tries its best to fit the training data with the risk of overfitting and losing the generality.

For financial time series data, the size of the data set is limited, and therefore the technique of regularizing helps reduce overfitting and improve the robustness of the model. Among all techniques of regularizing, Lasso has the nice property to improve the prediction accuracy and interpretability by altering the fitting process to select only a subset of the explanatory variables to fit the final model. By adding a constraint to force the sum of the absolute value of the regression coefficients to be less than a fixed value, Lasso can efficiently chose the less statistical significant coefficients and set them to zero. The basic form is as following:

The objective of lasso is to solve:

$$\min_{w} \frac{1}{2N_{sample}} ||Xw - Y||_2^2, \qquad \text{subject to } ||w||_1 \leq t$$

which is equivalent to solve:

$$\min_w \frac{1}{2N_{sample}}||Xw - Y||_2^2 + \alpha||w||_1$$

where,
$\alpha$ is the regularization multiplier, which is given before optimizing the regression coefficients.

In order to choose the best $\alpha$, we use the method of validation. First we divide the data into training set (80%) and validation set (20%). Then we used the grid search to get different value of $\alpha$ and get different regression coefficients $w_\alpha$ in the training set, and then using validation set to compare the value of the objective function for different $w_\alpha$. We picked the alpha with the lowest value of the objective function as the final regularizer multiplier.

One possible disadvantage of regularized regression is that regularizer adds bias to the model in the training data, however, since the goal of the model is to predict the unobserved data, adding regularizer helps reduce the variance of the model and tends to generalize the hidden pattern.

### 1.5 Ridge Regression

The idea of ridge regression is very similar to Lasso regression, the only difference is the form of regularization. Other than posing constraint to the sum of the absolute value of regression coefficients (L1 norm), ridge regression poses the constraint to the sum of the squared value of regression coefficients (L2 norm).The basic form is as following:

$$\min_w \frac{1}{2N_{sample}}||Xw - Y||_2^2, \text{ subject to } ||w||_2 \leq t$$

which is equivalent to solve:

$$\min_w \frac{1}{2N_{sample}}||Xw - Y||_2^2 + \alpha||w||_2$$

where,
$\alpha$ is the regularization multiplier, which is given before optimizing the regression coefficients.

Unlike the lasso regularizer, ridge tends to shrink the parameters instead of zero them out, and therefore ridge regression generally yields out better predictability. Also, because the Lasso regression includes the the calculation of absolute value, which needs more computation than ridge regression.

One drawback of ridge regression is that all parameters are kept in the model, and it will not select the model automatically, where the techniques of dimension reduction like PCA might help.

### 1.6 Huber Regression

Although the traditional OLS regression have lots of good properties, like if the assumptions is satisfied, the estimator would be BLUE (best linear unbiased estimator), it may be highly sensitive to outliers, which leads to the violation of the assumptions. Robust regression methods are designed to be not overly affected by violations of assumptions. And huber regression is a typical robust regression method.

Huber regression take huber loss as its loss function instead of the $l_2$ norm loss function in OLS regression. The loss function is shown below,

$$\min_{w\sigma} \sum_{i=1}^{n} (\sigma + H_m(\frac{X_i w - y_i}{\sigma}\sigma)) + \alpha||w||^2$$

where,

$$H_m(z) = \begin{cases} z^2 \ if \ |z| < \epsilon \\ 2\epsilon|z| - \epsilon^2 \ otherwise \end{cases}$$

And if $\alpha$ is set to zero, then no regularizer is applied.

From the loss function, we can find if the deviation of the fitted value from the original value is not large, we would take a $l_2$ norm and if the deviation is large (dependent on $\epsilon$), then $l_1$ norm (linear loss) is applied. It can be shown in the figure 1 (green line for huber loss and blue line for $l_2$ loss). From the figure, it is more clear that we give less penalty when the deviation is large compared with $l_2$ norm and thus huber regression is less sensitive to outliers.

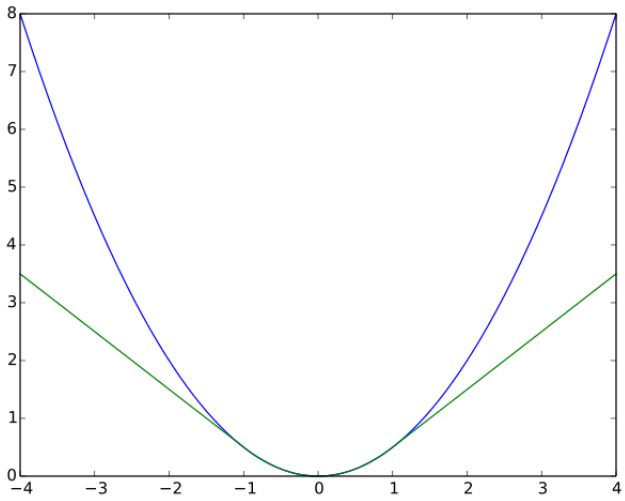



Figure 1: Plot for Huber loss

In our research, we change the OLS loss ($l_2$ norm) to huber loss and test whether it can imporve the efficiency.

