University of Sheffield
Department of Computer Science

Predicting Financial Markets using Text on the Web

Zsolt Bitvai

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science of the University of Sheffield
May 2016

Abstract

It remains a question whether consistent inefficiencies can be discovered in financial markets in order to realise risk free excess profits above the market rate of return over long periods of time. Previous research in finance claims that markets are fundamentally unpredictable. Therefore, theories such as the Efficient Market Hypothesis, Capital Asset Pricing Model and Modern Portfolio Theory aim to match the market return for optimal asset allocation. However, Behavioural Economics provides competing psychological explanations for seemingly irrational behaviour of market participants. This suggests that risk free excess profits are possible to achieve by exploiting these phenomena. Another limiting factor in this field is that the public availability of previous studies is restricted because once new inefficiencies are discovered, they are often arbitraged away, and may not be published to preserve competitive advantage. Therefore, there is a clear need for a comprehensive study to highlight alleged inefficiencies in markets and determine their sources.

In this thesis, we aim to empirically document market inefficiencies, by proposing several novel Machine Learning and Natural Language Processing algorithms to intelligently forecast the evolution of markets. This data driven approach is in contrast to more theoretical frameworks found in traditional finance. Each of our models captures different characteristics of markets that are uncovered by constructing new modelling techniques suitable for the task. These tasks range from measuring the influence of related securities across time periods, to quantifying uncertainty in prices and underlying market dynamics, and to even simulating a real life human trader that aims to maximise their balance by assuming positions based on what economic and news information they have read. Our models aim to preserve generalizability across a multitude of markets by carefully trading off modelling complexity with simplicity and computational capacity. They treat market dynamics as arbitrarily complex systems and are able to explain key influential factors affecting market evolution. Our multi modal data sources include time, structured market and economic data as well as unstructured text related to news articles and social media posts. Our models are able to execute trades while adapting to the particular aspects of each market domain they operate on. We also explicitly model temporal dynamics via non-stationary processes, which is challenging due to the high levels of noise observed in markets. We further propose multiple ways to capture trading profit as the model objective directly, instead of employing simplified surrogate objectives.

In the first part of the thesis, we detail a novel linear model where we directly optimize for trading profit in a realistic stock market trading scenario, and model different companies and trading periods with multi-task learning regularizers. In addition, we empirically validate technical analysis, which relies on human constructed heuristics for trading. Next, we outline several non-linear probabilistic models within the state-of-the-art Gaussian Process framework, where first we incorporate the above profit objective with a Gaussian approximation. Second, we show how we can model multiple types of input data with a novel kernel combination technique. Finally, we propose a way to exploit the posterior uncertainty of the model output in order to capture significant arbitrage opportunities. In the final part of the thesis, we describe several new deep learning model architectures inspired by recent advancements in representation learning. We demonstrate that obtaining progressively higher levels of representations for concepts in a text document considerably boosts predictive power, as compared to shallow architectures. Then, we show a method to identify influential phrases within a text document, which considerably aids the interpretability of the black box nature of Neural Networks. In the final experiment, we propose an end-to-end fully differentiable learning framework to simulate a real life human trader that reads raw market, economic, and news data every day and places limit order trades directly on the market, while optimising for the net worth of their own portfolio.

The results of the experiments are demonstrated on a variety of developed markets in the United Kingdom and the United States over several decades of test data. These include three major stock markets as well as an emerging peer-to-peer lending market. Last, we demonstrate a way to forecast box office revenues from critic reviews for prediction markets. The contributions of this thesis is to show that several financial markets are predictable and risk free excess profits are possible to achieve via automated trading over long stretches of time. We also show that temporal, structured and textual characteristics of the data have high influence on predictability. Thereby, this work provides evidence against the Efficient Market Hypothesis.

Acknowledgements

I would like to express sincere gratitude to my supervisor Dr. Trevor Cohn, who helped me along the way of learning about cutting edge machine learning and natural language processing algorithms, and provided critical guidance when obstacles were hit. He also invited me to spend more than a year at Melbourne, which I found very beneficial. In addition, I am immensely grateful to Prof. Neil Lawrence for providing high quality tutorials about Gaussian Process in the GP summer schools. Further, I would like to thank Kashif Shah, Michal Lukasik, P.K Srijith and Jiazhen He for the illuminating discussions that greatly aided me in better understanding this research field. Last, I would like to acknowledge the support of my partner Vaewy, who was there keep my morale up.

‘A blindfolded chimpanzee throwing darts at the Wall Street Journal can select a portfolio that performs as well as those managed by the experts.’

Burtin G. Malkiel, A Random Walk Down Wall Street

‘“Don’t blame you,” said Marvin and counted five hundred and ninety-seven thousand million sheep before falling asleep again a second later.’

Douglas Adams, The Hitchhiker’s Guide to the Galaxy

‘If you don’t want a generation of robots, fund the arts!’

Cath Crowley, Graffiti Moon

‘In this world nothing can be said to be certain, except death and taxes.’

Benjamin Franklin

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context of the Study

Financial markets affect our lives in profound ways. They capture the information content available in society at large and encode our expectations for the future. What value people give to something is answered by the markets. These systems are emergent phenomena (Tesfatsion, 2002) that are based on the collective action of individual members of society. They allow people to trade goods with each other, such that the overall welfare of society increases. They fill the vital role of finding optimum ways to allocate resources and make society function better with the help of the "invisible hand" of the markets (Smith and Garnier, 1838).

With the advent of the digital revolution, people have managed to automate many mundane and complex tasks, thereby significantly cutting down on costs of production while at the same time achieving superior performance. Automation on such a grand scale has transformed society. Even though software has been a key component of this break through, artificial intelligence has enabled the automation of the writing of software itself by extracting patterns from huge amounts of data generated every day. These smart systems are capable of learning from their own mistakes, and hence have outperformed human planning and reasoning on many tasks where precisely weighted decisions need to be made by carefully evaluating many different factors.

Markets are regarded as fairly complex systems, which are affected by countless number of factors, and which encode the dynamics of human interactions and the environment. Because of this large number of interactions, it has been a challenging task to make sense of their inner workings. Hence, they have been often described as completely random or chaotic systems with large amounts of noise (Campbell et al., 1997). However, people have been trying to understand markets for a long time, and with the advent of artificial intelligence and sophisticated data analysis, more patterns can be discovered to forecast the evolution of markets than ever before.

The digital revolution has also brought about the interconnected system of the internet, which has revolutionized access to financial markets, many of which have turned into completely electronic systems. With low barriers to entry and access to unprecedented amount of information available online, anyone can participate in markets and become a professional trader. These circumstances have allowed individuals to construct their own strategies for trading, with the aim of earning a trading profit by beating the average market rate of return and while avoiding taking on excess risk in their dealings.

By following their own sophisticated rules for investing, these entities increase the overall efficiency of the financial eco-system, and therefore they provide an essential service to society. First, they give an accurate assessment for the value of goods in the form of trading price discovery, and second, they create liquidity for carrying out trading transactions in a seamless manner (Beddington et al., 2012). However, in order to provide this service, it is critical that these traders have a good internal model of market evolution which they can incorporate in their own trading strategies (O'Hara, 2003). In the next section, we investigate whether it is possible to construct such a model.

1.2 Statement of the Problem

In the literature, it has been an open question whether it is possible for individuals to consistently earn risk free excess trading profits above what is the market rate of return on an investment, and therefore be able to accurately forecast financial markets over long periods of time. However, with large amounts of data and recent breakthroughs in sophisticated computational methods that turn this data into actionable knowledge, measuring the dynamics and predictability of markets have become a more manageable aim.

On the one side of the argument, proponents of the Efficient Market Hypothesis (Malkiel, 2003) state that financial markets are unpredictable, because the prices follow a random walk. They claim that markets are chaotic systems with very high levels of noise. Therefore, it should not be possible to consistently outperform the market rate of return and achieve excess trading profits without excess risk.

On the other side of the argument, proponents of Behavioural Economics state that financial markets are predictable. because people often make emotional decisions which are not always rational (Kahneman and Tversky, 1979). These decisions lead them to exhibit predictable behaviours such as herding, imitation, aversion to losses, and overconfidence in gains. Therefore, scholars in this group claim that inefficiencies in markets exist in the short term, which can lead to the discovery of these patterns.

In an attempt to forecast markets, people in the finance community have employed technical and fundamental analysis. With technical analysis they used historical market price and volume data to perform the modelling, while with fundamental analysis, they constructed indicators to describe the properties of financial securities so as to predict their intrinsic value. Last, there have been sporadic attempts at applying text analysis to various forms of text, ranging from newspapers to online content generated by the crowds at large, in order to discover a signal for market dynamics. Recently, there has been an increased interest in attempting to forecast markets using sophisticated computational modelling frameworks coupled with fundamental and technical analysis (Lo et al., 2000; Abarbanell and Bushee, 1997), along with a newfound focus on text analysis (Feldman, 2013). However, the results were mixed and often counterarguments could be found to explain an observed behaviour. Therefore, it is not evident whether fundamental, technical or text analysis work, and in the finance literature the validity of these techniques remains disputed.

That issue is corroborated by the fact the markets are dynamic systems where existing signals may abruptly vanish due to their discovery and subsequent arbitrage by market participants. As a result, this phenomenon complicates research in this area where positive findings tend to be severely under-reported to preserve competitive advantage. Therefore, there is a clear need for a comprehensive and transparent research study to shed light on whether consistent risk free excess profits can be achieved on markets over relatively long periods of time.

1.3 Aim and Scope

The main goal of this study is to show a way to consistently earn excess profits without excess risks by trading assets on several financial markets over relatively long periods of time.

In this research, we restrict the modelling frameworks to state-of-the-art machine learning which includes linear regularised models, as well as non-linear models such as Gaussian Processes and Artificial Neural Networks. We compare our results against other known methods of machine learning and pattern mining. However, many of our proposed models extend upon these broadly defined learning frameworks mentioned above, taking them into novel directions.

Our datasets include several decades of daily stock market data from major British and American stock exchanges, comprising of almost a thousand of the largest companies in both countries. It also includes news wire data from seven of the largest international English newspapers spanning the same period, as well as economics data from governments and other institutions. In addition, we have collected a new crowdfunding dataset for online social lending that includes detailed description of lending opportunities along with user generated comments over time. Last, with a view on exploiting prediction markets, we make use of a movie review dataset gathered from major American newspapers in order to forecast the future box office revenue of movies.

