Eötvös Loránd University

Faculty of Science

Fóra Gyula Krisztián

Predictive analysis of financial time series

BSc Thesis

Supervisor:

Lukács András

Department of Computer Science

Budapest, June 2014

Acknowledgements

I would like to express the deepest appreciation to my supervisor Lukács András whose lectures on algorithms and data mining gave rise to a profound interest in me towards these areas of mathematics. Without his supervision and help this dissertation would not have been possible. I would like to thank my loved ones, who have supported me throughout the entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

Contents

Acknowledgements i

Contents ii

List of Figures iv

List of Tables v

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 The stock market . . . 2
  - 1.2.1 Technical analysis and data mining . . . 3
  - 1.2.2 Criticism of technical analysis and the EMH . . . 4

2 Introduction to Machine Learning 6
- 2.1 Supervised learning . . . 6
- 2.2 k-Nearest Neighbors . . . 7
- 2.3 Logistic Regression . . . 8
  - 2.3.1 Simple Logistic regression . . . 9
  - 2.3.2 L1 and L2 regularized Logistic regression . . . 11
- 2.4 Support Vector Machines . . . 12
  - 2.4.1 Linear SVM . . . 12
  - 2.4.2 Lagrange duality . . . 14
  - 2.4.3 Solving the optimal margin problem for the separable case 15
  - 2.4.4 The non-separable case . . . 16
  - 2.4.5 Solving the non-separable case . . . 17
  - 2.4.6 Kernels . . . 18
  - 2.4.7 The SMO algorithm . . . 20
- 2.5 Random Forests . . . 22
  - 2.5.1 Decision Trees . . . 22
  - 2.5.2 The Random Forest algorithm . . . 24

3 The Dataset 26
- 3.1 Dataset . . . 26
  - 3.1.1 The NASDAQ Composite Index . . . 26
  - 3.1.2 Data sources . . . 27
- 3.2 Features . . . 27

3.2.1 Technical indicators 27
3.2.2 Economic variables 28
3.2.3 Feature extraction based on lagged correlations 29

4 Development environment **30**
4.1 Development environment 30
4.1.1 Python 30
4.1.2 Python libraries 30

5 Model selection and testing **32**
5.1 Classification model and performance measures 32
5.1.1 Model 32
5.1.2 Performance measures 32
5.2 Trading strategy 33
5.3 Benchmark results 34
5.3.1 Buy-only 34
5.4 Model training and Validation 35
5.4.1 Parameter selection and validation 35
5.4.2 Noise filtering 35
5.5 Results 36
5.5.1 k-NN 38
5.5.2 Logistic Regression 38
5.5.3 Support Vector Machine 39
5.5.4 Random Forest 41
5.6 Summary of Results 41

6 Conclusion **42**

Bibliography **43**

List of Figures

2.1 Geometry of SVMs 13
2.2 Illustrating the constraints for two αs 21

3.1 NASDAQ Close Price 26
3.2 Daily % change histogram 27

5.1 Buy-Only performance 34
5.2 Inc-Dec Center Euclidean distance 36
5.3 Inc-Dec Center Cosine similarity 36
5.4 Parameter selection for Logistic Regression 38
5.5 L2 logistic regression performance 39
5.6 RBF grid search 39
5.7 RBF-SVM performance 40
5.8 SVM 40
5.9 Logistic regression 40

List of Tables

5.1 Benchmark results 36
5.2 Classification results: k-NN and Logistic Regression 37
5.3 Classification results: Random Forest and SVM 37

Chapter 1

Introduction

1.1 Introduction

Machine learning and other predictive statistical methods play an increasingly major role in applied economics. The continuous increase in the data available calls for novel methods to explore the patterns that interest economic scientists. One field where predictive analytics triumphs is financial forecasting. It is hard to find an other field where motivation can be so purely translated into currency units. In financial prediction an algorithm slightly better in almost any sense could instantly make anyone rich. This observation results in a fascinating field of research where cutting edge algorithms are constantly researched and used extensively, however these results are mostly kept as secrets. In this thesis I will write about machine learning techniques effectively used in predictive stock market analysis, although we will see that the word "effective" has a different meaning when it comes to forecasting future prices.

The main goal of this thesis is to give a general overview of the viability of technical analysis in stock market forecasting combined with state-of-the-art machine learning techniques. I do not aim to propose a trading strategy that can beat the market and make anyone rich (if I could I would not be writing this right now), but to find supporting empirical evidence to the claims, technical analysis makes.

In this first introductory chapter, I will give a short description of the stock market and the basic problem of prediction along with efficient market hypothesis often cited as a critique for these methods. In the second chapter I will lay down the foundations of the machine learning algorithms I used for prediction and I

will also describe the computational algorithms behind them. In the third and fourth section I will perform my empirical analysis on the NASDAQ stock index and compare the results of the different methods presented in chapter two. In the final chapter I will propose some areas of future research, and give my concluding remarks.

1.2 The stock market

The idea behind the stock market is very easy to grasp. By definition it is "the market in which shares of publicly held companies are issued and traded either through exchanges or over-the-counter markets" [1]. It also stands as a symbol of capitalism and the free-market economy where everyone (who is willing to pay the price) can own a small percentage of the desired firm. There are basically two types of stock exchanges, the auction and the dealer market. Now I will describe the substantial difference between these types by introducing two of the biggest stock exchanges in the world, the NYSE and the NASDAQ.

NYSE

The New York Stock Exchange, based in New York City is the world's largest stock exchange where brokers can trade approximately 8000 listed issues. It was established in 1792 and plays a major role in world economics ever since. "It represents one-third of the world's equities trading volume" [2]. The NYSE is an auction market, where competing buyers and sellers enter their bid and sell prices at the same time and a specialist matches the the highest bid with the lowest sell and executes the transaction setting the transaction price as the new market price. In this case the buyer and seller trade directly with each other made possible through the market specialist. Orders are placed either personally (floor trading) or electronically [3][4].

NASDAQ

The NASDAQ (National Association of Securities Dealers Automated Quotations) stock exchange is the second largest stock exchange after the NYSE. It was founded in 1971, as the world's first fully electronic stock market with no floor trading possible. In contrast with the NYSE auction based trading system, the NASDAQ stock exchange uses a dealer market system. The mean difference between the dealer and the auction market is that in a dealer market, the buyer and the seller don't directly interact with each other, but only with a dealer (market maker). Market makers are large investment companies. In this case

market makers maintain inventories of the stocks they are selling and have to post a two sided quote -bid/ask- on which they accept transactions. The number of market makers and the constraint on the size of the bid-ask spread force market competition and thus leads to effective trading [3][4].

1.2.1 Technical analysis and data mining

Let us begin with the definition of technical analysis: *"Technical analysis is the study of market action, for the purpose of forecasting future price trends."* Market action basically means, "what happened on the market", which can be translated into the change in prices and volumes over time. Before explaining the rationale behind it let me draw parallels between technical analysis and data mining. Predictive data mining is the "the process of discovering interesting and useful patterns and relationships in large volumes of data" [5]. We could easily say that technical analysis is exactly data mining on stock market data with the intention of providing accurate predictions for future price movements. Technical analysis uses a large collection of statistics computed from historical market data to give useful heuristics for prediction. While many say that technical analysis is a form of art instead of an exact scientific method, most of the techniques used can be formalized, into simple decision rules very much like in classification problems.

Although it is easy to see the similarities in methodology, we still need to justify why technical analysis should be possible in the stock market at all. Most methods in technical analysis are rooted in the several articles Charles Dow published in the *Wall Street Journal* [6] His company with Edward Jones was the first to publish the first stock index in 1889, which was the average closing price of eleven stocks. Dow's ideas about market speculation are what we now call as *Dow's Theory*, and it consist of six basic tenets that constitute the rationale behind technical stock market analysis.

Basic tenets [6]

1. **The averages discount everything.**
   The idea is that all information that is needed about the market is already present in the stock prices. No additional information, like exchange rates or commodity prices is useful, it is already in the price.

2. **The market has three trends.**
   The three types are; *primary, secondary* and *minor*. The primary trend

is the main direction of movement which could last from less than a year to several years. The secondary trend, or intermediate reaction may last from several days to a couple of months, and is responsible for 30-60% of price movements. The minor trends or swings generally last from hours to days.

3. **Major trends have three phases**
   The three phases are; *accumulation, public participation* and *distribution.* The accumulation phase is when the most rational and informed investors trade, who are the fastest to react any insider or global information. The public participation phase is when rapid volume and price change occurs. This is when trend followers start to "buy-in" motivated by the improving business conditions. The last, distribution phase, is when the better economic conditions become publicly apparent and public speculative trading increases.

4. **The averages must confirm each other.**
   Although Dow referred to two well defined averages, the concept behind the idea is that signals should be able to confirm each other. One signal may not be signal at all.

5. **Volume must confirm trend.**
   Dow considers volume as an important secondary trend which should move together with the primary trend. Simply put, in an uptrend where price constantly increases, volume would increase too. On the contrary in a downtrend, when price falls, there will be a fall in volume also.