### 1.7 SVR

SVM (support vactor machine) is a famous classifaction method. Its loss function is hinge loss,

$$l_{hinge}(x, y; w) = (1 - yw^T x)^+$$

And the plot is shown in figure 2.

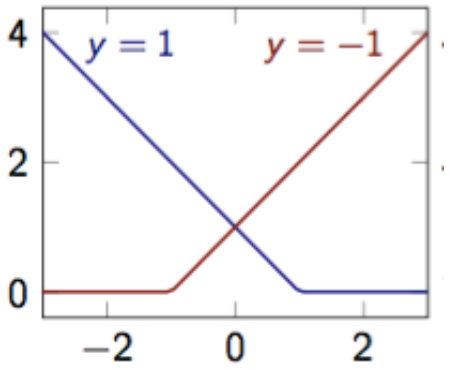



Figure 2: Plot for Hinge loss

How can we understand hinge loss? Suppose we are have the following optimazition problem.

$$\begin{aligned} &\text{minimize } ||w||^2 \\ &\text{subject to } \sum_{i=1}^{n} l_{hinge}(x_i, y_i; w) = 0 \end{aligned}$$

If we can classify each point correctly, we would have,

$$1 \leq yw^T x$$

Take norm of each side, we would have,

$$1 \leq ||w||^2 ||x||^2$$

Thus it can be viewed as there exits a margin, inside which there are no data points and the positive points and negative points are in the different sides of the margin, which can be shown in figure 3. And the width of the margin is decided by $||w||^2$, and as $||w||^2$ gets smaller, the margin gets bigger. What we want to do for SVM is basically try to enlarge the margin as much as possible unless there are points fall into the margin area.

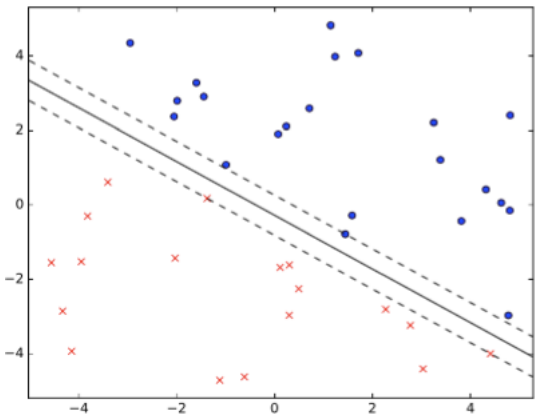



Figure 3: Interpretation of SVM

For a general setting, when data points cannot be perfectly classified, only data points inside the margin would contribute to the loss but the points outside the margin would not contribute to the loss.

The general optimization problem would be shown below but we would not get into details here.

$$\text{minimize } \lambda||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\eta_i$$
$$\text{subject to } y_i w_i^T x_i \geq 1 - \eta_i$$

where $\eta_i$ would influence the width of the margin.

In 1997, the SVM was extended to regression problems called SVR (support vector regression)[5]. The idea is similar to what we have talked about the margin. The optimization problem is,

$$\text{minimize } \frac{1}{2}||w||^2$$
$$\text{subject to } |y_i - w^T x_i - b| \leq \epsilon$$

It can be understood as we wanted to have as much data points as possible inside the margin and the data points inside the margin would not contribute to the loss, only datapoints outside the margin would contribute to the loss (compared with the interpretation of SVM).

To be more general, SVR can be extended to non-linear relationship with a trick called kernel. Basically speaking, kernel is a non-linear map to change the features to higher dimention. One popular kernel is called Radial Basis Function (Gaussian Kernel), the kernel function is defines as,

$$K(x, z) = e^{\frac{-||x-z||^2}{\sigma^2}}$$

compared with linear kernel for the original SVR model,

$$K(x, z) = x^T z$$

What we basically do is changing all the inner products in the gradient steps to the kernel function. And specific for Gaussian Kernel, its corresponding feature vector is infinite dimensional. Thus it can better depict the data relationship above the linear relationship.

Thus, in our research, we also replace the linear kernel to Radial Basis Function and compare the performance of the factor model.

## 1.8 Neural Network

Neural network are devised in the mid-20th as a computational model of the human brain. A neuron is a cell that has serval inputs that can be activated by some outside process. Depending on the amount of activation, the neuron can be either active or inactive. The structure shows in Figure 4(`http://scikit-learn.org/stable/modules/neural_networks_supervised.html#regularization`):

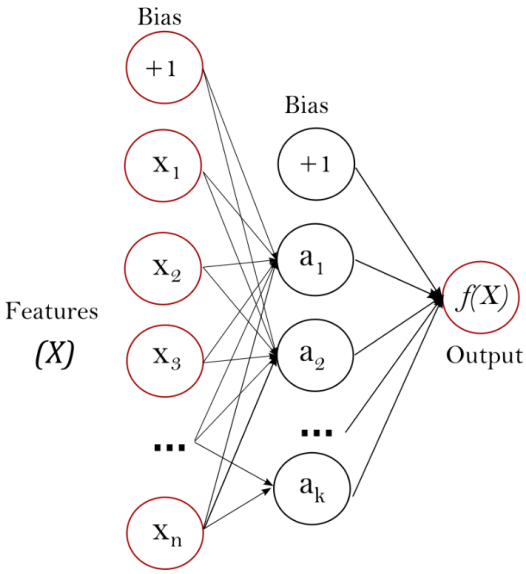



Figure 4: Interpretation of Neural Network

With a hierarchical structure, Neural Network is able to perfectly fit any given data, as long as there are no constraints on the number of hidden cells. Since the super strong fitting power of Neural Network, we tried to catch some non-linear structure of our factor model with the help of carefully choosing the number of hidden layers, the activate functions and the loss function. The detailed choice will be showed in empirical analysis.