Our focus is measuring predictability of markets on the medium daily term, instead of the very short or very long terms, though we evaluate our proposed methods on several decades of data. In particular, in our simulation we treat very fine grained unmodelled aspects of the markets as noise, and measure statistical significance of the results from the baseline models.

1.4 Significance of the Study

In this thesis, we demonstrate through a series of experiments that it is possible to consistently earn excess trading profits without taking on excess risk on financial markets. This way, we document market inefficiencies which give evidence against the Efficient Market Hypothesis. Further outcomes of the research are as follows:

- First, we investigate the characteristics of our data sets and we aim to empirically validate the concepts of technical, fundamental, text and time series analyses. We provide supporting evidence that both technical and fundamental analysis contain predictive power for the forecasting of financial markets, and therefore they are valid practices. Next, we investigate whether text content on the internet is predictive of financial markets. We again report positive results in which we show that purely text based predictive systems can achieve significant accuracy in predicting future profits. Last, we investigate temporal characteristics of the data. The outcome is that taking into account temporal variations can significantly increase predictive accuracy. This suggests that financial markets are highly non-stationary systems.

- The second aim deals with exploring model structures to combine various kinds of data in an optimal way in the system. First, we investigate ways of representing the data. Here, we demonstrate that learning representations of data is preferable to using human constructed indicators or features as this way the system can automatically uncover the optimal data representation. This also saves development time in feature engineering. Furthermore, we explore the concept of domain adaptation, and demonstrate a way to extract relevant information for related domains that can boost predictive power of the system. This shows the benefits of knowledge transfer between similar learning tasks where we obtain improvements from using joint models to simultaneously predict related stock markets in consecutive time frames. Finally, we review how best to combine different data sources. We propose multiple ways to effectively combine different modalities of data, (i.e. structured, unstructured, and time), in a unified framework, thereby allowing us to precisely weigh the many different factors affecting financial markets. We show that we achieve the best predictive power by combining all available information into the system.

- The third and final aim deals with the modelling aspects of the systems. First, we show that modelling non-linearities is an important aspect of financial market forecasting. We compare directly with linear models, and show that non-linear models considerably outperform linear ones. Next, we take the concept of non-linearity further, and we show that decomposing the learning problem into multiple hierarchical steps that can be composed of one another improves performance. By constructing a system of multiple deep non-linear projections of the data, we are able to outperform other conventionally shallow methods of learning. Further, we discuss the benefits of incorporating uncertainty in the modelling framework. We show that by exercising a probabilistic interpretation of multiple hypotheses at the same time, we can achieve better predictive performance than by optimising for only a single possible solution. We also demonstrate a way to exploit the uncertainty around the predictions to quantify expected profits and rank investment opportunities. In addition, we investigate what the best way of formulating the objective of the predictive system is. We show that directly encoding the trading profits as the objective considerably increases earned profits, surpassing that of surrogate systems treating the problem as conventional regression or classification. Finally, we make the argument for end-to-end learning where we completely automate the learning process from raw input data to actionable outputs while minimising human heuristics and optimising for the real-world objective of the task. Thereby, we achieve improved predictive performance at reduced development costs. This way, we demonstrate an autonomous trading agent that fully simulates a real world human trader in that it frames the entire trading task as a fully differentiable learning problem, and skips rule based pre and post processing in parts of the pipeline.

1.5 Overview of the Study

In chapter 1 of this thesis, we give a brief outline of the history of research in financial market forecasting and detail some recent changes in the field. We identify a core problem with the current state of the art and propose a way to test whether risk free excess profits can be realized in markets. Next, we list several research questions and note the research outcome for each of them under the scope of this study.

In chapter 2, we review current financial theory regarding the efficiency of markets, and detail several limitations regarding these theories. We review an alternative theory of markets based on behavioural economics and identify environmental and emotional triggers of market participants that may prompt them to act irrationally. Next, we make the argument for computational modelling of markets, highlighting the potential benefits of this technique. Last, we showcase various previous attempts at technical, fundamental and text analysis with respect to market forecasting.

In chapter 3, we review several machine learning frameworks. We show recent advances in linear modelling frameworks with regularized objectives and hint at possible ways of extending them. Then, we move on to probabilistic modelling of markets and discuss advantages and disadvantages of this approach. Finally, we finish with a review of deep representation learning and some recent breakthroughs in this field. We take inspiration from use cases in related fields where machine learning has been successfully applied, as well as investigate ways to adapt these modelling frameworks to the financial domain.

In chapter 4, we introduce a new dataset for modelling FTSE 100 companies listed on the London stock exchange. We design a novel linear model that captures trading profit and directly optimizes for that. We show that this results in improved accuracy to conventional regression models. We give evidence for technical analysis, and propose a new regularization framework to transfer knowledge between related companies in the market, thereby improving accuracy. Last, we show the benefits of doing knowledge transfer over adjacent trading periods in time. Our results are evaluated on more than a decade of data on 100 companies and confidently beat the baseline buy-and-hold strategy. This research is published in (Bitvai and Cohn, 2015a).

In chapter 5, we focus on how we can incorporate uncertainty into the models and uncover inherent non-linearities. We demonstrate that using a probabilistic interpretation of profit, we can achieve better performance and less likely to overfit. We also investigate ways in which we can incorporate trading profit in a Gaussian Process model using approximate inference techniques. Next, we introduce a new dataset for peer-to-peer social lending that we collected for the purposes of this research. We show a way to incorporate temporal information along with structured data and user generated text using a novel non-linear kernel combination technique. We extract semantic content from text using topic modelling and relative term frequencies and identify key characteristics of the data automatically. Finally, we propose a further extension on top of the Gaussian Process framework that allows us to optimize for expected arbitrage profit. Parts of the work in this chapter are published in (Bitvai and Cohn, 2015c).

In chapter 6, we introduce non-linear representation learning. First, we show that on a movie review dataset, we can achieve significant improvement in prediction accuracy by learning a deep representation for the text instead of a bag of words approach. This results in 40% relative improvement over the previous state of the art. We propose a convolutional approach for text representation that is combined with structured data in a hierarchical manner. Last, we propose a new way to identify key phrases in the text that have the highest impact on the predicted movie revenues. Next, we introduce a new dataset of over 500 companies listed on the New York and Nasdaq stock exchanges along with text from several news papers and economic data for the span of almost 20 years. We propose an end-to-end learning framework to capture the different modalities of the data, model temporal dynamics in the markets both locally and globally, and finally output limit order trades directly, all the while maximizing for the net worth of the portfolio of the trader. Using this method, we beat the baseline buy and hold with statistical significance. Parts of the work in this chapter are published in (Bitvai and Cohn, 2015b).

In chapter 7 , we draw conclusions for the current research and give recommendations for future directions.

Chapter 2

Financial Market Modelling

In the previous chapter, we have given a brief introduction to this thesis. In this chapter, we evaluate the concept of applying artificial intelligence techniques to market trading. Further, we review relevant literature to financial market prediction and establish dominant theories of markets, citing reasons why it may or may not be possible to forecast markets. Then we investigate prevalent techniques used for market forecasting, and identify several shortcomings of these. Finally, we review recently proposed computational methods in machine learning literature and examine how these methods could be adapted to a financial domain.

In the first part of the review, we aim to give a high level justification for using artificial intelligence for financial market trading. We make the argument for computationally modelling markets, giving reasons why this may be feasible from a practical point of view. In particular, we focus on the idea of applying machine learning techniques to trading, and highlight the benefits and possible disadvantages of taking such an approach. We discuss the issues of passive versus active investing, and under what circumstances one might pick one or the other. We then elaborate on the challenges of constructing a good approximate model of markets given limited resources and how we could incorporate the various factors affecting markets into our model. We further evaluate the potential of defining and capturing arbitrary value functions in our framework that may reflect the preferences of the trader. In the end, we highlight some of the differences between theoretical versus data driven approaches to market prediction.

After the above discussion of artificial intelligence (AI), we dive into pure financial theory. We cover the concepts of the Efficient Market Hypothesis (EMH) and the related Capital Asset

Pricing Model (CAPM) and Modern Portfolio Theory (MPT) that have been historically the defining theories in the financial literature for investing. We note that according to these theories, it is not possible to forecast markets and realize excess risk free profits. Next, we briefly mention Behavioural Economics which states that inefficiencies do exist in the markets in the short term, where emotional and occasionally irrational thinking could distort the true prices of assets. Here, we review several phenomena in markets that have been observed but for which no clear explanations exists.

After reviewing these dominant theory of markets, we move on to how people previously attempted to construct a model of markets for forecasting and what the limitations of these models were. We cover fundamental analysis that deals with investigating the properties of financial assets, technical analysis that deals with forecasting future value of assets from historical observations, and text analysis that addresses the issue of extracting signal about market dynamics from human communication via generated texts, such as newspapers articles or social media posts. We also address the question as to whether the Efficient Market Hypothesis is still relevant today.

Next, we take a detour to consider our setting of financial market modelling. In particular, we explain what experimental setups we consider for modelling markets, what datasets we use, how we define the objectives of the experiments, and how we evaluate the results. This gives us an overview of how we can apply market modelling techniques in practice and ties in with the purpose of this study, which is to document and highlight inefficiencies in markets.

Following this train of thought, we next explore in depth various computational and machine learning approaches to market forecasting and review relevant advances in this field with regard to predicting in various domains, with use cases coming from natural language processing, computer vision and speech recognition. We examine how we could apply some of these methods to financial markets coupled with fundamental, technical and text analysis, and what modifications may be necessary to achieve good performance.

We categorize these computational methods into two categories: linear and non-linear, though this distinction in some cases is not that clear. In the first part, we cover linear models and sophisticated regularization techniques. Linear models are attractive because they are robust to overfitting, a problem commonly encountered in modelling complex systems such as markets. Furthermore, various multi-task regularization techniques allow us to make linear models progressively more complex in a controlled manner.

Next, we progress to non-linear modelling techniques where we first review Gaussian Processes (GP), and show how we can inherently model non-linear functions via kernels. In addition, GPs allows us to inject uncertainty into the model and enable a probabilistic interpretation of the output, which are both very attractive in financial markets due to the large amounts of noise. This also allows us to quantify probabilistic expectations which we may have regarding the evolution of markets.