6. **A trend is assumed to be in effect until it gives definite signals that it has reversed.**
   This is self-explanatory it simply implies that, whenever a change in trend occurs there will be definite signs to support it.

We can see the basic ideas of technical analysis relate to asymmetric market information and the exploitation of human feelings, such as greed or fear. While these information-theoretical or psychological concepts are not articulated explicitly, it is hard to imagine that technical analysis could work without them.

1.2.2 Criticism of technical analysis and the EMH

Technical analysis is a highly debated topic among economists, while many use it and agree with its methodology based on behavioral psychology, there is a

great share of them who simply think it is useless and is against basic economic principles. In fact there is a Nobel-prize winning economic theory called the *Efficient Market Hypothesis* by Eugene Fama which contradicts all the basic assumptions of technical analysis. There are several forms of the EMH in which I will not go into detail. The EMH says that market prices fully and instantly reflect all the available information, basically saying that future prices cannot be predicted from past market information.

The concept of EMH can be summarized in a good joke I found very fitting [7]

"An economist is strolling down the street with a companion. They come upon a $100 bill lying on the ground, and as the companion reaches down to pick it up, the economist says, 'Don't bother – if it were a genuine $100 bill, someone would have already picked it up."

The efficient market hypothesis is one of the most controversial theories in economic sciences, even after decades of research economists are yet to reach a consensus whether financial markets are truly efficient or not. While the EMH has theoretical implications for the usage of technical analysis it is not my goal in this thesis to prove or disprove these, but I think it is very important to have an overview of the market-theoretical criticism.

Chapter 2

Introduction to Machine Learning

In this chapter after laying out the foundations of supervised learning and binary classification, I will introduce some of the most popular off-the-shelf classification algorithms. The algorithms presented in this section will give the base of my empirical analysis. I will discuss three types of algorithms, first I will explain the k-Nearest Neighbors method which uses a very simple basic idea but also gives a good foundation for more sophisticated methods. The next algorithm I will present is simple and regularized Logistic Regression, which is a highly popular classification method in statistical analysis. After logistic regression I will introduce the Support Vector Machines and the Random Forest classifiers which can be considered state-of-the-art methods in machine learning.

2.1 Supervised learning

Definition 2.1.1. ***Supervised learning*** is the task of learning a target function $T : X \to Y$ that maps an example set X to the set of possible labels Y [8]. On the condition that T gives a "good" approximation for the labels on the training set:

$$T(x^{(i)}) \approx y^{(i)}, \quad i = 1 \ldots, m. \tag{2.1}$$

The target function T is also called the classification model if Y is discrete valued and regression model if Y is continuous.

Definition 2.1.2. ***Binary classification*** is a classification problem in which the set of class labels Y contains only two elements. Without loss of generalization from now on we will assume that $Y = \{0, 1\}$.

Now I will define the decision boundary, associated with a decision function T.

Definition 2.1.3. ***Decision boundary*** is an N-dimensional hyper-surface which partitions the points in the underlying vector space in two sets, one for each class.
A decision boundary is associated with the classifier C if all points in one partition are classified as 0 and all points in the other as 1.

Definition 2.1.4. ***Linear classification model*** is a decision function $T(x)$ for which the decision boundary is a hyper-plane in the N dimensional Euclidean space.

Later in this chapter we will see that there are basically two types of classification models. The first category assumes the dataset to be generated by some probability distribution. In this case the conditional probability distribution of Y on condition of X is modelled (also called discriminant function methods) I will go into further details when discussing logistic regression. The other category is when the decision boundary is explicitly modeled (optimal separating hyperplane methods). The appropriate model choice depends heavily on the assumptions we can make about the dataset [9].

2.2 k-Nearest Neighbors

One of the simplest yet useful classification algorithm is the *k-Nearest neighbors* algorithm which can work exceptionally well with dense input space. The idea behind the algorithm is if a an input vector is "close" or "similar" to an other input vector, it is likely that they will have the same class labels.

k-NN classifier

$$T(x) = \arg\max_{v} \sum_{(x_i, y_i) \in D_z} I(v = y_i), \tag{2.2}$$

where D_z is the set (x_i, y_i) pairs for which x_i is the closest k vectors to x by some distance or similarity measure. The classifier in this case simply takes a

majority vote among the k nearest points to decide the predicted class label. A cheap extension of this model to weight the "votes" of the points by their distance (w_i), assigning larger weight to closer points [8].

Distance weighted k-NN

$$T(x) = \arg\max_{v} \sum_{(x_i,y_i)\in D_z} w_i \times I(v = y_i). \tag{2.3}$$

k-NN often serves as a good benchmark in almost any classification problem, for its easy implementation and straightforward interpretation.

2.3 Logistic Regression

Logistic regression is a simple and widely used linear classification model. It is a member of a family of linear regression models called Generalized Linear Models (GLM). In this section I will describe the assumptions and the derivation of the method, and also different regularization methods used to increase forecasting accuracy [10].

By assuming that the samples $(x^{(i)}, y^{(i)})$ are generated by some probability distribution X and Y, finding the target function can be viewed as finding the $y \in Y$ for which the conditional probability given x is the highest:

$$T(x) = \arg\max_{y\in Y} P(Y = y|X = x). \tag{2.4}$$

A popular way of constructing classification models is to model the conditional probability distribution $P(Y|X = x)$ as a function of x characterized by some parameter θ. Given a function form, learning the model is the process of finding the optimal θ parameter. There are two main approaches in finding θ, one is the *maximum likelihood* estimation which corresponds to the frequentist ideas, and the other is the *maxmimum aposteriori* estimation which corresponds to the Bayesian approach:

$$L(\theta) = \prod_{i=1}^{m} P(Y = y^{(i)}|X = x^{(i)}; \theta) \tag{2.5}$$

$$\theta_{ML} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \log L(\theta) \tag{2.6}$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log P(Y = y^{(i)}|X = x^{(i)}; \theta). \tag{2.7}$$

The function $L(\theta)$ is called the likelihood function, and $l(\theta) = \log L(\theta)$ is the log-likelihood function.

This maximum likelihood estimate will be used to find parameters for simple logistic regression. The Bayesian view where θ is regarded as a random variable too, gives us a different estimation. Now we can pick θ based on maximizing the conditional probability of the parameter given the training set:

$$\widehat{\theta}_{MAP} = \arg\max_{\theta} \prod_{i=1}^{m} L(\theta)p(\theta) = \arg\max_{\theta} [l(\theta) + \log p(\theta)]. \tag{2.8}$$

with $p(\theta)$ being some prior distribution on the parameters. We will see that this technique will be used later as an effective method of model regularization for logistic regression.

2.3.1 Simple Logistic regression

With simple logistic regression the label variable Y is assumed to have Bernoulli conditional distribution given x feature vector, parametrized by a function dependent on some linear combination of the input features.

$$\begin{aligned} P(y = 1|x; \theta) &= h_\theta(x) \\ P(y = 0|x; \theta) &= 1 - h_\theta(x). \end{aligned} \tag{2.9}$$

Or more compactly:

$$P(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}. \tag{2.10}$$

The function $h_\theta(x)$ is called the link function. In this case let us define $h_\theta(x)$ as:

$$\begin{aligned} h_\theta(x) = g(\theta^T x) &= \frac{1}{1 + e^{-\theta^T x}} \\ g(z) &= \frac{1}{1 + e^{-z}}. \end{aligned} \tag{2.11}$$

Assuming an independently generated sample we can write the log likelihood as a function of the parameter θ:

$$\begin{aligned}
\log L(\theta) &= l(\theta) = \log p(\overline{y}|X;\theta) && (2.12)\\
&= \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)\\
&= \log \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}\\
l(\theta) &= \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})). && (2.13)
\end{aligned}$$

This yields the following optimization problem:

$$\arg\max_{\theta} l(\theta). \tag{2.14}$$

Theorem 1. ***Convexity***
The optimization problem defined in 2.14 can be solved by convex optimization.

Proof.

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})).$$

After substituting in for h_θ we get:

$$\begin{aligned}
l(\theta) &= \sum_{i=1}^{m} y^{(i)} (\theta^T x^{(i)}) - (1 - y^{(i)}) \log (1 + \exp \theta^T x^{(i)}) && (2.15)\\
&= \sum_{i, y^{(i)}=1} (\theta^T x^{(i)}) - \sum_{i, y^{(i)}=0} \log (1 + \exp \theta^T x^{(i)}). && (2.16)
\end{aligned}$$

To show that $l(\theta)$ is a concave function we only need to show that Hessian-matrix is negative semi definite. First let's compute the second partial derivatives:

$$\frac{\partial}{\partial j \partial k} l(\theta) = - \sum_{i, y^{(i)}=0} x_j^{(i)} x_k^{(i)} \frac{e^{\theta^T x^{(i)}}}{(1 + e^{\theta^T x^{(i)}})^2}. \tag{2.17}$$

$$
\begin{aligned}
a^T H a &= -\sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k \left(\sum_{i,y^{(i)}=0} x_j^{(i)} x_k^{(i)} \frac{e^{\theta^T x^{(i)}}}{(1+e^{\theta^T x^{(i)}})^2} \right) \\
a^T H a &= -\sum_{i,y^{(i)}=0} \frac{e^{\theta^T x^{(i)}}}{(1+e^{\theta^T x^{(i)}})^2} \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k x_j^{(i)} x_k^{(i)} \\
a^T H a &= -\sum_{i,y^{(i)}=0} \frac{e^{\theta^T x^{(i)}}}{(1+e^{\theta^T x^{(i)}})^2} \left(\sum_{j=1}^{n} a_j x_j^{(i)} \right)^2 < 0. \qquad (2.18)
\end{aligned}
$$

$\Rightarrow a^T H a < 0$ for $\forall a \in R^n \Rightarrow l(\theta)$ is concave therefore 2.14 can be solved by convex optimization.