One advantage of Neural Network is the ability of handling high dimension data. The structure of the model reduce the dimension and choose the representatives automatical, which is very convenient when dealing with high-dimension data. One possible drawback of the neuron network is the representative is not that obvious as the simpler model(like OLS), since there is no direct one-to-one coefficient-factor pairs. The hierarchical structure, though has the power of non-linearity, also makes the interpretation harder.

# 2 Strategy

## 2.1 Algorithm

The strategy uses the same logic as in Markowitz setting. However, it provides different ways to draw information from past data, and provide better views on expected returns as well as covariance.

The strategy rebalances on a monthly basis (21 trading days). At the end of each month, we draw daily data from a recent training window (denoted as training period), for example one month, including the factors matrix $F_{\text{month}_k}$, and the return matrix $R_k$ of all the assets. The training period for our models is not necessarily one month, and we set it a parameter to be examined.

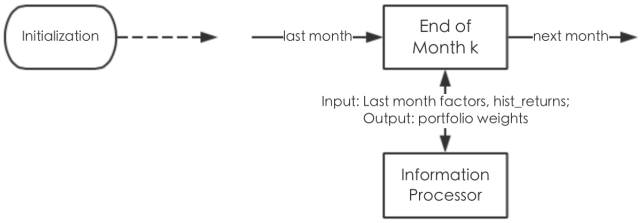



Figure 5: Algorithm at month k

At inception, assets are equally weighted. After month $k$, first carry out PCA on $F_{\text{month}_k}$ to reduce the dimension of the factors. Then, pass the new factors and historical return matrix into selected models to generate expected return, as well as covariance estimation. Finally, plug in the expected return and covariance matrix into Markowitz setting to derive the optimal weights for the next month.

The complete algorithm is described below.

Table 2: Strategy Algorithm

Using prediction method $Method(k, F_{\text{month}_k}, R_k, \text{freq}, t)$

Input: Factor data $F$, return matrix $R$, rebalancing frequency, training period t;
Output: Sharpe Ratio, Maximum Drawdown, Cumulative Return (vector)

Algorithm Start:
Number of trading times in total $K_{\max} = \lfloor \frac{T-t}{\text{freq}} \rfloor$.
for $k = 0, 1, \cdots, K_{\max}$:
    if $k == 0$ :
        $\omega = (\frac{1}{d+1}, \cdots, \frac{1}{d+1})^T$
    end
    else:
        $F'_{\text{month}_k} = \text{PCA}(F_{\text{month}_k})$
        $R_{\text{expect}},\ \Sigma_{\text{expect}} = Method(k, F'_{\text{month}_k}, R_k, \text{freq})$
        $\omega = Markowitz(R_{\text{expect}}, \Sigma_{\text{expect}})$
    end
    strat_return[k+1] = $\omega^T R_{k+1}$
end
SR = Shape_ratio(returns[1:($K_{\max}$+1)])
CR = Cumprod(1+returns[1:($K_{\max}$+1)])
MD = Maximum_drawdown(CR[1:($K_{\max}$+1)])
return(SR,MD,CR)

Algorithm End.

## 2.2 Evaluation

### 2.2.1 Cumulative Wealth

We often assume we have $ 1 at the beginning, and then we can calculate the cumulative wealth for our strategy according to the formula mentioned in the last section. Generally, higher cumulative wealth, better the strategy is. It should be mentioned that when we want to compare the results for different datasets, we'd better use annulized percentage yield, which is defined as $APY = \sqrt[y]{W_n}$ because the final wealth is dependent with time.

### 2.2.2 Sharpe Ratio

Annualized sharpe ratio is one of the good ways to measure the risk-adjusted return for portfolios. Sharpe ratio is defined as:

$$\text{SR} = \frac{\text{APY} - R_f}{\sigma_p}$$

The higher the sharpe ratio, the higher return per risk taken.

### 2.2.3 Maximum DrawDown

Maximum draw down is a good method to measure the tail risk of the portfolio, which measure the decline from a historical peak of portfolio cumulative wealth. Maximum draw down is define as:

$$\text{MDD} = \sup_{t\in(0,n)} \{\sup[0, \sup_{i\in(0,n)} S_i - S_t]\}$$

The smaller the maximum draw down, the more drawdown risk the strategy can tolerate.

## 2.3 Benchamark Strategy

### 2.3.1 Historical Estimator Method

This is the original Markowitz method. We estimate the mean vector and convariance matrix from the past return data and take them into the mean variance method to give out the optimal allocation decision.