Last, we review significant recent breakthroughs in deep representation learning in various fields. These fully differentiable learning machines originally based on Artificial Neural Networks (ANN) are attractive because they make minimal assumptions about our problem domain and enable us to perform end-to-end learning with the possibility of completely simulating a real life human trader all the way from reading raw data feeds to placing trades on the markets. As markets are affected by a multitude of factors and as a result are highly dynamic, the property of being able to jointly fully model this complex system can be beneficial to explore. In addition, ANNs give us the ability to decompose a complex problem into multiple hierarchical steps, which again given the intricacy of modern markets can be highly useful for efficient computational learning.

2.1 Role of Computational Modelling

In this section, we investigate several properties of financial markets and highlight ways in which machine learning could be beneficial in modelling them. We first begin with an overview of the future challenges of computer based trading.

2.1.1 Future of Computer Based Trading

The Foresight report (Beddington et al., 2012) is a UK governmental report aimed at assessing the future of financial markets and how high frequency trading (HFT) and algorithmic trading (AT) impact them over the next 10 years.

First, the cost of computing power is predicted to significantly decrease through the widespread availability of cloud computing. Special purpose built hardware may improve the speed of computing too. This will lead to the phenomenon of autonomous robot based trading increasing considerably. More and more trading activity will be carried out by robots between themselves, to the detriment of human trading. Markets will become social-technological systems, where robots and human traders will coexist for some time.

With the transfer of knowledge from research to production, small and medium sized middleware suppliers for trading systems may provide specialised components as market services to other market participants, which will make the technology available more widespread. Therefore, the key driver for the future financial systems will be technology. As such, the adoption of technology will be the primary determining factor in the establishment of new financial centres. This way, emerging countries quick to adopt new technologies may challenge the existing hegemony of Wall street and western financial centres. Thus, the fin-tech revolution is expected to have profound effects in the industry.

There have been many key advantages to computer based trading. First, transaction costs have significantly dropped, due to the availability of better market structures facilitated by high frequency trading. The bid-ask spread of most financial instruments has also significantly reduced due to better forecasting availability of trades. Trading systems are becoming more efficient, and can operate with smaller amount of capital. The price discovery of instruments has also improved due to the better prediction accuracy of the systems, which makes markets more efficient overall.

On the other hand, while generally trading activity and liquidity have increased in markets, there have been dangers of periodic illiquidity and market instability. That is because high frequency traders do not have the obligation to provide liquidity and can rapidly react to changing market dynamics, as seen in flash crashes. Regulatory frameworks need to address these challenges, while not suffocating innovation in the industry. It is generally believed that robots have increased volatility in the markets. However, there is little practical evidence to support this statement. Nonetheless, robots versus robot trading can give rise to self reinforcing feedback loops that can have adverse consequences, even if there are control systems in place to mitigate their effects.

Some potential policies that may be beneficial to regulate computer based trading (CBT) are as follows. Circuit breakers may be effective when there is temporary imbalance in the order book. For example, this can halt trading for some time during market crashes, but coordination across markets may be a challenge. This can stop the negative feedback loops from going out of control. Another approach may be to enforce a minimum tick size of major markets so that up-to-date data is not instantly available. This can disincentivise rapid firing of HFT systems. Other options may be the introduction of notification of algorithms, minimum order resting time, limiting order-to-execution ratios, different maker-taker pricing, a virtual central limit order book, constraining dark trading, or enforcing call auctions. However, these options may have negative economic benefits in practice.

It is also hard to determine whether computer based trading has increased market abuse through predatory trading. There is generally a lack of data available to researchers, due to most data sources being kept secret. Therefore, manipulation of prices cannot be conclusively proven. However, care must be taken to address perceptions of abuse in the public eye, as this can have adverse effects on the economy. One way to do that is to invest into detection of abuse by analysing vast amounts of standardised data, similar to the way it has been implemented in the USA via the Dodd-Frank Act.

Furthermore, financial trading is likely to increase in complexity due to rapid technological advancement. This makes it increasingly difficult to understand. This may exaggerate the information asymmetry of market agents, and erode trust in the system, which in turn makes the market operate sub-optimally. One way to combat this challenge is to force exchanges to publish accurate, high resolution, synchronized timestamped and real-time data publicly in a standardised format, and greatly improve the connectivity of their platforms.

All in all, CBT has improved the efficiency of markets in multiple ways, despite the controversy surrounding them. Major benefits include increased liquidity, reduced transaction costs, and more efficient price discovery. However, big challenges remain in coordinating regulation across all markets in Europe and elsewhere globally. Markets are turning into socio-technical systems, at the meeting point of software engineering and finance. Behaviour leading to increased systemic risk that may not be immediately tangible to individual traders must be properly discouraged. Standardisation should be applied where possible. Lessons can be learned from the safety critical industry in order to mitigate systematic risk. Additional software techniques to discover malpractice by mining financial data may improve trust in the system as a whole. This will require the making of this confidential data available to regulators and academia.

Since finance is globalised, this will need to be done on an international scale.

We next move onto emphasising the utility of active investing below.

2.1.2 Active versus Passive Investing

Active investing refers to the practice of doing an active analysis of investment opportunities in order to pick the best one for investing. On the other hand, passive investing recommends investing in all assets with high spread (diversification) in order to match the overall average growth of the market. The Efficient Market Hypothesis (EMH) states that markets are informationally efficient, and therefore no risk free adjusted profits can be made with or without insider information (Malkiel, 2003). This hypothesis is based on the assumption that information can freely propagate in a network and everybody has access to the latest and most advanced pricing models available. However, in practice frictions exist in markets, and hence it may be possible to uncover inefficiencies. To take this into account, the efficient market hypothesis is sometimes modified to claim that markets are efficient only in the very long term (Fama, 1998), but may not be over the short term. By this theory, doing any analysis of price is completely unnecessary as one can never outperform the market rate of return in the long run. Therefore, passive investing should be preferred to active.

Here, we propose that it is likely the case that proponents and opponents of the EMH are both right. To see this consider the following two scenarios. First, if everybody was a passive investor that considered markets fully efficient and accepted the price of shares without doing any analysis, then this would give opportunity for a market participant to turn into an active arbitrage maker and earn above market rate of returns at no risk. That is because they would be free to manipulate the price of the asset since other participants rely on them for making a judgement on what the price should be. The arbitrage maker can count on the fact that there would be always a *greater fool* (Dooley and Shafer, 1976) who will buy the asset from them at worse prices than he originally bought. On the other hand, next consider a different scenario where everybody tried to actively discover arbitrage opportunities in the market by performing independent analysis of the underlying intrinsic value of the asset. In this case, likely all inefficiency would cease to exist and people might as well passively accept the price of the asset as accurate and stop doing analysis, thereby saving the costs of analysis. That is because they can count on the fact that there would be always a *lesser fool* who will do the price analysis for them, and therefore they do not need to perform this themselves.

This situation gives rise to an equilibrium where the efficient market hypothesis can hold true only because there are others who reject its validity. Therefore, it is vital for individuals to continue to actively uncover inefficiencies in the market using advanced methodologies such as machine learning, so that other passive investors are able to enjoy the benefit of this work without them having to do it. Hence, the efficiency of markets may generally hold true even though above market returns can be achieved by select individuals on a consistent basis. This can be regarded as a service provided to passive investors who effectively benefit from the work of active investors while paying for it with slightly worse prices. Of course, performing this pricing analysis work is dependent on the individual's ability to correctly approximate the processes underlying markets, which we discuss next.

2.1.3 Approximating Markets

To become an active investor, people have to construct a model of markets, which they then use to quantify the attractiveness of various investment opportunities. Since "all models are wrong, but some are useful" (Wolkenhauer and Ullah, 2007; Box, 1976), these models often attempt to *approximate* the underlying dynamics of markets as best as possible. Approximating markets is a very difficult task, as they are often claimed to be chaotic systems with high levels of noise (Magdon-Ismail et al., 1998; Ghosn and Bengio, 1997). This means that small changes in initial conditions could lead to unpredictable outcomes. Since markets are affected by participants that need to monitor the behaviour of all other participants, this can often lead to an exponential number of interactions, giving the appearance of a chaotic system. However, high levels of noise and unpredictability are often symptoms of a process where the complexity exceeds the computational capacity of the modelling framework. It may be the case that with more powerful techniques, what might appear to be noise or chaos in the big picture may not be when looking at the fine grained details. A more efficient and better approximation of the complex world of markets may well give rise to temporary trading advantages in forecasting the system, at least until other market participants also gain the capability of increased forecasting power themselves, at which point any advantage may disappear. In evolutionary biology, this never ending race for progress and efficiency between competing agents is often described as the Race of the Red Queen (Ridley, 1994).

All such forecasting systems must be limited by the resources of the real world. Still, extremely high levels of efficiency may well give rise to the apparent illusion of a chaotic and noisy system. Granted if no market participant attempted to improve their understanding of the world, and passively accepted the market prices as they are, then markets could be claimed to be at stable equilibrium. However, in practice this is not the case, as market participants compete with each other to discover and trade inefficiencies away and new information constantly arrives and becomes available to act on. Machine learning is a useful toolkit in the hands of the trader because it provides fast evolving cutting edge technology to approximate arbitrarily complex functions of changing market behaviour efficiently, trading off approximation accuracy and speed with resource consumption directly. As the field of machine learning has itself gone through a revolution in the last few years, it may be the case that new techniques that have recently emerged are capable of uncovering inefficiencies in markets previously not known publicly or privately. Statistical methods are a form of machine learning that has been successful in the past in uncovering patterns in data.

In summary, that means markets are likely to be efficient but only to the extent that doing the additional analysis of uncovering further marginal inefficiencies and benefiting from them does not exceed the cost of doing said analysis. In this view, we can regard the latent inefficiency of markets as a resource in itself where people compete with each other for their discovery and exploitation. This suggests that it is possible to gain a competitive advantage and earn excess profits at no additional risk by either reducing the cost of analysis or by increasing its effectiveness, for example through more advanced computational methods for approximating latent market processes. An additional way of improving accuracy is to feed into the model a diverse set of datasources that capture the various factors affecting markets. In the next section, we detail what these sources might be.