□

2.3.2 L1 and L2 regularized Logistic regression

The main purpose of regularization in machine learning is to increase the predictive performance of the model (generalization) by imposing constraints on the model parameters. It seems intuitive to prefer models to assume that most components of θ(the coefficients for different variables in the linear combination) should be close or equal to zero, especially in high dimensional settings. Also a model with more small coefficients is preferred to one with only a few big coefficients (please note that the words small and big only have meaning in the actual context).

This can be achieved by creating a cost function $C(\theta)$ in the form of

$$C_{L2}(\theta) = -l(\theta) + \frac{1}{2}\lambda\|\theta\|_2^2. \qquad (2.19)$$

There are alternative formulations of this type of regularization that are widely used[1]. My empirical results will correspond to this type of formulation of the regularization problem. Penalizing L2 norm (Euclidean length) of θ or similarly the L1 norm by:

$$C_{L1}(\theta) = -l(\theta) + \lambda\sum_{i=1}^{n}|\theta_i|. \qquad (2.20)$$

[1]Scikit-learn uses: $C_{L2}(\theta) = -\lambda l(\theta) + \|\theta\|_2^2$

This type of formulation of the cost function can be reached by a Bayesian approach to the classification problem and introducing an a priori distribution of θ and computing the MAP estimate [11].

Minimizing C_{L2} is equivalent to the MAP estimation of Y with θ having a normal a priori distribution of:

$$\theta \sim N(0, \lambda^{-1} I).$$

Proof.

$$p(\theta) = \frac{1}{\sqrt{(2\pi)^n |\lambda^{-1} I|}} \exp\left(-\frac{1}{2}\theta^T \lambda I^{-1} \theta\right) \tag{2.21}$$

$$\begin{aligned} p(\theta) &= \frac{1}{\sqrt{(2\pi\lambda^{-1})^n}} \exp\left(-\frac{1}{2}\lambda \|\theta\|^2\right) \\ \widehat{\theta}_{MAP} &= \arg\max_{\theta}[l(\theta) + \log p(\theta)] \\ &= \arg\max_{\theta}[l(\theta) + \log \frac{1}{\sqrt{(2\pi\lambda^{-1})^n}} - \frac{1}{2}\lambda \|\theta\|_2^2] \\ &= \arg\max_{\theta}[l(\theta) - \frac{1}{2}\lambda \|\theta\|_2^2]. \end{aligned} \tag{2.22}$$

□

Similarly in case of $C_{L1}\theta$ is assumed to have a Laplacian prior[2] [11].

2.4 Support Vector Machines

Support Vector Machines can be safely considered to be one of the state-of-the-art off the shelf machine learning algorithms today. Its popularity and effectiveness lies in the wide range of data structure it can model. Although it does not provide automatic and universal remedy to our predctive problems, if used correctly it can outperform most of the other methods currently available. As we will see, the algorithm that constructs SVMs is very elegant and can deal with non-linearity by using kernels. In this section I will introduce the basic idea behind SVMs and also show how it can be extended to non-linear data structure.

2.4.1 Linear SVM

In this section I will derive the quadratic optimization problem that characterizes Linear Support Vector Machines [12].

[2] $p(\theta) \propto \exp(-\lambda \|\theta\|_1)$

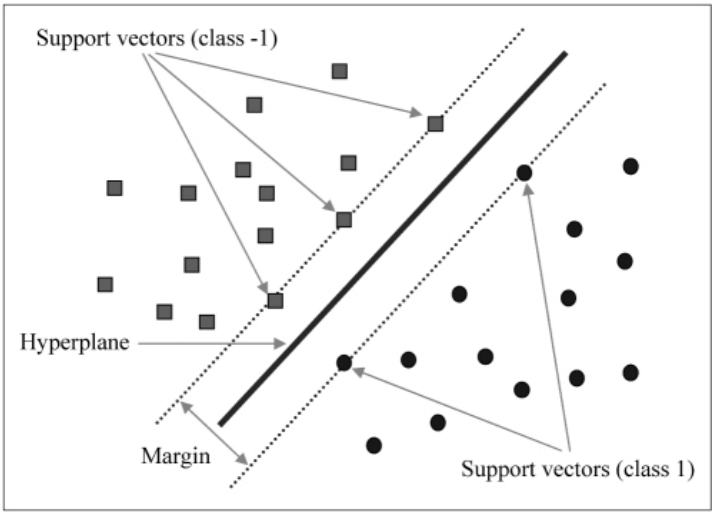

FIGURE 2.1: Geometry of SVMs

Let w and b denote the parameters of the separating hyperplane (L).

The signed distance of any point x to L is given by:

$$\frac{1}{\|w\|}(w^T x + b). \tag{2.23}$$

To maximize the margin between the two classes consider the following optimization problem:

$$\begin{aligned} &\max_{w,b,\|w\|=1} M \\ &y^{(i)}(w^T x^{(i)} + b) \geq M, \quad i = 1, ...m. \end{aligned} \tag{2.24}$$

The conditions ensure that all the points are at least M distance from the decision boundary, but the condition of $\|w\|$ is non-linear. Fortunately we can get rid of this by replacing the conditions with:

$$\frac{1}{\|w\|} y^{(i)}(w^T x^{(i)} + b) \geq M. \tag{2.25}$$

Since w is arbitrarily scalable, we can set it to $\|w\| = \frac{1}{M}$ which simplifies (by substitution and turning the maximization problem into minimizing the reciprocal) the above problem to the equivalent form of:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, ...m \quad . \end{aligned} \tag{2.26}$$

To solve this problem we need to introduce Lagrange duality [13].

2.4.2 Lagrange duality

$$\begin{aligned} &\min_w f(w) \\ &g_i(w) \leq 0, \quad i = 1, ..., k \\ &h_i(w) = 0, \quad i = 1, ..., l. \end{aligned} \tag{2.27}$$

To solve this optimization problem let us define the *generalized Lagrangian* as:

$$L(w, \alpha, \beta) := f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w). \tag{2.28}$$

The primal problem is defined as:

$$p^* = \min_w \max_{\alpha,\beta;\alpha_i \geq 0} L(w, \alpha, \beta) \tag{2.29}$$

$$= \min_w \theta_P(w). \tag{2.30}$$

The dual problem is defined as:

$$d^* = \max_{\alpha,\beta;\alpha_i \geq 0} \min_w L(w, \alpha, \beta) \tag{2.31}$$

$$= \max_{\alpha,\beta;\alpha_i \geq 0} \theta_D(\alpha, \beta). \tag{2.32}$$

Theorem 2. ***Weak duality***
For any primal/dual optimization problem, the following inequality holds: $d^* \leq p^*$.

Proof. Lets $\hat{w}$, $\hat{\alpha}$, and $\hat{\beta}$ be the solutions for 2.32. Then:

$$L(w, \hat{\alpha}, \hat{\beta}) \geq L(\hat{w}, \hat{\alpha}, \hat{\beta}) \quad \forall w \tag{2.33}$$

$$\max_{\alpha,\beta} L(w, \alpha, \beta) \geq L(w, \hat{\alpha}, \hat{\beta}) \quad \forall w \tag{2.34}$$

$$p^* = \min_w \max_{\alpha,\beta} L(w, \alpha, \beta) \geq L(w, \hat{\alpha}, \hat{\beta}) \geq L(\hat{w}, \hat{\alpha}, \hat{\beta}) = d^*. \tag{2.35}$$

□

Theorem 3. ***Slater's conditions(Strong duality)***
Consider a convex optimization problem of the form 2.27, whose corresponding primal and dual problems are given by 2.30 and 2.32. If there exists a primal feasible solution w for which each inequality constraint is strictly satisfied(i.e., $g_i(x) < 0$), then $d^* = p^*$

Theorem 4. ***Karush–Kuhn–Tucker***
If w^* is primal feasible and (α^*, β^*) are dual feasible and if

$$\nabla_w L(w^*, \alpha^*, \beta^*) = 0, \tag{2.36}$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \ldots, m, \tag{2.37}$$

then w^* is primal optimal, (α^*, β^*) are dual optimal, and strong duality holds.