### 2.3.2 OLS Factor Model

Taking the factor-after-PCA as the independent variable and fit a OLS regression with return data to the factors, we can get the estimators for the mean vector and convariance matrix. When taken them as inputs, Markowitz can give out the optimal weights for asset of the next period. This is the most important benchmark for us because we revise the regression methods for factor model, if our revision can make improvements, our back testing results should outperform the OLS factor model.

# Part III

# Empirical Analysis

## 1 Data Description

Because we research into factor models, there are two main data for us. One is the factor data, the other one is the return data. And we merge the two datasets together to keep the data consistent.

### 1.1 Factor Data

There are two main sources for our factor data. The first one is from Fama-French five factor model (`http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`). The other ones are from our peers, Zile Huang, Lin Zeng and Ke Zhang (but we calculate on our own because we want data with higher frequency). And the data starts from 1995-01-05, and ends at 2017-03-31.

The summary of the factors is shown in the table 3.

Table 3: Summary of Factors

| Factors | Defination |
|---|---|
| Mkt - Rf | The excess return of the market portfolio |
| SMB | The return differce of portfolio of small cap stock with portfolio of large cap stocks |
| HML | The return differce of portfolio of high book to market value stocks with portfolio of low book to market valuestocks |
| RMW | Difference between the returns on diversified portfolios of stocks with robust and weak profitability |
| CMA | Difference between the returns on diversified portfolios of low and high investment stocks, which we call conservative and aggressive |
| Term Spread | The yield difference between 10-year treasury bond with 3-month treasury bill |
| Default Spread | The yield difference between BAA corparate bond with AAA corporate bond |
| Short-term Interest Rate | The differenced yield of 3 month treasury bill |
| Long-term Interest Rate | The differenced yield of 20 year treasury bond |
| VIX | The VIX index |
| Oil Price | The WTI oil price |
| Oil Growth | The Growth rate of oil price |
| Dollar | The US Dollar index |
| Gold to Platinum Ratio | The ratio of price of gold to the price of platinum |

### 1.2 Return Data

We have chosen two datasets to get the return data. One is for stock market and the other one is for ETF market.

#### 1.2.1 Stock Data

Value stock are more liquid, thus it is more suitable for a long-term investor. Hence we choose value stocks which are components of DJIA (Dow Jones Industrial Average). Our stock pool contains 28 stocks from May 5, 1999 to March 31, 2017 because we want to have as long trading period as possible so that we can test the robustness of our methods.

#### 1.2.2 ETF Data

Small stocks have comparatively higher idiosyncratic risks and thus might not be super suitable for factor models. Thus we choose ETF data to lastly test our algorithms. There are 29 ETFs chosen by us, and some of them are ETFs for different sectors in USA, some are ETFs for other assets like bond and oil and some of them are equity ETFs for other regions across the world. The period starts at Dec 20, 2007 and ends at Mar 1, 2017.

## 2 Factor Model with Regularizer

### 2.1 Lasso Regression Method

As we mentioned earlier, when adding $l_1$ norm as the regularizer, the OLS regression would be changed to lasso regression. One of the most attractive feature for lasso regression is that it tends to give out sparse parameters, which is meaningful for factor models. The sparsity means we want to find out the most important factors for the asset.

For this part, we just take a sample from Dec 20, 2007 to June 2, 2014 with ETF as return to research whether it is reasonale to add regularizers becasue as we mentioned earlier, regularizers have a vital disadvantage, that it would add bias to the estimators.

As we mentioned earlier, one important parameter for regularized problem is to regularzier multiplier $\alpha$. When $\alpha$ is set to be 0, no regularizer is applied. The process of choosing multiplier is called validation. For our research, we choose 80% of the data as the training set and 20% of the data as validation set.

However, our research give out disappointing results. No matter how long horizon of the data we choose, the validation tends to give out either the higher bound of $\alpha$ or 0. Although for just a small fraction of tests, it may give out the $\alpha$ in between, we think it is not reasonably enough to apply lasso to the back testing part.

The reason for this phenomenon is that firstly, regularizer is designed to solve overfitting problems. And for our model, we have plenty of obervations and comparatively small number of features (after PCA). Thus overfitting actually is not a huge problem for us, which makes regularizer less useful, thus it tends to choose $\alpha = 0$. Secondly, the regression parameters for our models is really small due to the reason we use daily return (the scale is really small). You can think about the situation as a line really close to the x coordinate in the 2D dimention. And all the y just lie really close to the x coordinate. Thus, even though we set all $w$ to be zero, the total loss is still small. Thus, when $\alpha$ is large enough, all the $w$ is compelled to be 0. But for validation set, the loss may be smaller compared with OLS regression result (that is to say, when $\alpha = 0$). Thus, the validation algorithm may also give out the higher bound (or more accuraly speaking, the first $\alpha$ that compelled $w$ to be 0).

Due to the disappointing results, we did not take lasso regression into the future back testing.

### 2.2 Ridge Regression Method

We have similar process and results for ridge regression with lasso regression. The only difference is that the validation would only give out 0 as its chosen $\alpha$.