2.1.4 Factors Affecting Markets

By becoming an active investor, one aims to construct an approximate model of markets that can be greatly improved by feeding into it data from various sources that are presumed to describe the market dynamics. Markets are affected by countless number of factors as they are part of the interconnected web of society. These factors are various manifestations of human behaviour. As we have seen above, a key assumption of EMH is that market participants are selfish rational utility maximising agents. However, in related research, scholars argued that people often act irrationally, following their emotions (Kahneman and Tversky, 1979). Money matters usually profoundly affect the livelihood and security of people, which is one of the top Maslow hierarchy of needs in psychology (Maslow et al., 1970). Therefore, many times people react out of fear or greed in the markets or simply follow a herding behaviour. What this suggests is that if enough people act this way, then consistent predictability may exist in historical data. Emotional behaviour of humans could manifest itself in a multitude of ways, such as in historical trading actions, or through human language. Text online is one of the main ways humans communicate information and express their emotional state. In addition, data from the broader economy can explain the context through which the individual humans function in the larger society. Last, historical patterns of trading, which may be guided by some predefined rules or emotions, could be visible in historical trade prices and volumes, and therefore may harbour latent signals of future trading behaviour.

As we can see from the examples above, markets are affected by a multitude of different sources of information that are conveyed through different *modalities*. Machine learning is a useful tool to aggregate information from all these different sources and analyse their complex interaction. We can feed historical market data, economic data and social media text data into machine learning models and measure how they affect a metric we care about when constructing the modelling system.

Market prediction is a difficult task, partly because markets are affected by a huge number of factors in the interconnected system of the world. Even though hard sciences such as physics deal with a very complex world that follows some set of rules, here we add the whims of human behaviour to the mix, a human being one of the most intriguing objects ever studied. In addition, due to the fact that inefficiencies are always discovered in historical datasets, they may no longer be present in future datasets. That means, a state of the art machine learning model may only discover noise in new data, due to the fact that any previous signal has been exploited by other market participants. Therefore market forecasting is a very difficult benchmark for any kind of modelling technique, which makes it an ideal target for assessing the state of the art machine learning techniques against. Sometimes, the state of the art may not even be publicly known as people tend to keep recent innovations secret in order to trade inefficiencies themselves. Therefore, there is a clear need for transparent research in this field in order to shed light on the issue of market inefficiency and determine the factors affecting markets using the latest modelling technology available.

This peculiar characteristic of research in this field has a further consequence. As previously noted, market participants are in a race for efficiency, which means the underlying processes that affect markets may change over time, thus making them a highly dynamic system. For example, as humans are always in need to increase their own returns or utility, as soon as some inefficiencies are discovered, they may be quickly arbitraged away by market participants, making them not present in future data. A machine learning model can adapt to changing market conditions like these and update its parameters over time, allowing for the modelling of non-stationary underlying processes. It can adjust for shifts in the signal in order to discover new inefficiencies and maximise the utility gained from exploiting them. How we define the concept of utility or value is discussed in the next section.

2.1.5 User Defined Value

Another key question that needs to be addressed when discussing the possibility of achieving risk free profits is the concept of value, which is the quantity traders seek to maximise. Value can be defined in multiple ways, some of which are exchange value, utility value, intrinsic value or price value (Nicholas, 2011). There is no universally accepted definition of what the value of something is as there is inherently an element of subjectivity to it. One example for this in traditional finance is the risk-reward trade-off, where people may choose investments with higher or lower rewards and risks suited to their personal preferences (Sharpe, 1970). Therefore, it can be postulated that while the markets may evolve rapidly over time as new technology becomes available, it may still be possible for individual market participants to consistently improve their own utility in a way they define it by engaging in selected trades. Trading or exchange of goods has been known to increase the overall welfare of society as it matches the varied needs of different individuals and boosts productivity (Winters, 2004). By having more advanced ways of uncovering actionable knowledge from the data of other market participants, an individual may be able to negotiate a better bargaining position for themselves in an exchange using their own criteria of value, as often transactions or deals can occur in a given range of prices rather than at a fixed price. Therefore, while overall markets may be efficient, individual participants may consistently increase their own utility better than just accepting a default market rate of return provided by passive index funds.

In addition, the value participants give to a particular investment opportunity may be defined not exclusively by the rate of returns they obtain. For example, while investing in small medium sized business, many people cite a warm fuzzy feeling of helping ordinary businesses succeed. Likewise, people may invest in opportunities for varied reasons which may be idealogical, wanting to see an idea succeed, feeling connected to a community, or acquiring a social status by association with said investment, such as in crowdfunding projects (Gerber et al., 2012). Obtaining social status may in fact be the ultimate motivator in some cases, surpassing money (Deterding, 2012). People may also derive broader indirect utility, such as general growth in the local environment they live in, rather than just the direct monetary compensation they receive from a specific investment. After all, money is only one way for an individual to exert influence over their environment. In this regard, using machine learning as a tool for trading is well suited, as it allows for the definition of arbitrary utility or value functions, which can factor in the various facets of individual preferences. In the next section, we will finish the discussion of machine learning for trading with a comparison to more traditional theory driven approaches to market prediction.

2.1.6 Theory versus Data

Machine learning provides a data or evidence driven approach to finance that is founded upon the concept of being able to learn from an observed behaviour that had a specific outcome. On the other hand, there have been multiple alternative hypotheses proposed to explain markets. Conventional financial theories including the Capital Asset Pricing Model and Modern Portfolio Theory deal with idealized circumstances when constructing theories about how markets operate. One such example is that returns on a markets follow an independent normal distribution. Unfortunately, many of these assumptions do not hold in real life, therefore it is vital to impose a limit on them. A data driven approach provided by machine learning is a well suited way to reduce the amount of assumptions in theories and let the data “tell the story” by constructing viable hypotheses of markets. As it is not possible to repeat market history with different pasts under a controlled environment, the next best approach is to use historical data as a way to forecast markets, and skip human constructed heuristics. Machine learning and high performance computing gives us the ability to study a problem in details via intricate modelling capacity, which is something that was not possible when the theories of CAPM and MPT were first proposed.

Another important aspect of considering the existence of predictive signal in historical data is the influence of chance and modelling artefacts. Therefore, the presence of such signal in historical data is an often hotly debated topic. It is argued that although many models work on paper, in theory, or under the circumstances of the experiment, when put in a real life setting, they fail to produce good results. For example, it has been shown that many hedge funds actually perform in line with the market return over a long period of time, even though they may in sub periods outperform the baseline (Jensen, 1968). Therefore, it is important to evaluate any strategy on a long and large dataset, and possibly in a real life setting. Doing extensive backtesting is possible by gathering datasets stretching long periods of time and different market conditions, and then validating the proposed machine learning models under all these different circumstances. It is also important to note that there is a small percentage of individuals who can reliably produce a positive *alpha*, meaning they consistently outperform the market. Using historical data in the South Sea bubble, it was shown that a trader reliably and consistently outperformed and rode an economic bubble (even though the existence of bubble itself is contested), and moreover, the trader achieved this performance very likely without insider knowledge (Temin and Voth, 2004). Therefore, machine learning is suitable to uncover a consistently profitable trading strategy from large datasets, in case such a strategy indeed exists.

This concludes the high level of overview of applying artificial intelligence to financial market trading. In the next section, we give a more detailed introduction to various traditional theories of markets and their limitations.

2.2 Inefficient Markets

In this section, we review the Efficient Market Hypothesis (EMH), a theory still relevant today (Degutis and Novickyte, 2014), and the related Capital Asset Pricing Model (CAPM) and Modern Portfolio Theory (MPT). We show what assumptions they make and list circumstances under which these assumptions may not hold. In addition, some of these circumstances are drawn from research in behavioural economics, which highlight the inefficient nature of markets.

2.2.1 Efficient Market Hypothesis

Stock markets, and in general financial markets, are thought to contain a large amount of noise and exhibit non-linear, even chaotic patterns (Magdon-Ismail et al., 1998; Ghosn and Bengio, 1997; Sornette, 2009). The Random Walk Hypothesis states that effectively the price movement of companies on the market evolve in a random walk and therefore they are fundamentally unpredictable (Magdon-Ismail et al., 1998). This is due to the Efficient Market Hypothesis which states that the price of a stock already contains all the available information about the asset, therefore the market is informationally efficient (Malkiel, 2003). According to this theory, in the long term one cannot beat the market consistently through speculation. Given that all investors are rational utility maximising agents as described by the Rational Choice Theory (Becker, 2013) that have access to the same information, everyone is in the same bargaining position. As a result, prices reflect the intrinsic value of assets or commodities. Therefore, technical analysis, which deals with historical price patterns, and fundamental analysis, which deals with present patterns in company performance, are both ineffective, and above market returns cannot be achieved. “A blindfolded chimpanzee throwing darts at the Wall Street Journal could select a portfolio that would do as well as the experts” (Malkiel, 2003) and “the stock market in the short run may be a voting mechanism, but in the long run it is a weighing mechanism” (Graham and Dodd, 1934). This means that in the long run the true values of assets emerge, but in the short term this may not be the case. As such, there is evidence that inefficiencies exist in a short time horizon and they primarily come from undiscovered patterns of market behaviour. As technology progresses, markets become more efficient, but never reach full efficiency. A well-known story that illustrates the concept of being “efficient” is of a professor and a student who come across a $100 bill lying on the ground. As the student stops to pick it up, the professor says, “Don’t bother — if it were really a $100 bill, it wouldn’t be there.” In theory, there should not be any bills lying on the ground, but in practice, one may encounter one, especially if they know where to search for it.