The condition 2.36 is the standard gradient stationarity condition of unconstrained optimization and the set of equalities in 2.37 are called ***KKT complementarity conditions*** which imply that,

$$\alpha_i^* > 0 \Rightarrow g_i(w^*) = 0 \tag{2.38}$$

$$g_i(w^*) < 0 \Rightarrow \alpha_i^* = 0. \tag{2.39}$$

2.4.3 Solving the optimal margin problem for the separable case

To solve 2.26 using Lagrange duality we need to rewrite the inequality constraints as:

$$g_i(w, b) = -y^{(i)}(w^T x^{(i)} + b) + 1 \geq 0 \tag{2.40}$$

Now our problem is in the form of 2.27(note that there are no equality constraints and thus β) so the Lagrangian is:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1] \tag{2.41}$$

To solve the dual problem we first minimize L with respect to w and b by taking the derivatives. This yields the following equations:

$$\begin{aligned} \nabla L(w,b,\alpha) &= w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \\ w &= \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \qquad (2.42) \\ \frac{\partial}{\partial b} L(w,b,\alpha) &= \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \qquad (2.43) \end{aligned}$$

Substituting back to 2.41 we get:

$$\begin{aligned} L(w,b,\alpha) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} \\ &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)} x^{(j)} \rangle \qquad (2.44) \end{aligned}$$

To solve our initial problem we now need to macimize 2.44 with respect to α:

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)} x^{(j)} \rangle, \\ \alpha_i &\geq 0 \quad i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i y^{(i)} &= 0 \quad i = 1, ..., m \end{aligned}$$

2.4.4 The non-separable case

It is not always possible to find a separating margin, in this case, we would like to minimize the error. A sensible way of doing this is to let some point have a margin $1 - \xi_i$ and we will penalize these points with $C\xi_i$ where C controls the trade-off between the margin size and the error. In this case the optimization problem changes to the following:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \xi_i \qquad (2.45) \\ & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \\ & 0 \leq \xi_i, \quad i = 1 \ldots, m. \end{aligned}$$

2.4.5 Solving the non-separable case

To solve 2.45 using Lagrange duality we need to rewrite the inequality constraints as:

$$\begin{aligned} -y^{(i)}(w^T x^{(i)} + b) + 1 + \xi_i &\leq 0 \\ -\xi_i &\leq 0. \end{aligned} \tag{2.46}$$

Now our problem is in the form of 2.27(note that there are no equality constraints and thus β) so the Lagrangian is:

$$\begin{aligned} L(w, b, \xi, \alpha, r) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i] - \\ &\quad - \sum_{i=1}^{m} r_i \xi_i + C \sum_{i=1}^{m} \xi_i \end{aligned} \tag{2.47}$$

$$\xi_i, r_i \geq 0. \tag{2.48}$$

To solve the dual problem we first minimize L with respect to w,b and ξ_i by taking the derivatives. This yields the following equations (KKT stationarity-conditions):

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{2.49}$$

$$\begin{aligned} \frac{\partial L}{\partial \xi_i} = 0 \quad &\Rightarrow \quad C - \alpha_i - r_i = 0 \\ &\Rightarrow \quad 0 \leq \alpha_i \leq C \end{aligned} \tag{2.50}$$

$$\begin{aligned} \frac{\partial L}{\partial w} = 0 \quad &\Rightarrow \quad w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \\ &\Rightarrow \quad w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}. \end{aligned} \tag{2.51}$$

Substituting back to 2.47 we get the following optimization problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)} x^{(j)} \rangle, \\ & 0 \leq \alpha_i \leq C \quad i = 1 \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \quad i = 1 \ldots, m. \end{aligned}$$

The KKT dual-complementarity conditions are as follows:

$$\begin{aligned} r_i\xi_i &= 0 && (2.52)\\ \alpha_i[y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i] &= 0, \quad i = 0, \ldots, m. && (2.53) \end{aligned}$$

Using $C - \alpha_i - r_i = 0$ and our initial constraints (2.46), we can reformulate these conditions to:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1\\ \alpha_i = C &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1\\ 0 < \alpha_i < C &\Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1. \end{aligned} \quad (2.54)$$

We will use these conditions to verify if our algorithm found a maximum.

After solving for α_i we need to substitute back in 2.51 and 2.45 in order to get w and b. After doing so we can use the following equation to form predictions:

$$\begin{aligned} y = sign(w^T + b) &= sign((\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)})^T x + b)\\ &= sign(\sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b). \end{aligned} \quad (2.55)$$

Please note that using α_is we can express our prediction in only dot products which we will see will be extremely useful later.

2.4.6 Kernels

What we have learnt so far is enough to create SVMs that classify linearly separable (ar almost separable) data in the input feature space. It usually happens that our data is linearly non-separable in the original feature space but it becomes separable once we map the input vectors in some higher dimensional space. The only problem is that sometimes we want to compute such high dimensional feature mappings that it becomes computationally unaffordable to do so. What is fortunate about the optimization problem described above is that it does not

depend on the actual input vectors $x^{(i)}$ only on scalar products of two of these vectors. The main idea behind kernels is to provide an efficient way of computing scalar products between mapped vectors without explicitly calculating the transformed vector.

Let us denote some feature mapping as ϕ then corresponding kernel is:

$$K(x, z) = \langle \phi(x), \phi(z) \rangle. \tag{2.56}$$

In many cases computing $K(x, z)$ is very easy, a popular example of this wold be the polynomial kernel:

$$K(x, z) = (x^T z + c)^d. \tag{2.57}$$

Which corresponds to the feature space of all monomials of $x_1, x_2 \dots$ up to order d ($\binom{n+d}{d}$ dimensional). Despite the high dimensions, to calculate the scalar product using the polynomial kernel takes only $\mathcal{O}(n)$ time.

We can see that kernels can be a very useful addition to our toolbox, but how do we now if a kernel function corresponds to a valid feature mapping? To investigate this relation we first need to define the *Gramm-matrix* for a specific kernel.

Definition 2.4.1. ***Gramm-matrix***
Given a set of records $S : \{x^{(1)}, \dots, x^{(m)}\}, (m < \infty)$ and a kernel K, the Gramm-matrix is $G_S \in \mathbb{R}^{m \times m}$ for which $(G_S)_{ij} = K(x_i, x_j)$.

Theorem 5. ***Mercer's theroem***
Let $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. K is valid if and only if for $\forall\{x^{(1)}, \dots, x^{(m)}\}$ training set, the corresponding Gramm-matrix is symmetric positive semi-definite.

Proof. [14] If K is a valid kernel then there exists some $\phi(.)$ for which 2.56 holds. Let V be the matrix whose columns are $[\phi(x_1), \dots, \phi(x_m)]$, then

$$(G_S)_{ij} = V^T V. \tag{2.58}$$

From this it follows that G_{ij} must be positive semi-definite, since for all $z \in \mathbb{R}^m$, $(z^T V^T)(Vz) \geq 0$
Assuming that S is finite, since G_S is symmetric PSD there exists a real non-singular lower-triangular matrix L for which

$$(G_S)_{ij} = LL^T. \tag{2.59}$$

For $\phi(x_i) = l_i$ where l_i is the i^{th} row of L which gives a valid feature mapping for x.

□

We could already see that the polynomial kernel can greatly help us with computing high dimensional feature mappings in linear time. Now I will introduce the *Radial Basis Function (Gaussian)* kernel also known as RBF-kernel, which in contrast to the poly kernel, corresponds to an infinite dimensional feature mapping and is widely popular for it superior performance.

$$K_{RBF}(x, z) = \exp(-\gamma\|x - z\|^2). \tag{2.60}$$

Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close' [15].

So now going back to 2.52 and 2.55, we can see why it is so useful that model training and prediction are both expressed in dot products between training vectors. Using this fact we can use any Kernel that satisfies the necessary conditions to find maximal margin separating hyper-planes in any dimensions using the same optimization problem.

2.4.7 The SMO algorithm

In this section I will show a popular algorithm to solve the SVM dual optimization problem of 2.52

$$\begin{aligned}
\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)} x^{(j)}) \\
& 0 \leq \alpha_i \leq C \quad i = 1..., m \qquad (2.61) \\
& \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \quad i = 1, \ldots, m. \qquad (2.62)
\end{aligned}$$

The SMO algorithm is conceptually very similar to other QP solver algorithms in the that it divides the problem into smaller QP subproblems. In the case of

SMO, it divides it into the smallest possible QP problems with only two variables [16].

The main idea behind the algorithm is that in every iteration it maximizes W with respect to two Lagrange mulitpliers. We need to change at least two α_is in order to obey the linear equality constraint. What we will see is that this maximization can be done analytically so we do not need expensive inner loops, making the algorithm scale easily. After every iteration the two jointly optimized Lagrange multipliers are updated until the KKT conditions(2.54) are satisfied with some ε error.