The reason is that as we mentioned earlier, the regression parameters for our models is really small. For ridge regression, we choose $l_2$ norm as our regularizer, which would make the regularization part much smaller than the loss function part compared with $l_1$ norm under lasso setting. Thus if we want to compel $w$ to be zero, we need to have a much larger $\alpha$. Thus, the ridge regression would tend to choose 0 as its $\alpha$.

Similarly, we draw the conclusion that ridge regression make little sense for our setting. Thus we did not take ridge regression into the future back testing.

# 3 Factor Model with Different Loss Functions

## 3.1 Huber Regression

In this part, we focus our research on ETF datasets since we think stocks are more exposed to idiosyncratic risks although we find out the results are similar for both datasets. And actually, in our preliminary researches, the performance for the two datasets are actually similar.

As discussed in the above sections, Huber loss function gives less weights to outliers, making it a more stable regression model. Empirical analysis demonstrates similar result, one of which is shown in figure 6.

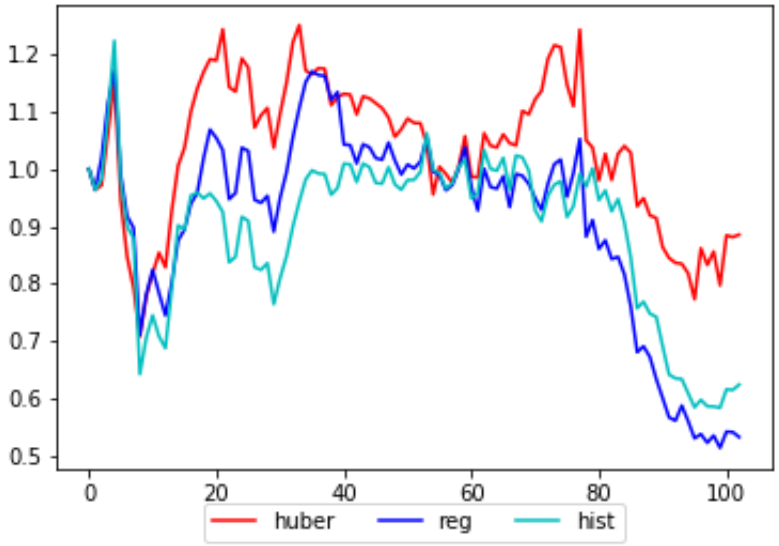



Figure 6: Plot for Backtesting with Huber Regression

Huber regression is the red line, while the blue line denotes OLS regression model. We can see that Huber beats OLS and provides a relatively more stable cumulative return than OLS regression. The parameter are $\epsilon = 1.2$.

The only parameter regarding the Huber regression is the $\epsilon > 1$, denoting the minimum distance for identifying the outliers. Taking $\epsilon \in \{1.1, 1.2, \cdots, 1.5\}$, the performance analysis is shown below.

Table 5: Huber regression sensitivity analysis

| $\epsilon$ | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 |
|---|---|---|---|---|---|
| Sharpe Ratio | -0.0351 | 0.0111 | 0.3228 | 0.0098 | 0.0043 |
| Maximum Drawdown | 0.3531 | 0.2998 | 0.0255 | 0.345 | 0.3553 |

The Sharpe Ratio reaches its maximum, and maximum drawdown takes its minimum value both when $\epsilon = 1.3$, which is very close to the default number 1.35.

The reason might credit to the outlier treatment. A small $\epsilon$ means we just treat really small error with $l_2$ loss but more other errors with $l_1$ loss. And we know OLS ($l_2$ loss) maintain lots of perfect properties. Thus if we treat considerable errors with $l_1$ loss, it would definately damage the performance of regression. A large $\epsilon$ is actually closer to OLS regression, subject to the influence of outliers. Thus it would have similar problem with OLS regression.

In our research for this part, we can find out the outliers would damage the performance of OLS regression. And huber regression can help solve the problem but with a suitable parameter $\epsilon$.

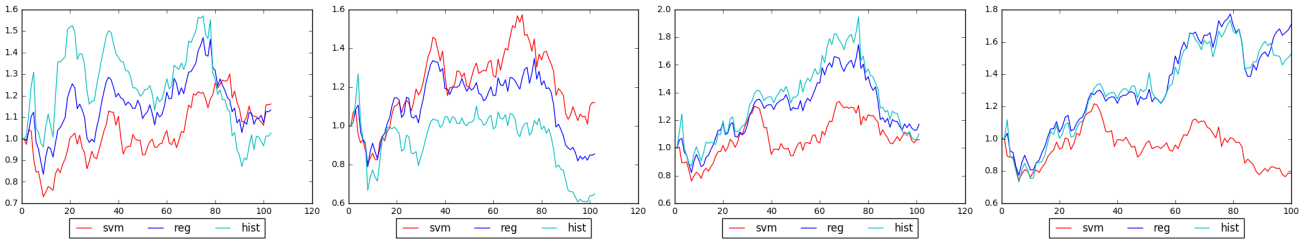
Therefore, the reasonable range for $\epsilon$ is [1.2,1.4]. Huber regression generates better and more stable performance within this interval.