Malkiel notes that in the real world there are market phenomena that can be interpreted as signs of inefficiencies. In the following, we list some of these phenomena. One source of inefficiency is short term momentum and under-reaction to new information (Lo et al., 2000). This may be attributed to the psychological bandwagon effect (Shiller et al., 1984). On the other hand, this may not allow investors to make excess profits with transaction costs. In addition, under and over reaction to news may in fact cancel each other out. Another source of inefficiency is long-run return reversals. That is stocks are expected to reverse to their mean values in the long run. However, in the short term investors may be overconfident and thus biased (Kahneman and Tversky, 1979). A third source of inefficiency is seasonal patterns. The January effect states that high returns can be achieved in that month due to tax filing in December (Haugen and Lakonishok, 1987). This effect has vanished since its wide spread discovery. Prices could be predicted from financial ratios too, such as initial dividend yields and initial price earnings multiples. Recently, these values have been relatively constant though. According to the size effect, smaller companies tend to outperform larger companies in economic performance (Bondt and Thaler, 1985). However, this finding might change depending on how risk is measured, or by the survivorship bias of only measuring companies that did in fact survive harsh economic times. Furthermore, value stocks may have higher return rates than growth stocks, as identified by the price to earnings and book value ratios of said companies (Nicholson, 1960). The equity risk premium puzzle shows a situation where investors prefer bonds to common stocks even though that results in lower risk adjusted returns for them (Weil, 1989). However, this may be a result of the US stock market survivorship bias during the 1920s-1940s relative to other markets. Last, some patterns may be simply a result of statistical artefacts or data snooping bias. The 1987 market crash and the 1990s Internet bubble are examples of short term market inefficiencies, though they may be related to external environment changes (Malkiel, 2003). There is also some evidence that actively managed funds under-perform passively managed index funds by their added expenses (Jensen, 1968). Therefore, a simple model with maximum diversification that spreads the risk and invests equally in all assets yields better returns than a complex model that aims to select stocks by active analysis. These examples show that there are several cases of alleged market inefficiencies in economic history, but there is no academic consensus as to what the cause of these inefficiencies are. Nevertheless, these observations suggest that active portfolio management could outperform passive management by exploiting inefficiencies in the market. In this work we seek to exploit such inefficiencies though machine learning models of market trading, and in doing so providing empirical validation against the Efficient Market Hypothesis.

Further source of inefficiencies include information imbalance or asymmetry that makes market participation unequal, since information does not propagate instantly across the market. There may be differences in the asset valuation functions of market participants too. In addition, Prospect theory behavioural economics shows that humans are subject to cognitive and emotional biases and therefore they are prone to make sub-optimal financial decisions because they rely on their intuition at least some of the time (Tversky and Kahneman, 1974; Kahneman and Tversky, 1979). In markets where volatility is a problem, i.e. investments cannot be traded quickly enough, predictable patterns may emerge. Nevertheless, it has been shown that soon after publishing the discovery of such patterns that may enable excess risk adjusted profits to be made, these opportunities are quickly arbitraged by investors (Malkiel, 2003), as a result the significance of the patterns may shrink, though not vanish completely.

Although, it may be the case that in whole the market achieves average returns, individual investors can gain above market returns. Active management refers to a portfolio management strategy where the manager makes specific investments with the goal of outperforming an investment benchmark index. In passive management, investors expect a return that closely replicates the investment weighting and returns of a benchmark index and will often invest in an index fund (Grinold and Kahn, 2000). The greater fool theory describes a situation where the price of an object is not being driven by its intrinsic value, but by expectations that irrational bidders for limited assets set the price (Dooley and Shafer, 1976). This can lead to speculative bubbles, as there is always a greater fool that will buy the asset. On the other hand, proponents of index funds believe in the lesser fool theory which states that diversification leads to maximum market returns, as there are always lesser fools that do the necessary work of determining the intrinsic value of assets.

In practical terms for markets to be fully efficient the following must be true: an absolute absence of emotion in human investment decision-making; a universally accepted pricing analysis system for assets; the willingness of all investors to accept that their returns will be identical to that of any other market participant; an unrestricted access to high-speed and sophisticated pricing analysis systems. Currently, not all of these requirements are satisfied. These examples show that markets may not be efficient at all times. Next, we review the Capital Asset Pricing Model and its limitations. This theory relies on the Efficient Market Hypothesis to price assets.

2.2.2 Capital Asset Pricing Model

The Capital Asset Pricing Model (Markowitz, 1968; Sharpe, 1964; Grinold and Kahn, 2000) describes the relationship between expected return and risk in order to price assets. The main claim of CAPM is that excess returns only come from assuming excess risk. This is in direct contrast with active portfolio management that aims to identify excess return without excess risk.

CAPM states that all returns on the market can be separated into the market rate of return, and the residual return of an asset. The market rate of return is achieved by a consensus in the market, whereas the residual is expected to tend to zero. One formulation of CAPM is shown by

$$E(R_i) = R_f + \beta_i(E(R_m) - R_f) + E(R_r) \tag{2.1}$$

$$\beta_i = \frac{Cov(R_i, R_m)}{Var(R_m)} \tag{2.2}$$

where $E(R_i)$ is the expected return on the capital asset i over time periods, R_f is the risk free rate of interest such as the rate of government bonds, $E(R_m)$ is the expected return of the market, $E(R_r)$ is the expected residual return on the asset, which tends to $E(R_r) = 0$, and β_i is the sensitivity of the excess asset returns to the excess expected market returns. The term $E(R_m) - R_f$ is sometimes referred to as the market premium whereas the term $E(R_i) - R_f$ is known as the risk premium. Market risk (also known as undiversifiable risk, volatility or systematic risk) is the inherent risk in the market that indicates that the investment portfolio value may decrease. It is denoted by $\sqrt{Var(R_m)}$. Residual risk (also known as diversifiable risk, specific risk, idiosyncratic risk or nonsystematic risk) is the risk in a specific asset that can be reduced by combining several different assets in a portfolio.

Active management can provide value by maximising the residual returns above the consensus market rate of return. Therefore, the portfolio of an active manager should not match that of the consensus market portfolio. However, according to CAPM, any residual return expected above zero is not possible, therefore active management is unnecessary. Higher returns only come with higher risk. That means market participants are only compensated for taking on higher market risk and not for taking on higher residual risk. The riskiness of a portfolio above the market rate of return risk is shown by beta. A beta of zero is equal to a risk free asset return, and a beta of one gives the market risk and rate of return. A beta above one indicates that the portfolio risk is larger than the risk of the market returns.

The three versions of the Efficient Market Hypothesis are consistent with CAPM. According to the weak version of EMH, one cannot beat the market using only publicly available historical prices and volume data. The semi-strong version states that one cannot beat the market using historical data and fundamental analysis, or using analysts' insights. The strong version of EMH states that prices reflect all available information all the time, and even with insider information, one cannot beat the market (Jensen, 1978).

In case there is a group of people who hold above market rate of return basket of goods, then there must be another group who hold negative residual return of goods. The latter group could protect itself by following a passive investment strategy and buying into the market rate of return. However, in reality people seldom admit that they are greater fools and like to think of themselves as they are the ones outperforming the market. For example, when a group of Hardvard Business school students were asked about their salary expectations, 80% of them claimed their salaries would be larger than the average graduate salary (Grinold and Kahn, 2000). People make their investment decisions based on future projections they construct, and their actual returns are only available on a historical basis.

CAPM talks about expectations and expected returns. The actual realised returns of any particular investor may be different from the market rate of return. If the portfolio beta is greater than the market rate of return, and as beta increases, the portfolio returns decrease, then the recommended investment is to invest in an almost risk free asset. Otherwise, if the expected returns increase with larger risk, i.e. with higher beta, then the investor is only compensated for taking on excess risk through a higher return. This is illustrated by the efficient frontier of random portfolios in Figure 2.1.

In practice however, market players may have different information and their consensus of what the market rate of return is can be ill-defined. They may also have different expectations with regard to the future evolution of the market. The burden of proof is on the active manager to prove that they have superior insight into the inner workings of the market, which enables them to beat the consensus rate of return. An active manager can avoid the risk of accurate market timing, and focus his analysis on finding positive residual returns, where the consensus is expected to tend to zero. Forecasting the future beta of stocks can also be regarded as a way

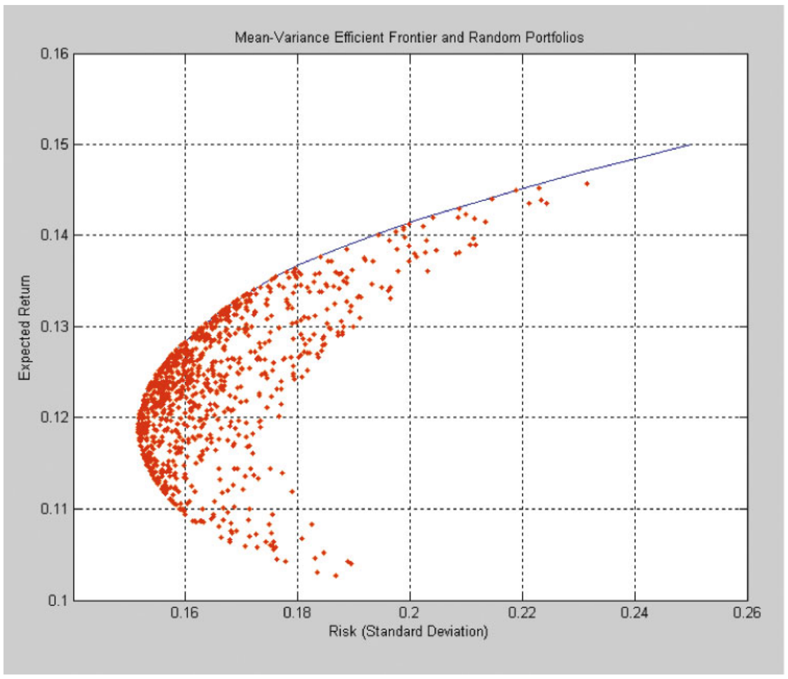


Figure 2.1: The efficient frontier showing risk-reward trade-off in the Capital Asset Pricing framework. At the frontier, excess return can only be achieved by taking on excess risk, and the goal is to construct a portfolio lying on the frontier expected return with different personal risk preferences.

of forecasting the returns of the asset (Grinold and Kahn, 2000), which is one of the goals of active portfolio management.

This relies on the fact that there is a linear relationship between beta and stock returns (Jensen et al., 1972). However, subsequent research showed that this relationship may not always hold (Fama and French, 1992). This concludes the review of the Capital Asset Pricing Model. In the next section, we discuss Modern Portfolio Theory and its limitations, which relies on CAPM to construct an optimal basket of goods on the market.

2.2.3 Modern Portfolio Theory

In the previous sections, we have outlined the weaknesses of the EMH and CAPM theories. In this section, we review some of the criticism associated with Modern Portfolio Theory, which extensively makes use of both EMH and CAPM.

Modern portfolio theory (MPT) is a theory of finance that attempts to maximise portfolio expected return for a given amount of portfolio risk, or equivalently minimise risk for a given level of expected return, by carefully choosing the proportions of various assets and making sure the portfolio lies on the Efficient Frontier of the risk-reward curve of the market (Sharpe, 1970). Although MPT is widely used in practice in the financial industry and several of its creators won a Nobel memorial prize for the theory, in recent years the basic assumptions of MPT have been widely challenged by fields such as behavioural economics.