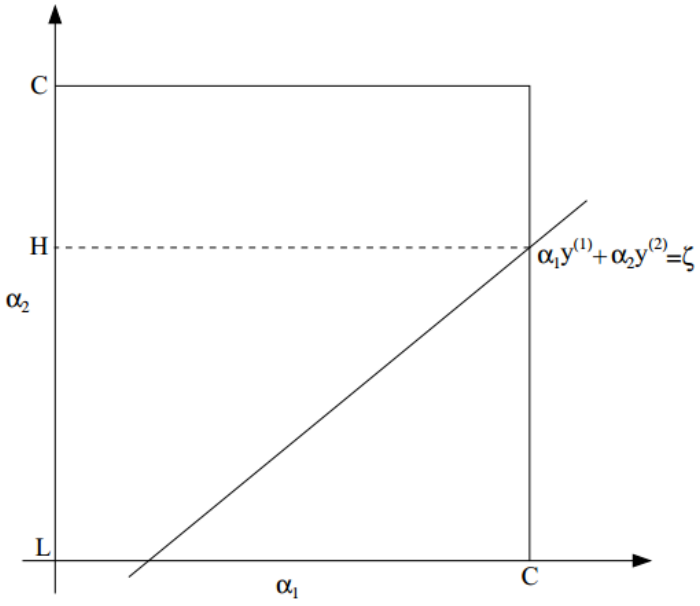

FIGURE 2.2: Illustrating the constraints for two αs

Figure 2.2 illustrates the effect of the constraint on the possible values of α_is. The $0 \leq \alpha_i \leq C$ constraint makes sure they lie in the square and the $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ linear equality constraint puts them on the line. So in every iteration the algorithm finds the optimum of the objective function on the specific line segment, which can be done analytically as follows:

Let us say that that we have a solution that satisfies 2.61-2.62. Then for some ζ constant the following equation holds:

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta \tag{2.63}$$

$$\alpha_1 + \alpha_2 y^{(2)} y^{(1)} = \zeta y^{(1)}$$

$$\alpha_1 = y^{(1)}(\zeta - \alpha_2 y^{(2)}). \tag{2.64}$$

Holding the other α-s constant an substituting 2.64 instead of α_1 int W we get a quadratic optimization problem in the form of:

$$\max \quad W'(\alpha_2) = a\alpha_2^2 + b\alpha_2 + c^3 \tag{2.65}$$

$$L \leq \alpha_2 \leq H. \tag{2.66}$$

where $a < 0$.

Since W' is a parabola it has only one maximum point which can be calculated analytically by taking the derivative. $\alpha_{2opt} = \frac{-b}{2a}$. Since this solution does not take the constrain into account we need to modify it accordingly:

$$\alpha_2 new = \begin{cases} H & \text{if} \quad \alpha_{2opt} \geq H \\ \alpha_{2opt} & \text{if} \quad H > \alpha_{2opt} > L \\ L & \text{if} \quad L \geq \alpha_{2opt} \end{cases}. \tag{2.67}$$

2.5 Random Forests

Recently there is a great interest in ensemble learning methods, which combine the results of different models into one classification result. This idea makes very much sense if we can ensure that the different models are somehow independent from each other. A popular technique is called bagging, when different models are built on bootstrap samples of the training set and then a majority vote is taken at the end. The main concept of Random Forests, as introduced by Leo Breiman in 1996, is that we add another layer of randomization on top of bagging decision trees, during the tree construction itself. But before diving into the Random Forest algorithm let us first introduce the basics of decision trees.

2.5.1 Decision Trees

Decision trees classify new instances by running them down a tree from the root to some leaf node which in the end contains the predicted class label. At each non-leaf node a simple rule is applied to the instance to decide which child node to select next. This decision rule is based on the value of one of the input attributes. Each child node represents a subset of the values of the decision attribute at the given node. If a leaf is reached the instance is classified as the

class label corresponding to that leaf-node. The purpose of different decision tree building algorithms is to find the appropriate splitting attributes and splitting values at every node which maximizes the predictive ability of the model. Now let us examine the ID3 algorithm which is a popularly used tree building technique.

The main concepts of the algorithm [17]

- A non-leaf node in the tree represents a "cutting" attribute and the edges represent disjoint subsets of the attribute values for the node-attribute. A leaf node corresponds to the predicted label when the input is described by the path from the root to that leaf node.
- In order to reduce model complexity, each non-leaf attribute should correspond to the input feature that contains the most information regarding the class label given the path leading to that node.
- Entropy is used to determine the informativeness of a given attribute and cutting-attributes are selected based on this measure.

In order to describe the algorithm we first need to define entropy and information gain.

For a node with a class label probability distribution $P = (p_1, p_2, \ldots, p_n)$, the entropy $H(P)$ is defined as follows:

$$H(P) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.68}$$

The class label distribution P can be estimated from the training data set, where the estimate of p_i equals the proportion of the training instances reaching the node with label i. From now on let $H(T)$ denote the entropy of the members of set T. For a split $(T_1, T_2, \ldots, T_n)$ we can write the information needed to classify the instance as:

$$H(X, T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} H(T_i) \tag{2.69}$$

which is basically the expected value of the entropies of the subsets generated by the split.

At each node we want to select the attribute A that adds the maximal amount of new information about the class label, a concept which is called Information Gain:

$$IG(A, T) = H(T) - H(X, T). \tag{2.70}$$

The ID3 algorithm will use information gain at every node to select the best splitting attribute for the training set.

The ID3 algorithm

1. If all records of T have the same label L, then return a single node with L
2. If all records have the same value for all the input attributes, then return a single node with the most frequent label in T
3. Compute the information gain for each input attribute and let A denote the attribute for which $IG(A, T)$ this is the biggest:
   return a tree where the root is labeled with A and the algorithm is recursively called for the child nodes represented by the subsets of T partitioned by the values of A

While this specific algorithm uses Entropy to select attribute splits, there are other possibilities, for example Gini-index.

2.5.2 The Random Forest algorithm

Now that we know how to construct decision trees, the Random Forest is very simple yet incredibly effective.

The algorithm[18]

1. Take a bootstrap sample T_B from the training set
2. Build a decision tree using T_B where instead of using the best splitting attribute, use the best splitting attribute of a random subset of the attributes.
3. Repeat 1. and 2. to build many random trees
4. Classify a new instance by running it down on all the random trees then take simple majority vote

Most machine learning libraries give you the opportunity to fine-tune the way each decision tree is built in the random forest. I will go into further detail when describing the empirical results.

Chapter 3

The Dataset

3.1 Dataset

In this section I will describe my choice of stock index to model, and also the sources I used to download daily stock market data.

3.1.1 The NASDAQ Composite Index

I chose to model the NASDAQ Composite Index, which is a stock market index representing the common stocks and securities listed in the NASDAQ stock market.

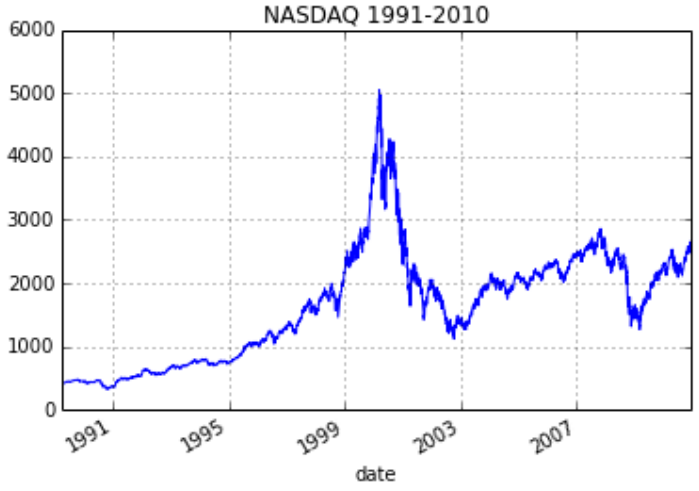

FIGURE 3.1: NASDAQ Close Price

The NASDAQ Stock Market(National Association of Securities Dealers Automated Quotations) is an American stock exchange and is the second-largest stock

exchange in the world by market capitalization. The NASDAQ Composite Index has more than 3000 components representing the performance of technology and growth companies.

Modelling a stock index instead of a single stock has many advantages. First of all since it is a weighted average of the constituting papers, it is much less volatile. Also the amount of stocks traded behind the index (volume) is larger by orders of magnitude making it possible to detect the possible patterns. While indices cannot be traded, there are many exchange traded funds on the market which produce derivatives tracking closely the different market indices.

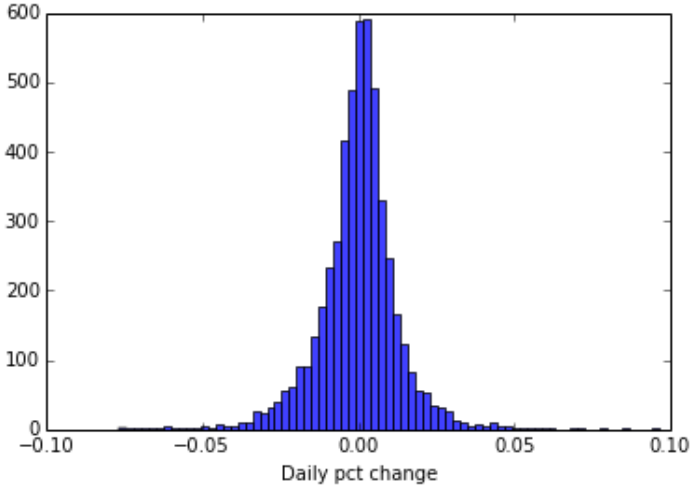

FIGURE 3.2: Daily % change histogram

3.1.2 Data sources

I used two sources to download daily data. For stock prices I used Quandl, which is a free time-series data repository available on the internet. The Python api provided by them makes it easy to automate the data downloading process.