## 3.2 SVM Regression Method

In this section, we use the SVM regression model to fit the ETF data and explore some properties of SVM model. In this part, we fit the model with SVM with Gaussian Kernel instead of linear kernel because in our preliminary researches, we find SVR with linear kernel cannot converge due to the reason the linear relationship cannot measure the data well. This is also consistent with the good performance of Huber regression (the outliers may be the reason that break up the linear relationship).

### 3.2.1 training period

We first show the results of different length of training periods. Training period refers to the time window of the training data, which means that it decides how long the history data the model takes as the input for the next re-balancing. The time window is chosen as month-based. The result is as Figure 7:



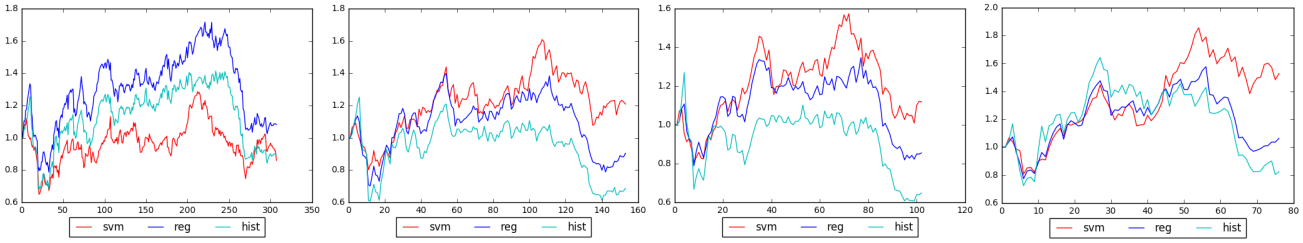

(a) 2 months (b) 3 months (c) 4 months (d) 5 month

Figure 7: Cumulative wealth of different training periods

From the above figure, we know that SVM model improves with a longer training period, and should have a relatively good performance at the time window of 3 months. We actually tested the training periods up to 12 months and noticed that after 3 months, all strategies performed worse as the training period goes longer. One possible explanation is that the information of factors would no longer have affect on the ETF returns after a certain period.(in our case, the threshold is around 3 months)

### 3.2.2 rebalance frequency

In the following sensitive test for the frequency of rebalancing, the training period is set to 3 months.




(a) 1 week (b) 2 weeks (c) 3 weeks (d) 4 weeks

Figure 8: Cumulative wealth of different rebalancing frequencies

From the figure 8, we can observe that SVM gradually beats two other benchmarks with lower rebalancing frequency. And Since the cumulative wealth didn't take the transaction cost into consideration. The performance of a high frequent rebalancing might be worse. And the high frequent rebalancing tends to be more risky (have a higher maximum drawdown showed in table 6.)

Table 6: sensitivity analysis of SVM rebalancing frequency

| frequency | 1 week | 2 weeks | 3 weeks | 4 weeks |
|---|---|---|---|---|
| Sharpe Ratio | -0.0278 | 0.02613 | 0.0249 | 0.1163 |
| Maximum Drawdown | 0.4194 | 0.3306 | 0.3586 | 0.2543 |

In this section, we can find that SVM model is able to perform better than OLS in terms of cumulative wealth, sharp ratio and maximum drawdown with carefully tuned time window size and rebalancing frequency. However, the model seems to be more sensitive than the Huber model and requires for manually searching for the best parameters.

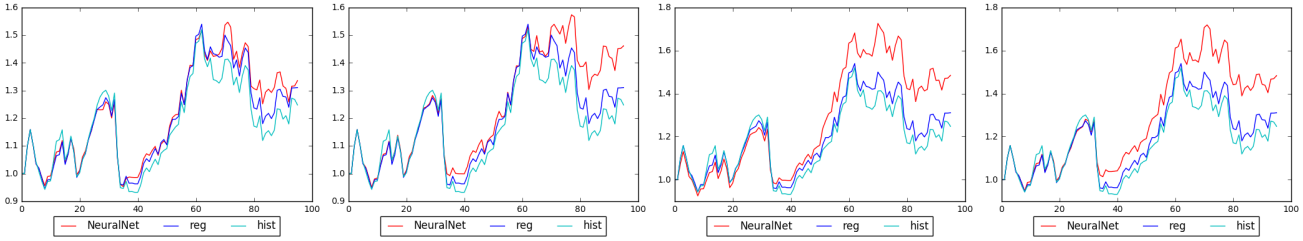
## 3.3 Neural Network

In this section, we tries several different architectures of Neural Network to optimize the cumulative return.

### 3.3.1 Depth of Neural Network

The depth of the Neural Network introduce the inductive biases to the model. We can observe from the Figure 9. In the shallow architecture (10 layers), the model fits the historical result perfectly, which means the model was too flexible and memorized the training data. In order to prevent the model capture the noisy information, a deeper structure introduces the biases and to some extend prevent overfitting. And due to applying Stochastic Gradient decent in each iteration, the model also introduce some randomness to reduce overfitting.

However, since a deeper neural network have some problem with vanish gradient, too deep structure will not continuously improve the performance. Also it usually takes much longer time to calculate and requires stronger computational power as the structure goes deeper.