The framework of MPT makes several assumptions about markets and investors (Xidonas et al., 2012). Some are explicitly stated in the equations, such as modelling returns as Normal distributions. Others are implicit, for example not accounting for tax or transaction fees. Each of these assumptions compromise MPT to an extent as they generally fail to hold up in real life. Next, we review some of these assumptions.

Investors are interested in optimizing the mean-variance tradeoff of returns. In reality, investors may have utility functions that are sensitive to higher moments of the distribution of the returns, such as skewness or kurtosis. Generally, the utility function is assumed to be quadratic or exponential.

Asset returns are jointly normally distributed. In fact, returns in various markets are frequently observed to be not normally distributed. Price points up to six standard deviations from the mean occur far more frequently than predicted (Mandelbrot and Hudson, 2014). Another common assumption is to model returns with joint elliptical distributions(Chamberlain, 1983; Owen and Rabinovitch, 1983), which are symmetrical in nature while empirical asset returns are typically not.

Correlations between assets are fixed and constant. Correlations depend on systemic relationships between the underlying assets, and when market conditions change, these relationships change as well. Such events might include a declaration of war on a country or a market crash. During financial crises many assets tend to be highly correlated. This leads to a breakdown of MPT precisely when investors need to be protected from risk the most.

Investors aim to maximise economic utility. This is often described as the amount of money they make, regardless to any other considerations humans may have. This assumption is also stated in the EMH which is a cornerstone of MPT.

All investors are risk-averse and rational. According to behavioural economics, people sometimes act irrationally. For example they act on their emotions or on stale market information, they may follow a herd behaviour, or may seek risk for the sake of it. Casino gamblers can be viewed as risk-seeking, as well as some traders. Rationality is an assumption of the EMH too.

All investors have access to the same information at the same time. In reality, markets exhibit information asymmetry, insider trading, and some people are simply better informed than others. Moreover, estimating the mean and the covariance matrix of the returns between assets are difficult statistical tasks. In case the price is modelled as a Brownian distribution, estimating the drift parameter is also hard.

Investors have an accurate conception of possible returns. In addition, their beliefs match the true distribution of returns. Alternatively, investors' expectations could be biased, which makes market prices informationally inefficient. In behavioural finance the overconfidence-based asset pricing model (Daniel and Titman, 2006) makes psychological assumptions to propose an alternative theory to CAPM.

There are no taxes or transaction costs. In reality, financial products are subject to taxes and transaction costs, i.e. broker fees, and this can significantly change the optimum portfolio composition.

Investors are price takers, and have no influence on prices. However, sufficiently large individual transactions could shift market prices, even for related markets due to cross elasticity of demand. It may not even be possible to assemble the optimal portfolio without moving the market too much.

Investors can lend and borrow unlimited at the risk free rate of interest. On the contrary, every investor has a limit for their credit.

Security sizes are infinitely divisible. However, most often fractional shares cannot be bought or sold, and there are also minimum order size limits for certain assets.

Risk or volatility of an asset is constant. In fact, bubbles such as the US mortgage and European debt crisis could lead to severely mispriced risks that change rapidly.

Although these points question the premise Modern Portfolio Theory is built on, in this work we do not reject MPT, but rather empirically test which aspects of it work in select market environments.

This section concludes our review of traditional financial theory. We have seen that although these theories are widely used in practice, their assumptions may not hold in every real world scenario. Therefore, inefficiencies may in fact exist in financial markets. In the next two sections we discuss previous work related to market forecasting and active portfolio management that attempt to exploit these inefficiencies.

2.2.4 Technical Analysis

This and the next section deal with previous methods for forecasting the behaviour of financial assets. We first begin with technical and then move on to fundamental analysis.

In finance, technical analysis is a security analysis methodology for forecasting the direction of prices through the study of past market data, primarily price and volume. Behavioural economics and quantitative analysis use many of the same tools of technical analysis, which, being an aspect of active management, stands in contradiction to much of modern portfolio theory. The efficacy of both technical and fundamental analysis is disputed by the efficient market hypothesis which states that stock market prices are essentially unpredictable. Since stock price movements appear to be chaotic, some theories of finance refute the possibility of realising risk free profit through predictive modelling. Despite this, a large body of technical analysis work maintains that price movements can be predicted solely from historical price data and markets are not completely efficient.

Certain kinds of technical analysis have been known to reveal behavioural patterns in trading activity on stock markets (Lo et al., 2000), whereas other kinds have had less success. Technical analysis has the potential to uncover behavioural cues of market participants and capture psychological biases such as loss aversion or risk avoidance. In its arsenal, sound statistical and quantitative analysis methods can be found in addition to pattern matching functions. Furthermore, it assumes that the price of a security embodies all available market information. It is in contrast with fundamental analysis that attempts to analyse external factors such as company performance reports and financial indicators as opposed to purely market information such as history of stock movements, in order to make predictions about the future. The three pillars of technical analysis are history tends to repeat itself, prices move in trends, and market action discounts everything (Lo et al., 2000; Neely et al., 1997). This means that all past, current and even future information is discounted into the markets, such as emotions of investors to inflation or pending earnings announcements by companies. Only unknowable information such as earthquakes is not incorporated. Therefore, technical analysis treats fundamental analysis, which analyzes external factors such as company performance reports and economic indicators, redundant for making predictions.

Furthermore, behavioural finance modelling has shown that technical analysis indicators give useful indication about the psychology of the crowds that is not captured by other conventional statistical analysis, nor by fundamental analysis (Neely et al., 1997). This means, technical analysis captures the collective intelligence of the market, which may include emotions and possibly irrational behaviour.

Technical analysis performs non-linear transformations on stock market data as these datasets can be viewed as non-linear systems. This enables the use of linear learning models on the projected data in the hyperspace. Although with highly flexible non-linear models, it is possible to train a model with raw price input. TA-lib is a popular technical analysis library[1], where indicators can be grouped by the family of overlap studies, momentum indicators, volume indicators, cycle indicators, price transform, volatility indicators and pattern recognition. For the list of technical indicators, please see appendix A.2.

Typically, daily market data is used for predicting stock prices, such as the FTSE 100 price history for more than a decade. The FTSE 100 index consists of 100 companies listed on the London Stock Exchange with the highest market capitalization. Daily market data consists of low, high, open, close and volume fields, that describes the price movement of a company on a given day. Popular sources for the data includes Google Finance and the Yahoo API, though sometimes these do not take splits, dividends and mergers into account. A selection of technical analysis indicators can be seen in Figures 2.2 and 2.3. Traders use these candlestick plots as a heuristic to determine the direction of prices and suggest entry and exit points for trades.

Technical analysis techniques have been developed over many centuries of financial market data (De la Vega, 1688). Joseph de la Vega was one of the first people to describe technical analysis of the 17th century Dutch markets. In the Far East, Homma Muehisa developed candle stick charting techniques in the early 18th century (Nison, 2001). In the mid 20th century, Technical Analysis of Stock Trends was published, which is one of the seminal works of the field (Robert

[1]`http://ta-lib.org/`

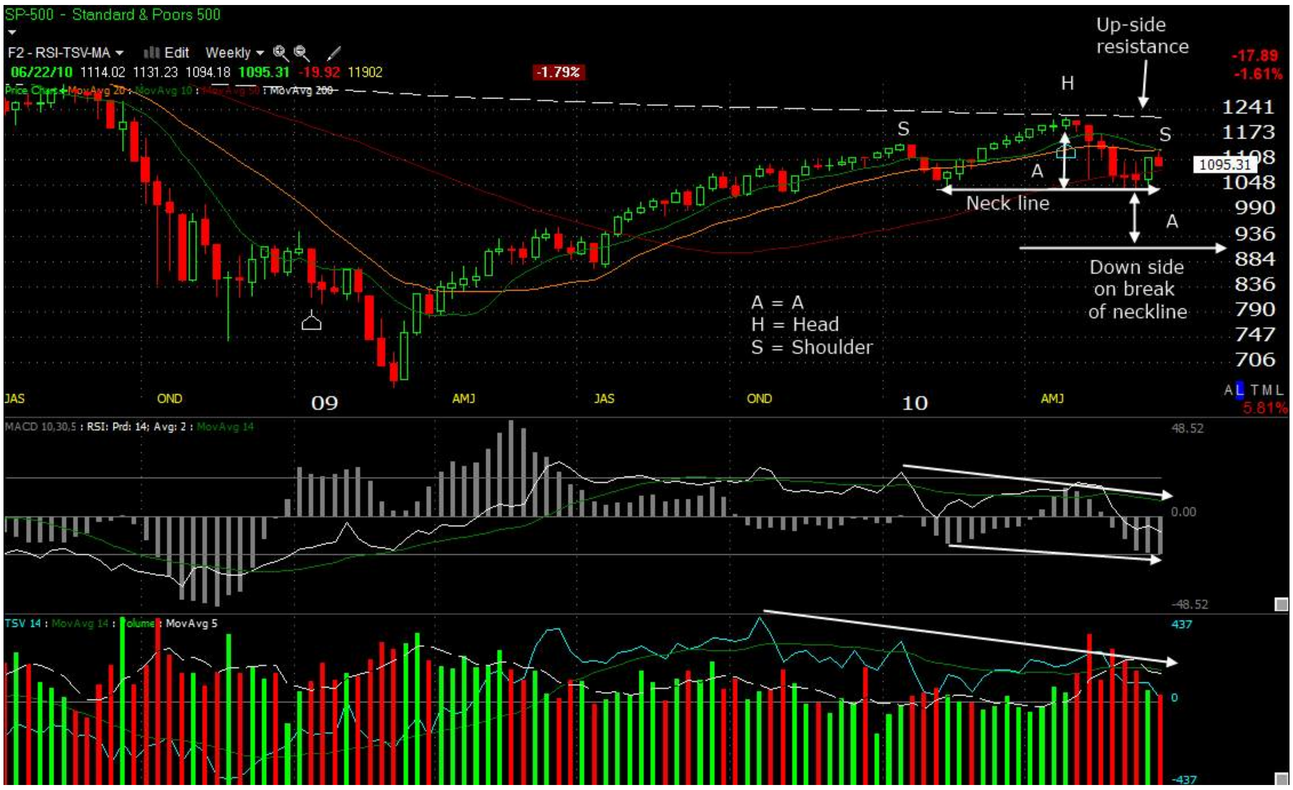


Figure 2.2: Example technical analysis chart with several patterns, including “head-and-shoulder”, manually annotated by trader. Head (H) with two shoulders (S) on either side, denoting expected price movements matching the pattern shape.