The other data source I used to download time-series regarding the US economy is the free api service provided by the St. Louis Federal Reserve Bank (FRED).

3.2 Features

3.2.1 Technical indicators

Technical indicators are numerical features calculated from historical price and market data. Most of the technical indicators used here have a usage history

going back decades. While no one says that one indicator is universally useful, many technical analysts think that there are trends in the price movements, and from time-to-time some technical indicators can pick up these trends.

While these technical indicators are just numbers calculated with an exact formula, the nature of technical stock analysis still remains in a form of an art.

The following table presents some of the most widely used technical indicator:

Indicator Name	Formula
EMA(N)	$EMA_t(N) = \frac{1}{N+1}\sum_{i=0}^{\infty} p_{(t-i)}(1-\frac{1}{N+1})^i$
MACD(N,M)	$MACD_t(N,M) = EMA_t(N) - EMA_t(M)$
K%	$\%K(N) = \frac{100(c_t - \max_{i=0..N}(p_t))}{(\max_{i=0..N}(p_t) - \min_{i=0..N}(p_t))}$
Arms Index	$\frac{NR_{Advances} \times VOL_{down}}{NR_{Declines} \times VOL_{up}}$

For prediction 70 different technical indicators were computed based on a detailed list provided by StockCharts.com [19].

3.2.2 Economic variables

Since the NASDAQ index is a composite index containing thousands of individual stocks across the USA, it seems reasonable to add economic variables to the dataset. Improving economic conditions in the US will affect companies across the country thus it will improve the stock market performance. In order to measure the performance of the US economy I have included several daily economic variables downloaded from FRED.

The following variables have been included:

- Oil price
- Willshire Market index (similar to other stock indices)
- US Dollar index (weighted avarage of the $ exchange rate against a basket of currencies)
- Corporate Bond index
- 1-year Treasury bond rate

- S&P500 index
- Economic Policy Uncertainty Index

In order to get the change in the economic conditions for every variable I took the daily percentage change in the analysis instead of using the absolute levels.

3.2.3 Feature extraction based on lagged correlations

The goal is to find technical indicators that are possibly correlated with the daily price movements. Although this correlation is not going to be significantly greater than zero we can try to compute the lagged correlations of the input features with the movement and select the highest ones. I have plotted the lagged correlations for the variables with the daily price change and selected those that seemingly had some pattern of correlation. Although this is a heuristic method and will most likely not imporve predictive performance, but using the correct machine learning method we can be sure that it will not decrease it either.

Chapter 4

Development environment

4.1 Development environment

4.1.1 Python

I used the Pyhton programming language for all parts of my thesis, including data acquisition, feature generation, algorithm development and testing and data visualization. The motivation behind this choice is very clear, because Python enables a very fast and dynamic way of development with a full variety of the necessary data mining and visualization libraries suitably for out-of-the-box use. In the next section I will give a short introduction to the Python libraries I used during development.

4.1.2 Python libraries

The four main libraries I used are NumPy, Pandas, Matplotlib and Scikit-learn.

Numpy provides fast array based operations comparable to Matlab-s functionality, which was necessary to scale up the algorithms to bigger amounts of data.

Pandas is a data analysis library which provides many useful tools for data cleaning and munging including the DataFrame object type which seamlessly integrates with Numpy arrays allowing a column labeled array type suitable for storing variables.

Matplotlib is a visualization library able to produce a wide variety of graphical content including graphs, bar charts and the like.

Scikit-learn was the main reason for choosing Python. It is a machine learning and data analysis library which contains all of the widely used classification, validation and other data mining algorithms making it easy to build or expand upon them. The algorithms in scikit-learn are based on NumPy and SciPy (scientific python library) and are well optimized allowing for parallel computations.

Chapter 5

Model selection and testing

5.1 Classification model and performance measures

In this section I will describe the classification models used for stock market prediction. I will also describe the different performance metrics used for model selection.

5.1.1 Model

I am modelling the market movements as a binary classification problem. The target variable is the direction of the the price movement from the daily opening price to the daily close price:

$$y^{(i)} = sign(P^{(i)}_{close} - P^{(i)}_{open}).$$

Days with 0 change will be discarded from the training set during noise filtering which will be described later.

The input features are the technical and economic indicators described in the second chapter. This results in an 88 dimensional input vector.

5.1.2 Performance measures

Besides the general classification performance metrics (e.g. confusion matrix) it seems reasonable to provide performance measures that take time into account for this is basically a time-series prediction problem. Most of the basic metrics

like accuracy or precision can be easily extended into sliding window versions which will compute the performance for the last N-days. This technique makes it possible for us to track the and compare the different models' performance during different time periods.

For each model five performance metrics are presented to make them easily comparable:

- **Accuracy** shows the total percentage of correctly predicted days on the given example set
- **% classified up** shows the total percentage of days classified up
- **Trade accuracy** shows the percentage of the total price movement correctly predicted based on the direction prediction
- **Avg. yearly return** is the mean return on our investment using the basic trading strategy
- **Std. of yearly return** is the standard deviance of the yearly return on our investment using the basic trading strategy

It is important to note that the mean yearly return is not directly computable from the mean trade accuracy for the accumulative nature of the return. I chose to include financial performance yearly return and trade accuracy in order draw a better picture from the traders' point of view. While I could have included many more performance measures (e.g. ROC curves) I did not find it necessary to do so.

5.2 Trading strategy

For algorithm evaluation I will use the same trading strategy for all the methods. The strategy will be very simple and is not intended to reflect a real life trading strategy where risk and money management are very important aspects. At every day I we buy or short-sell stocks on the market opening price. Our decision to buy or sell depends only on today's prediction so if we predict that the price will increase we buy on opening price and sell on close price, and if we predict that the price is going to fall, we short sell on opening price and re-buy on closing price. We will not hold stocks overnight. Also we are fully invested every day, so for every day our profits or losses will be equal to the price percentage

change during that day, depending on our predictions. I will also assume that there are no transaction costs and also it is free to enter into long(buy) and short(sell) positions. While these restriction may seem to decrease the relevance of the performance evaluation, in my opinion they actually do the opposite. The assumptions one makes about trading strategy and transaction cost can distort the model performance measures, so by taking the one of the simplest trading strategies we can be sure that we measure the quality of the prediction algorithm, not the trading strategy.

5.3 Benchmark results

In order to be able to asses the quality of the predictors we need to take some benchmark results into account. This is especially important because we want to be able to distinguish good model performance from good market performance. For example in good economic conditions when the stock market shows an increasing trend there will be more upward price movement than downward. In this case always predicting ↑ will give us good model performance for that period even this model will have no predictive ability outside this time interval.

5.3.1 Buy-only

In this strategy we assume that the market is performing well and we think that the price is more likely to go up each day. In this case our trading strategy is to buy stocks on the open price and sell them on the close price. It is clear that the performance in this case reflects the number of up days compared to down days.

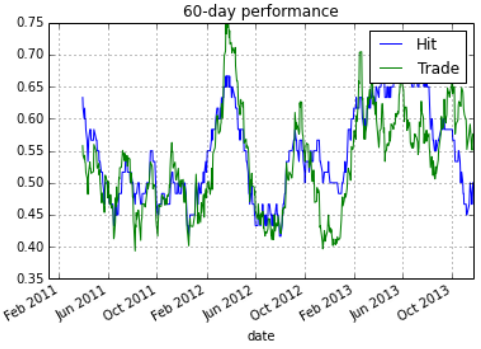

FIGURE 5.1: Buy-Only performance

Looking at the performance plots we can see that market has performed well in the last two years and by using this simple strategy I described, we would have gained significant revenue.

Unfortunately using this strategy for the whole time-frame would have generated different results.

5.4 Model training and Validation

5.4.1 Parameter selection and validation

For parameter selection I used simple grid search to avoid getting stuck in local optimum. Although this method is computationally very expensive, the size of the dataset and parameter space still makes it viable in our case. The parallel design of the learning and validation algorithms allowed me to test a large set of model parameters.

For model validation I used a modified cross-validation algorithm which I tailored specifically for this problem. The basic validation algorithm behind mine is a sliding window cross-validation where the model is trained on N consecutive days of training data and tested against the next M days following the training set. The training window then slides forward providing a new training and testing set. It is possible to use accumulative training, where the only the right boundary of the training set slides, using all of the previous days for training.

The standard version of these algorithms uses the prediction accuracy for parameter selection but I modified it to allow the selection based on trading accuracy as described earlier. This method assigns more weight to days with large price movements, even though we are using only binary classification.

5.4.2 Noise filtering

The main idea behind technical analysis is that price movements, especially larger ones can be somewhat predicted by historical price data based various trends in human decision making. It seems intuitive to expect the direction of larger price movements is more predictable from past data, while very small movements may have very little signs in past data. To incorporate this idea into our model, it seems intuitive to set a threshold for minimal price change and drop the data not reaching the threshold (where the two classes highly overlap). By increasing the minimal price change threshold we can observe an increase in the distance between the center (mean) of the two set of points. Similarly, there with increased there is a decrease in cosine similarity between the class centers.