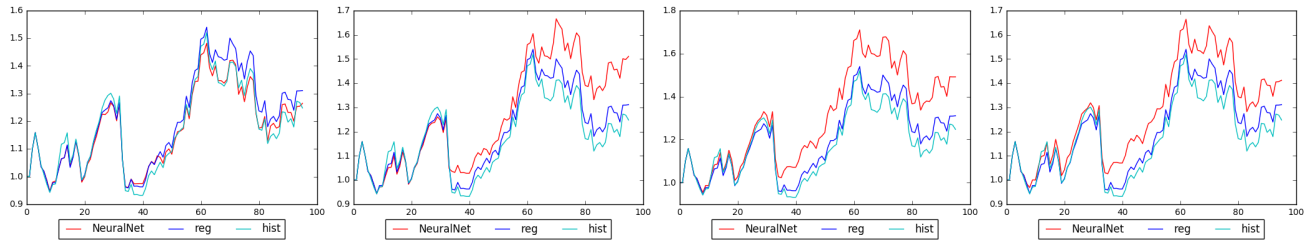

(a) 10 layers (b) 20 layers (c) 30 layers (d) 40 layers

Figure 9: Depth of Neural Network

### 3.3.2 Penalty of Neural Network

Since the Neural network is able to fit the data perfectly with too flexible architecture, we need to add an L2 penalty to prevent the model from overfitting the observed returns.

The penalty regularizer L is chosen as $L = 0, 0.1, 1, 10$. The result is as following:




(a) L=0 (b) L=0.1 (c) L=1 (d) L=10

Figure 10: penalty term of Neural Network

Without the penalty term, we can observe from the left most figure(L=0), the result is nearly the same as the historical data, which means Neural network perfectly fits the training data. Adding the penalty term improves the result of the cumulative wealth, which suggests that when applying neural network to fit the time series data, it's important to adapt techniques to prevent the overfitting, (e.g. by adding penalty terms or drop out).

Compared with the previous loss functions, neural network still can't beat the Huber regression, and since the unobserved corresponding coefficient-factor pairs, Neural Network doesn't performance that well in this setting. Also, during the implementation, if we increase the number of factors into the model, Neural network tends to performance better. So in the future, we might want to utilize the power of Neural Network when the input data has really high dimension.

## 4 Adjustments to Optimization Method

As we can see from above results, Huber regression gives the most stable results. Therefore, we set Huber for loss function, and analyze the performance of Markowitz optimization model with respect to different parameters.

We take all combinations of $\sigma$ and upper bound constraint for the weights, with $\sigma \in \{0.2, 0.3, \cdots, 0.7\}$, and upper $\in \{0.5, 0.6, \cdots, 1\}$. The Sharpe ratios are shown below.

Table 7: Markowitz Sharpe ratio sensitivity analysis

| | upper $= 0.5$ | upper $= 0.6$ | upper $= 0.7$ | upper $= 0.8$ | upper $= 0.9$ | upper $= 1.0$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.2$ | 0.0284 | 0.0255 | 0.0268 | 0.0275 | 0.0328 | 0.031 |
| $\sigma = 0.3$ | 0.0107 | -0.0015 | -0.0045 | -0.0059 | -0.0086 | -0.0075 |
| $\sigma = 0.4$ | 0.012 | -0.0025 | -0.0121 | -0.0224 | -0.0225 | -0.0233 |
| $\sigma = 0.5$ | 0.0061 | -0.0113 | -0.0254 | -0.0363 | -0.0425 | -0.0453 |
| $\sigma = 0.6$ | 0.0059 | -0.012 | -0.0287 | -0.0406 | -0.0501 | -0.0565 |
| $\sigma = 0.7$ | 0.007 | -0.0112 | -0.0266 | -0.0411 | -0.0511 | -0.0588 |

To see it more clearly, plot them on a grid, and a clear trend can be seen. Note that the axis has been reversed. According to the graph, the tighter constraints, the better Sharpe ratio. It can be explained as lower volatility constraints and lower higher bound constraints would all make our portfolio less volatile, which makes sharpe ratio comparatively higher.