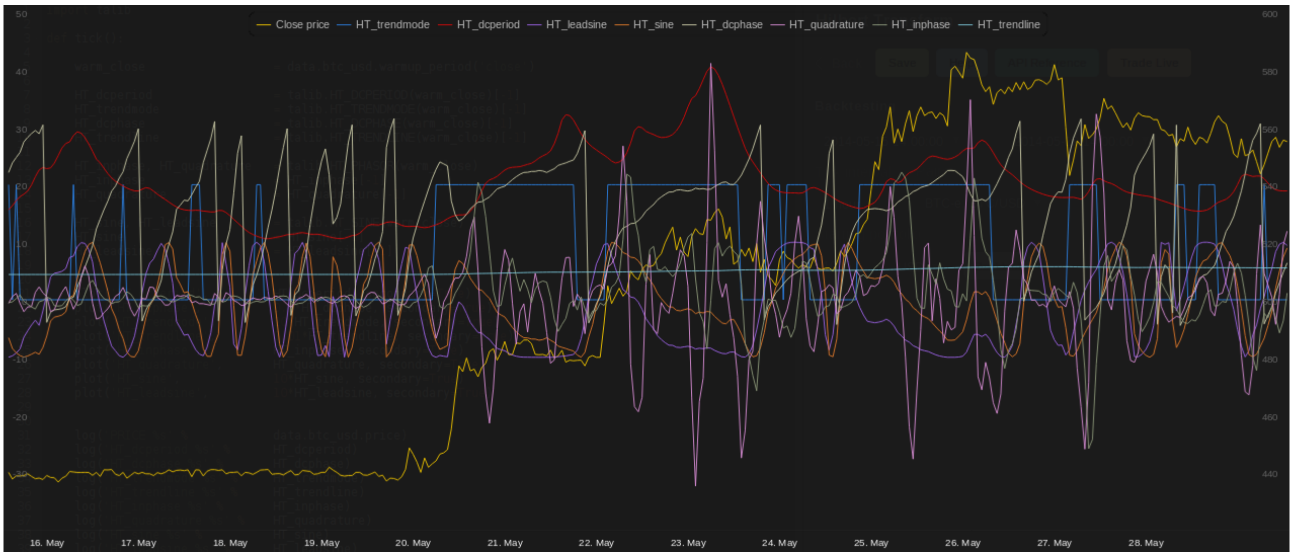


Figure 2.3: Example cycle indicators, a special case of technical analysis, that transform the price series with Hilbert Transform and its components.

and John, 1948). This work almost exclusively investigated chart patterns for trend analysis, due to insufficient computing power at the time to perform statistical analysis. In recent years, technical analysis has shifted to computerized pattern discovery, away from manually designed features.

Using charts, technical analysts seek to discover price patterns and trends in financial markets in order to exploit them (Murphy, 1999). Some example patterns are the head and shoulders and double top/bottom reversal (Elder, 1993). Moving average and looking for lines of resistance, support, channels are important. Sometimes more advanced formations, such as pennants, flags, balance days and cup and handle indicators can be seen (Elder, 1993). Other indicators are derived from mathematical transformations of price, including up, down, volume, advance, decline and other inputs. These help asses the price trend and the probability of direction of the continuation. Occasionally, indicators measure relationships between various assets and market indices. Some examples are moving average, relative strength index, and MACD. Other times, correlations are measured, for example between implied volatility in options and call/put ratios of price. Recently, various sentiment indicators have been defined, based on bull/bear ratios or short term interest rates.

Many techniques of technical analysis include candlestick charting, Dow theory, Elliott wave theory, and various combinations exist. Often times, subjective expert judgement is applied to interpret patterns and how one should trade based on them. Occasionally, strictly mechanical, systematic approach is used for pattern identification.

Fundamental analysis is in contrast with technical analysis, as it studies current economic factors affecting price, rather than historical trends. Technical analysis states that these fundamental factors are already incorporated into the price of assets, so it instead tries to uncover trends. Neither of these techniques is perfect, so trader may use one or the other or a combinations to guide their decisions (Elder, 1993).

Next, we review some works that made use of technical analysis in a machine learning system. These works are meant to provide a sample of representative research occurring in finance with technical analysis and machine learning. Note, that there are countless ways to come up with new technical analysis indicators, and there are countless way to frame them in a machine learning settings. It would be hard to find seminal or historically significant papers in this field, as likely any well working system described is kept private in order to exploit economic advantage, or it no longer works due to the signal discovered in the paper becoming public and therefore arbitraged by other investors (Liew and Mayster, 2017). That means the longer a paper has been published, the higher the chance its results would no longer hold on a contemporary dataset. Therefore, below are a few papers that can be regarded as representative of technical analysis with machine learning. They describe some possible ways of first coming up with new indicators, i.e. features in a machine learning system, creating the machine learning problem, training the system and evaluating on a held out dataset.

Technical analysis has been shown to provide useful non linear predictors for stock market prediction (Lo et al., 2000). Support vector machines and neural networks are often combined with technical analysis to model stocks (Kim, 2003). In this study Kim used a select few technical indicators related to quantifying price movement in historical data, in order to predict the directional change of an index using support vector machine with a Gaussian kernel. They showed that polynomial kernels do not work that well. The prediction performance on around 500 held out trading days ranged between 51-57%. The authors compared SVMs to neural networks and nearest neighbour type models, and showed SVM to outperform them. However, both the training and test data sizes were quite limited.

(Liew and Mayster, 2017) is an example of recent work where the author attempted to forecast exchange traded funds performance using historical returns, historical volume data and some additional categorical variables related to day of week and month. The author found that short term predictability is hard to find in the data, but long term predictability is better, i.e. over 270 days horizon. The author tried various machine learning methods, such as multi layer perceptrons, support vector machines and random forests in order to forecast the direction of price movement. It was found that actually volume indicators were quite meaningful in the long term. Transaction costs were not taken into account, nor was there any backtesting, instead the author reported ROC performance and test set classification accuracy.

(Akita et al., 2016) is another recent work that explores the option of modeling textual and numerical time series together. The authors note that previous research did not consider the interaction of these two modalities of data, nor the temporal aspects of how text evolves over time. Many works also failed to perform joint modeling of multiple companies, instead opting to model them individually. In this thesis, we address all of these issues. In addition, Akita only chose to model 10 companies at a time, while our models are much larger, modeling several hundred at the same time across constituents of stock indices. They represented news text using paragraph vectors, averaging multiple documents over a period, or inserting a zero vector where no news was available. In contrast, in our approach we max pool news articles that has the effect of filtering to key words with high predictive power. In this paper, they also predict stock price directly whereas in our approach we optimize for trading profit and predict actual trades. The evaluation period is also quite short, spanning only one year, whereas in our work we evaluate over a decade of test data. Last, the work considers the stock price prediction as a two disjoint optimization problems, first deriving the news representation, and then predicting the price. In our work we optimize the representations end-to-end. The authors do not cite buy-and-hold results, instead compare their methods to other machine learning approaches.

Although the Efficient Market Hypothesis rejects its validity, technical analysis is commonly applied to stock market forecasting. (Kim et al., 2002) measured the distributional differences of a select few technical indicators that were chosen by expert knowledge instead of automated statistical analysis. He used profit per trade as a measure to evaluate performance of a trading system with transaction cost. (Cha and Chan, 2000) proposed a system that output buy, sell and hold trading signals for stocks that were not part of the training set. In his system he extracted local maxima and minima of prices and trained a neural network to predict these points. He used a dataset of around 800 trading days with three companies on the Hong Kong Stock Exchange. He considered investing proportional to the strength of the trading signal but left the implementation of such objective to future work.

In our system, profit per trade was used as a measure of performance but we relied on automated learning methods to extract relevant information from the dataset, instead of expert knowledge. Similar to their work, we consider investment based on the predictive signal determined by a learning algorithm that invests based on the strength of the signal after squashing it through a sigmoid function. Our modelling approach is different to theirs as we do not explicitly model peaks and troughs, but consider making daily trades based on the recent historical market context. Moreover we train on several thousand data points across almost a hundred companies on the London Stock Exchange, a much larger dataset than that used in (Cha and Chan, 2000).

Auto-regressive recurrent neural networks with exogenous input have been combined with technical analysis (Ghosn and Bengio, 1997). This work investigated whether sharing hidden layers of neural networks between companies can improve the selection of high return stocks. This was framed as a multi-task learning problem (Ghosn and Bengio, 1997). Ghosn examined whether neural net parameters trained on one company can be applied to produce predictions for other stocks. He also investigated whether selective parameter sharing of various neural net layers can aid prediction. In his paper, he achieved significant market returns with a trading system that was based on this model. This shows that relationships between companies contain valuable information that can be mined for trading.

Last, statistical methods have also been successful at uncovering signals in financial data. For example, cointegration based strategies used in pairs trading, and Granger casualty analysis are methods employed to identify correlations between various assets in the markets, even though these assets may follow a random walk in their price evolution (Nelson and Plosser, 1982).

These works have shown that it is possible to forecast markets using technical analysis. Next we detail some preliminary experiments that give us suggestions as to how to incorporate technical analysis in a machine learning framework. Our research hypothesis is that markets are not completely efficient and we aim to empirically test this using predictive modelling. In this thesis we aim to demonstrate that technical analysis contains predictive power over forecasting market returns. Traders traditionally look at candlestick charting, also known as technical analysis, to predict future price movements of securities using their own expert judgement (Lo et al., 2000). In our experiments, we aim to avoid relying on expert judgement to make predictions, but rather we apply a statistical machine learning algorithm to automatically learn useful hidden patterns from past historical data. Such an algorithm can assist the trader to decide which technical analysis indicators are important for which companies in which time periods and how much predictive power they have. In addition, trading actions can be automatically executed based on these recommendations.

It is an open question whether excess profits can be achieved using technical analysis by outperforming a buy-and-hold baseline. Since markets are non-stationary and the price movements of multiple companies may be related, it is not evident what manually constructed technical analysis indicators can capture such changing cross-market relationships. We hypothesize that by automatically weighing technical analysis indicators in a sophisticated machine learning framework, we are able to uncover these relationships.