In order to increase predictive ability we need to find a good threshold that gives good separation while retaining much of the input data. I have set this threshold arbitrarily to 0.003 which meant that about 25% of the training dataset was labeled as noise during training.

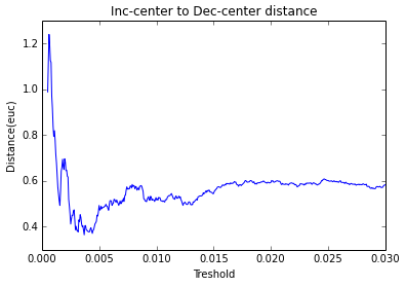

FIGURE 5.2: Inc-Dec Center Euclidean distance

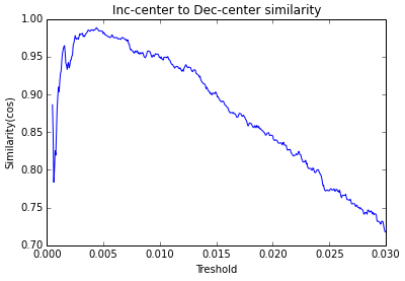

FIGURE 5.3: Inc-Dec Center Cosine similarity

5.5 Results

Tables 5.2 and 5.3 shows the model performances for each algorithm for the best parameters selected during model selection.

Every model is tested on 3 subsets of the dataset. The *Training* column shows the model performances measured by cross-validation on the training set. This result was used for model selection. The *Test* columns shows the performance measured by training the models on the full training set and evaluating it on the test-set (2011-2014). Column *Test-S* shows the results of *simulated testing* which shows the model performance on the test-set when the model are re-fitted after every 100 days with the new data.

TABLE 5.1: Benchmark results

	Buy-only		**Random prediction**	
	Train	**Test**	**Train**	**Test**
Accuracy	52.83%	54.96%	49.92%	51.60%
% classified up	100.00%	100.00%	50.64%	51.46%
Trade accuracy	47.70%	52.35%	50.12%	51.98%
Avg. yearly return	-2.17%	4.05%	-0.95%	6.31%
Std. of yearly return	19.68%	7.97%	16.07%	10.43%

Looking at 5.1 we can see the described metrics for the benchmark models. As predictable the Random-prediction benchmarks gives accuracy close to 50% with similar trade accuracy and avg. return close to 0% [1]. As described Buy-only performance depends heavily on economic conditions. On the Test set, where an upward trend can be observed we can see that we get an 52.35% accuracy with 4.05% avg. yearly return. While the avg. return is positive this still cannot be considered a good investment strategy considering the std. of the return which would call for extreme risk management measures.

[1] These results can differ substantially by a new set of random predictions.

TABLE 5.2: Classification results: k-NN and Logistic Regression

	k-NN			**L1-Logistic**			**L2-Logistic**		
	Train	**Test**	**Test-S**	**Train**	**Test**	**Test-S**	**Train**	**Test**	**Test-S**
Accuracy	53.58%	54.37%	54.81%	54.05%	53.79%	54.66%	54.84%	53.50%	55.39%
% classified up	57.26%	71.14%	77.41%	51.39%	69.39%	74.93%	54.64%	74.93%	75.66%
Trade accuracy	54.64%	51.84%	53.36%	53.93%	53.59%	53.86%	56.10%	52.35%	54.60%
Avg. yearly return	21.55%	2.28%	9.35%	23.37%	13.48%	13.95%	32.33%	9.02%	14.61%
Std. of yearly return	37.58%	6.59%	8.67%	30.03%	9.31%	7.13%	25.17%	9.00%	8.84%

TABLE 5.3: Classification results: Random Forest and SVM

	Linear-SVM(L1)			**Random Forests**			**RBF-SVM**		
	Train	**Test**	**Test-S**	**Train**	**Test**	**Test-S**	**Train**	**Test**	**Test-S**
Accuracy	55.11%	56.41%	55.25%	53.04%	51.60%	50.73%	55.22%	55.25%	55.10%
% classified up	53.41%	72.30%	82.22%	49.79%	57.00%	62.54%	52.64%	84.26%	82.36%
Trade accuracy	56.83%	56.29%	53.80%	53.62%	50.80%	49.50%	57.51%	52.47%	53.75%
Avg. yearly return	38.17%	21.39%	11.78%	18.49%	0.01%	1.87%	39.31%	9.48%	10.86%
Std. of yearly return	35.67%	12.49%	7.71%	27.29%	7.03%	11.08%	33.46%	11.82%	5.92%

5.5.1 k-NN

By using grid search with sliding window cross-validation to find the optimal value of k the following model parameters were selected:

k: 7, weighted by distance

We can see that the model was able to reach 54.64% trade accuracy on the training set which corresponded to 21.55% yearly return on average. However the standard deviation of the return is 37.58% which is highly undesirable.

However the simulated performance on the unseen dataset shows only 54.36% trade accuracy which pushes the average return down to 9.35% with a std. of 8.67%

5.5.2 Logistic Regression

For logistic regression I trained both L1 and L2 regularized versions. As we can see from the results the L2 regularized version performed much better on the training data while it was only slightly better on the test set.

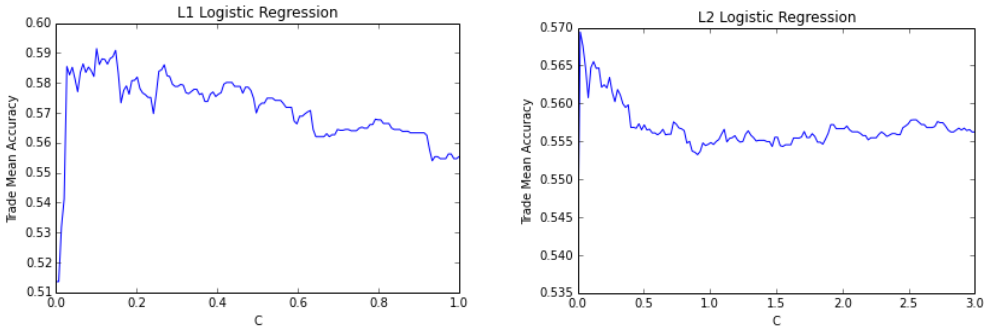

FIGURE 5.4: Parameter selection for Logistic Regression

The figures on 5.4 show us the mean trade accuracies with sliding window cross validation for the different values of C. The parameter selected for the models are the ones that maximize this measure. For L1 this is 0.10 and for L2 this C is 0.02.

The results show that simulated on the training set, both L1 and L2 logistic regression outperforms the k-NN both in return and variability even though the L1 regularized version shows a slightly worse average trade accuracy.

On the simulated test the difference between k-NN and logistic regression gets more accentuated. In contrast with k-NN, trading based on L2 logistic regression produced 14.51% average return which is approximately 5% greater with

technically the same standard deviation. L1 logistic regression also produced similar results with even less return variability.

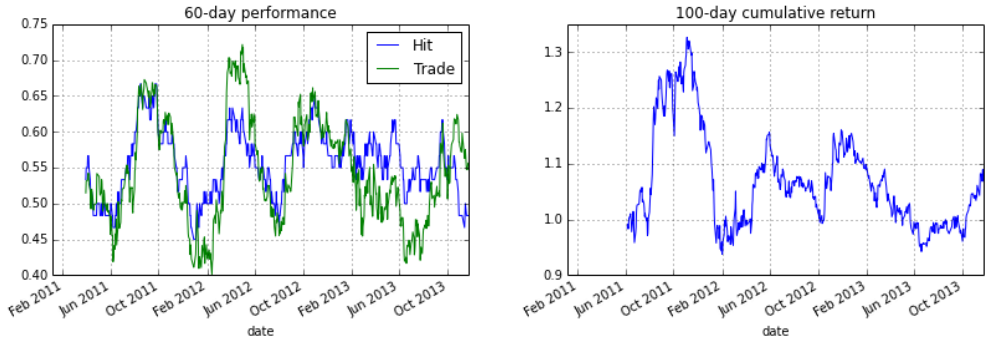

FIGURE 5.5: L2 logistic regression performance

5.5.3 Support Vector Machine

For support vector machines I tried both linear and kernel versions. I have used the widely popular RBF kernel described in the second chapter. To select the appropriate C and gamma parameters i used grid search over the parameter space where C and gamma were both added in an exponentially increasing magnitude following the advice of a guide made by the National Taiwan University [20]. The result of the grid search can be easily visualized with a heat map on 5.6.