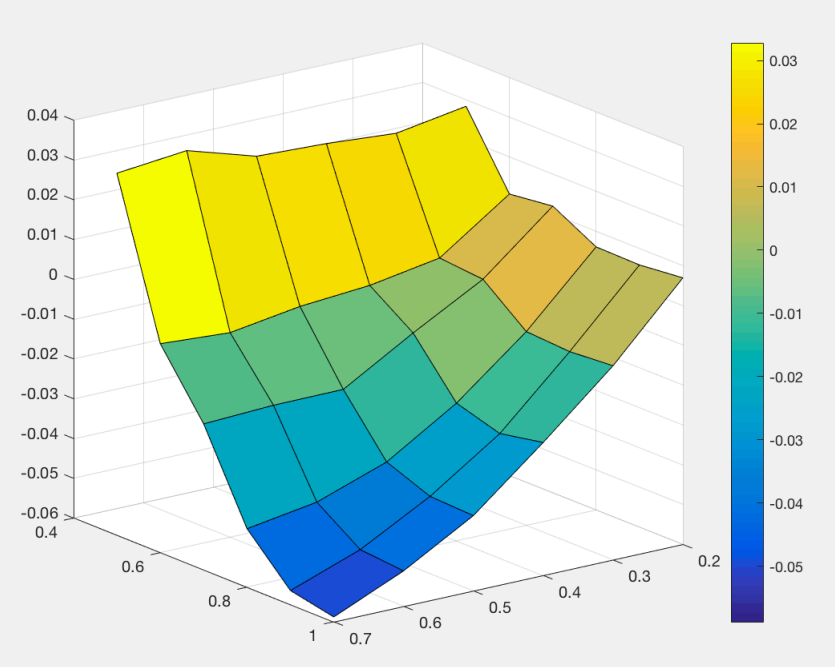



Figure 11: Plot for Markowitz sensitivity

Now we take a look at the maximum drawdown for the parameter combinations.

Table 8: Markowitz maximum drawdown sensitivity analysis

| | upper $= 0.5$ | upper $= 0.6$ | upper $= 0.7$ | upper $= 0.8$ | upper $= 0.9$ | upper $= 1.0$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.2$ | 0.3293 | 0.3228 | 0.3123 | 0.3053 | 0.3022 | 0.3015 |
| $\sigma = 0.3$ | 0.3969 | 0.4123 | 0.4105 | 0.4075 | 0.4076 | 0.4014 |
| $\sigma = 0.4$ | 0.3971 | 0.4222 | 0.4587 | 0.4937 | 0.4931 | 0.4964 |
| $\sigma = 0.5$ | 0.3971 | 0.4222 | 0.4808 | 0.5341 | 0.5713 | 0.6013 |
| $\sigma = 0.6$ | 0.3971 | 0.4222 | 0.4848 | 0.5411 | 0.601 | 0.6517 |
| $\sigma = 0.7$ | 0.3971 | 0.4222 | 0.4848 | 0.5435 | 0.6041 | 0.661 |

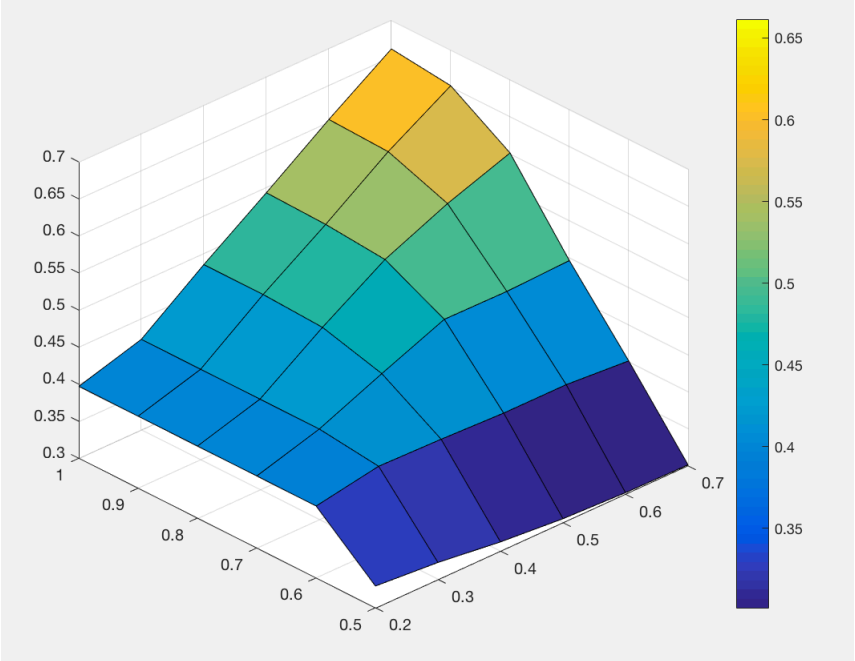



Figure 12: Plot for Markowitz sensitivity

Plotting the result on the grid, we discover that looser constraints result in larger drawdown. The result is intuitive, since looser constraints allow larger weights, meanwhile bigger expected volatility, which exposes the strategy to larger random risk.

# Part IV

# Conclusion

Factor models can mitigate the problems brought by historical estimates for application in Markowitz mean and variance methods. And with the development of machine learning techniques, more and more regression methods are introduced and may be a good candiadate for factor models. In our research, we change the OLS regression in original factor model by other regression techniques to see whether the performance can be improved.

Through our empirical analysis, we find adding regularizers is not a good choice since there would not be severe overfitting problems for linear factor models. And adding regularizer would add to bias to the model.

And then we change the loss function of OLS regression and Huber regression, support vector regression and neural network are applied to our datasets. And we find huber regression can consistently beat the original OLS regression for back testing trading strategy with Markowitz mean and variance method. SVR and neural network can achieve good performances but need carefully-chosen parameters, which limits its practical applications.

# Part V

# Reference

## References


[1] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[2] Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56, 1993.

[3] Eugene F Fama and Kenneth R French. Dissecting anomalies with a five-factor model. *Review of Financial Studies*, 29(1):69–103, 2016.

[4] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[5] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik, et al. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.