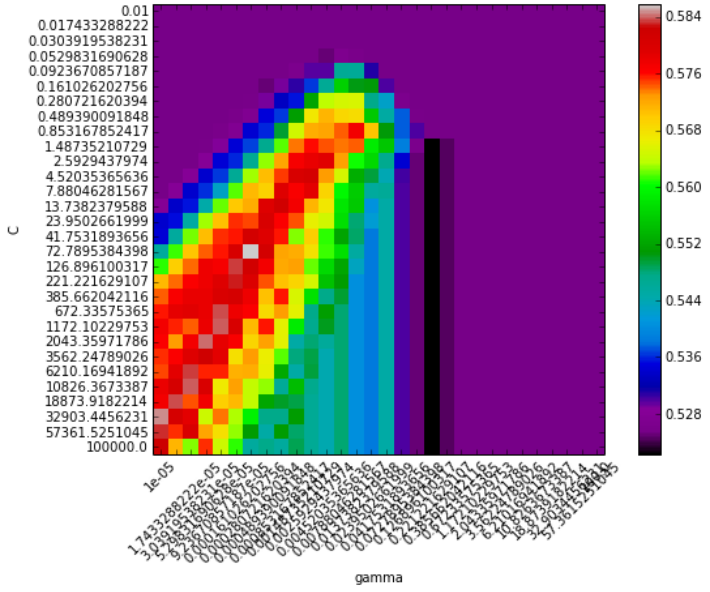

FIGURE 5.6: RBF grid search

We can see a clear pattern regarding the best trade accuracies on the picture; they lie on an approximately straight line on this exponentially scaled picture. Although it would probably be interesting to extend the grid for lower gamma

and C values to see the full pattern, it becomes increasingly expensive to solve the optimization problem. As we can see the best parameter combination selected by cross-validation is $C = 72.78954$ and $gamma = 0.00028$, giving 57.51% trade accuracy and 39.31% average return simulated on the training set, which is better than any of the previous models. As for the linear SVM, the best C penalty parameter of 0.08172 resulted in a trading accuracy of 56.83% on the training set.

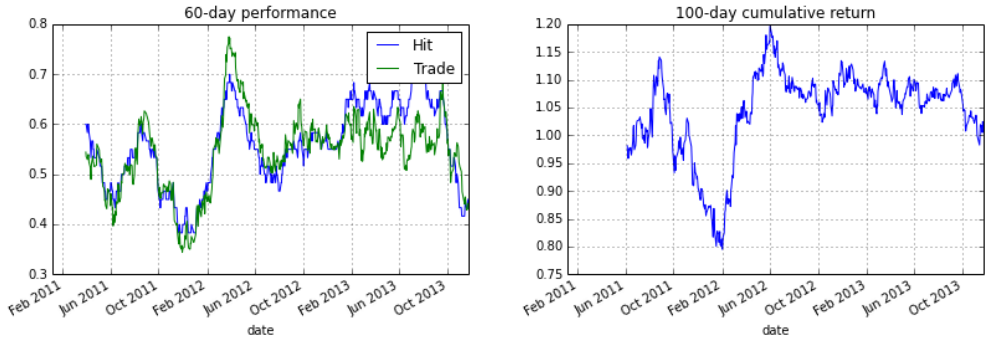

FIGURE 5.7: RBF-SVM performance

Even though both the linear and RBF SVM showed superior performance during model selection the picture slightly changes when we try them on the previously unseen data. The most promising RBF kernel showed 53.75% trade accuracy which is similar to k-NN and L1 logistic regression but is worse than L2 logistic. However the standard deviation for the simulated test result was better than any of the previous models showing a more consistent performance.

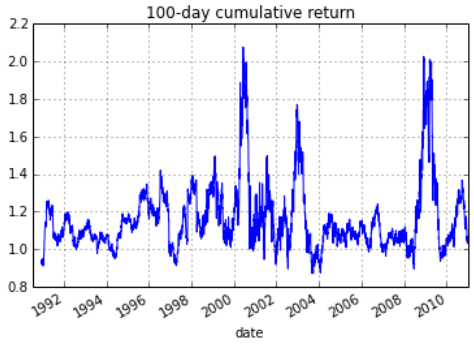

FIGURE 5.8: SVM

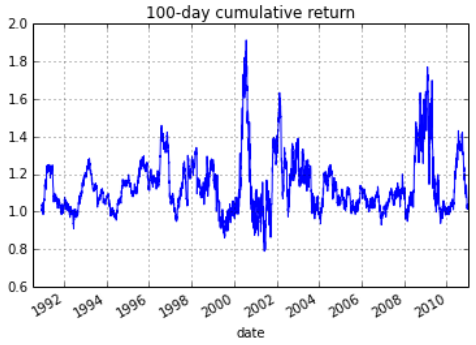

FIGURE 5.9: Logistic regression

The great difference between the results on the training and test set is interesting, because I payed special attention to avoid overfitting with sliding window cross-validation, however the large parameter set can still give space for possible overfitting. Another explanation could be that it was simply a bad period for the SVM (it performed very poorly from the beginning to mid 2012). If we compare the 100-day returns on the training set simulation we can see that it almost

always outperformed the L2 logistic regression however there are times when logistic regression was simply better. These two possible explanations highlight the difficulty in picking the best predictive models in difficult forecasting problems.

5.5.4 Random Forest

Random Forest classifiers can also be trained using different parameters which control the way how individual trees are built thus control overfitting and predictive ability. I used grid search to select the appropriate value for the following parameters (find the selected parameter in parenthesis):

number of trees (160), *splitting criterion* (entropy), *number of features considered for each split* ($\sqrt{\text{nr of features}}$), *maximum tree depth* (no limit), *minimum samples per leaf* (1)

We can see that Random Forests performed very poorly compared to the previously presented algorithms. Even on the training set it was outperformed by all the models including k-NN. On the test set we can see that it achieved less then 50% accuracy which indicates that the predictions were no better than random guesses. It is interesting to see that this very popular off-the-shelf technique had much worse technique than the other models. I could not come up with any explanation regarding this.

5.6 Summary of Results

We could see that most of the machine learning algorithms presented in Chapter 2 could outperform both the buy-only and the random strategy. While the results varied greatly, it clearly shows that there is room for machine learning and technical analysis in stock market analysis.

The best performing model (L2-Logistic regression) on our test set could achieve a 54.6% trading accuracy which I considered very good given the nature of the problem. The average yearly return on the test set ranged 10 and 15 % for the best models, but with high standard deviance. In absence of transaction costs during my evaluation it is hard to compare these values with actual returns but even so it looks promising. We could also see that the difference in performance among the models changes by time, and it is hard to pick a single model for prediction.

Chapter 6

Conclusion

In this thesis I intended to give an overview of the viability of technical analysis and machine learning in stock market forecasting. After introducing the basics of the stock market, I reviewed several popular machine learning algorithms and tested them by trying to predict the direction of the NASDAQ composite index. While my results are far from a real-world trading they still indicate a significant possibility for the application of machine learning in stock prediction.

We could see that myriad of machine learning algorithms can provide great new tools for financial forecasting. Even though there are theories to contradict the possibility of stock market predictions, we know that there is a huge field which tries to go into the opposite direction. I am certain that with the growth of the available data and the increasing sophistication of the learning models, we will see many success stories even if their life-span will be inevitably short.

Future work

While trying to predict daily price movement is a challenging problem, it has much less real-world applicability then prediction for very short periods of time, also known high-frequency prediction and trading. Predicting high frequency price movements poses a whole different problem where model training time has a very significant impact. It seems clear that technical analysis combined with machine learning and distributed processing frameworks can provide valuable tools for high frequency trading systems which would give a very interesting research topic.

Bibliography

[1] Stock market, March 2014. http://www.investopedia.com/terms/s/stockmarket.asp.

[2] Nyse quick facts, March 2014. http://www.nyx.com/who-we-are/quick-facts.

[3] Basics of stock exchange, March 2014. http://www.ctrustnetwork.com/c427p1/business-and-investment/stock-market/stocks-101/basics-of-stock-exchange.html.

[4] Nasdaq vs. nyse, March 2014. http://www.diffen.com/difference/NASDAQ_vs_NYSE.

[5] Christopher Clifton. Data mining - encyclopedia britannica, March 2014. http://www.britannica.com/EBchecked/topic/1056150/data-mining.

[6] John J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications (New York Institute of Finance)*. New York Institute of Finance, 1999. ISBN 0735200661.

[7] Andrew W. Lo. Efficient markets hypothesis. *THE NEW PALGRAVE: A DICTIONARY OF ECONOMICS*, 2007.

[8] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. ISBN 0321321367.

[9] Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang. *Principles and Theory for Data Mining and Machine Learning (Springer Series in Statistics)*. Springer, 2009. ISBN 0387981349.

[10] Andrew Ng. Generative learning algorithms. http://see.stanford.edu/materials/aimlcs229/cs229-notes2.pdf, 2008.

[11] Su in Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient l1 regularized logistic regression. In *In AAAI*, 2006.

[12] Andrew Ng. Support vector machines. http://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf, 2008.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787.

[14] Prof. David Sontag. Kernel methods and optimization. http://cs.nyu.edu/dsontag/courses/ml12/slides/lecture7_notes.pdf, 2012.

[15] Rbf svm parameters, March 2014. http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

[16] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[17] David McG. Squire. The id3 decision tree algorithm. http://www.csse.monash.edu.au/courseware/cse5230/2004/assets/decisiontreesTute.pdf, 2004.

[18] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[19] Technical indicators and overlays, March 2014. http://stockcharts.com/help/doku.php?id=chart_school:technical_indicators.

[20] Chih-Jen Lin Chih-Wei, Chih-Chung Chang. A practical guide to support vector classification. http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf, 2010.