In addition, it is likewise not evident how one should actually trade based on the technical analysis output of the systems. Previously, people relied on heuristics here as well, but recently (Cha and Chan, 2000) indicated that investing based on the strength of the predictive signal after squashing it through a sigmoid function could be beneficial, but left this to future work.

In his work, Cha modelled significant peaks and troughs explicitly on a few stocks. In our work, we aim to use a much larger dataset than Cha, which comprises of the constituent companies of large indices. In addition, we aim to make regular daily trades, rather than sporadic ones, where the term “significant” may be subjective. This also gives us higher statistical significance in the system evaluation as more trading data is available. Last, we aim to maximise profit per trade and invest proportional to the predictive signal.

Some of the largest developed stock markets we study in this thesis are the FTSE100, the Nasdaq and the New York stock exchanges. We validate technical analysis on the FTSE100 dataset in Section 4.1 and construct raw learning methods from historical market data for companies listed on Nasdaq and NYSE in Section 6.2. A review of linear models used in the FTSE100 experiment can be found in Section 3.1 and a review of neural networks for the Nasdaq and NYSE experiments is available in Section 3.3. In addition, in Section 6.2 we propose a framework to automatically learn good technical analysis indicators from data without any manual expert feature engineering. This way, we can determine whether past market data is predictive of future returns, as claimed by technical analysis. In the next section, we review fundamental analysis and how we could empirically test its validity.

2.2.5 Fundamental Analysis

In the previous section, we showed that technical analysis can be useful for predicting future returns on financial markets. In this section, we examine fundamental analysis for this function. Fundamental analysis of a business involves analysing its financial statements, health, management, competitive advantages, competitors and the markets it operates in. Fundamental analysts examine dividends, new products, earnings, quality, assets, ratio, research, and other economic factors thought to affect the price of investments.

In this thesis we focus on predicting credit risk or loan interest rates for small and medium sized businesses on the debt market of a crowdfunding platform. Then we aim to demonstrate a way to exploit these predictions for risk free excess trading profits, and show how we can incorporate multiple modalities of data into our model, including structured data, user generated text posts, temporal variations while grouping related information together.

Funding Circle is an online crowd-funding marketplace.[2] It is situated in the new peer to peer lending industry that enables people to lend and borrow money from each other directly. Thus, Funding Circle allows retail investors to lend money directly to small and medium sized businesses. New loan applications can be competitively bid on in a primary market and already possessed loan parts can be traded between investors on a secondary debt market. A lifetime of a loan on Funding Circle starts with the borrower applying for a loan auction, which is then listed on the primary market for seven days. During this time, investors bid on the auction offering some money and an interest rate they are willing to lend at. When the auction ends with an average loan rate, the monthly loan repayments begin to the investors.[3] At the same time, investors can also begin trading their loan parts on the secondary market for a premium or discount, which is called markup. In the end, a loan can either default, be repaid early, or mature.

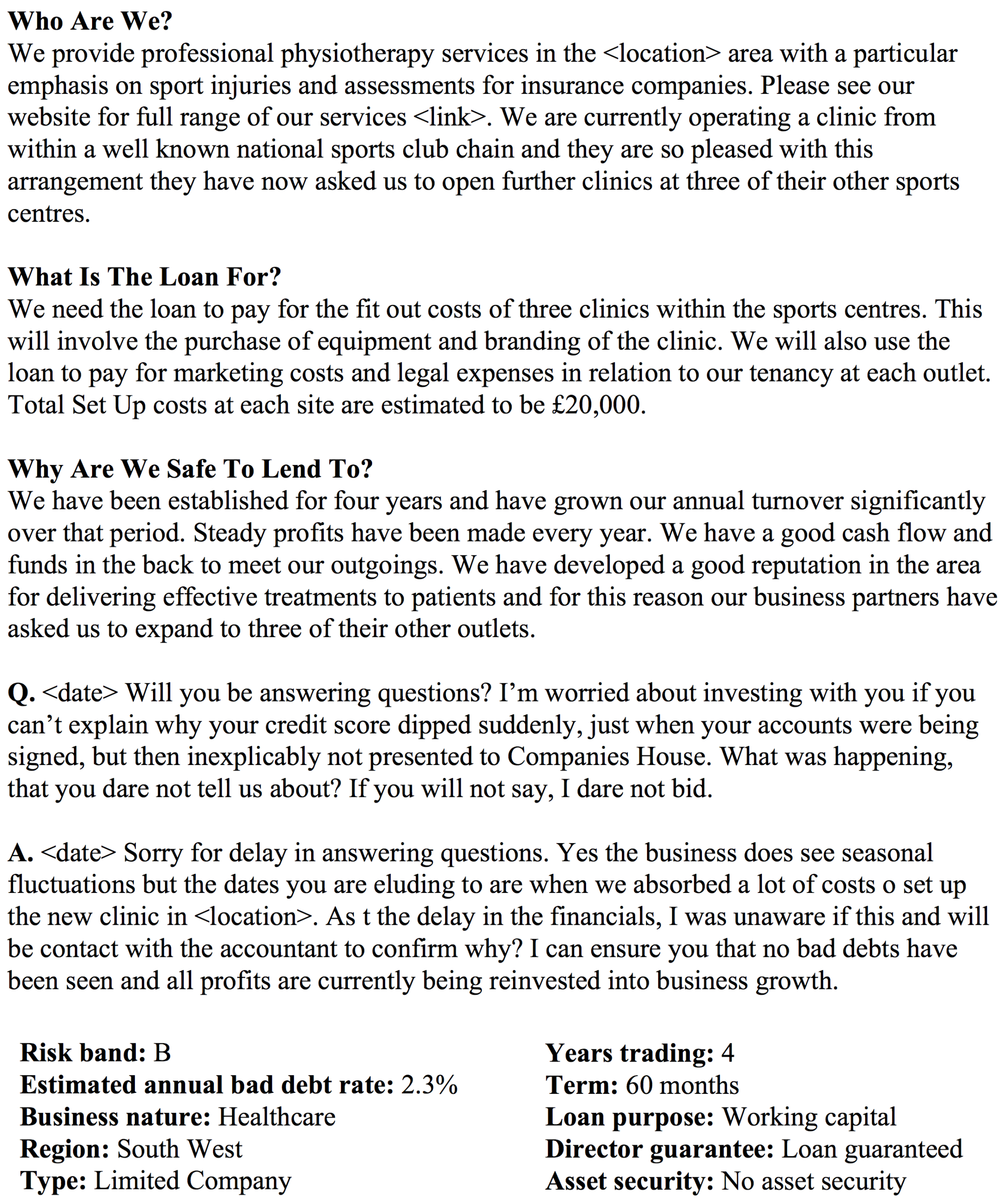
When applying for new loans on Funding Circle, users have to fill out an application form and they are subject to credit scoring. Then there is a period of conversation between potential lenders and the borrower in the form of Questions and Answers. Lenders can competitively offer money to the borrower for the period of two weeks in the form of an auction. Once the auction is closed, the borrower can choose to accept the offers made with the lowest interest rate or reject the auction. If accepted, the loan becomes live and investors can trade their loan parts with one another in a secondary market, where sales often represent a premium or discount relative to the original investment. The secondary market rate is the quantity that we seek to model, based on the rich details of the loan along with temporal factors. In the end, a loan either defaults, where investors lose their money, is repaid early if the borrower no longer needs the funds, or matures until its term length and expires naturally. Investors are repaid their capital plus interest on a monthly basis.

Each loan application contains information on the purpose of the loan and the borrower's business. Lenders and borrowers can also communicate with each other in the form of comments. In addition, the credit history of the borrower is available. This includes how long the borrower has been trading, how his past repayment performance was, what industry he is in, what the latest filed management accounts are, i.e. how much profit or loss was made in the last year by his business... etc. Other numerical data includes the requested amount of pounds the borrower

[2]`https://www.fundingcircle.com/`

[3]Funding Circle has recently introduced fixed rate loans but our study was carried out prior to this.

asked for, the number of years his business has been trading, the target rate he would like to borrow at, the estimated annual bad debt for similar businesses in his industry, the term length of the loan, the length of the auction, since the borrower can choose the accept a loan early or wait until the end of the auction.



Who Are We?
We provide professional physiotherapy services in the <location> area with a particular emphasis on sport injuries and assessments for insurance companies. Please see our website for full range of our services <link>. We are currently operating a clinic from within a well known national sports club chain and they are so pleased with this arrangement they have now asked us to open further clinics at three of their other sports centres.

What Is The Loan For?
We need the loan to pay for the fit out costs of three clinics within the sports centres. This will involve the purchase of equipment and branding of the clinic. We will also use the loan to pay for marketing costs and legal expenses in relation to our tenancy at each outlet. Total Set Up costs at each site are estimated to be £20,000.

Why Are We Safe To Lend To?
We have been established for four years and have grown our annual turnover significantly over that period. Steady profits have been made every year. We have a good cash flow and funds in the back to meet our outgoings. We have developed a good reputation in the area for delivering effective treatments to patients and for this reason our business partners have asked us to expand to three of their other outlets.

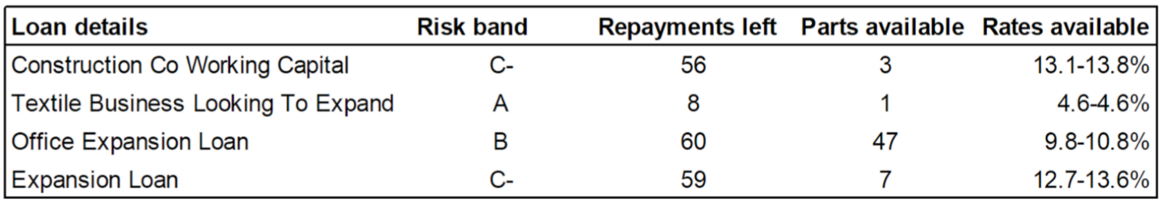
Q. <date> Will you be answering questions? I'm worried about investing with you if you can't explain why your credit score dipped suddenly, just when your accounts were being signed, but then inexplicably not presented to Companies House. What was happening, that you dare not tell us about? If you will not say, I dare not bid.

A. <date> Sorry for delay in answering questions. Yes the business does see seasonal fluctuations but the dates you are eluding to are when we absorbed a lot of costs o set up the new clinic in <location>. As t the delay in the financials, I was unaware if this and will be contact with the accountant to confirm why? I can ensure you that no bad debts have been seen and all profits are currently being reinvested into business growth.

Risk band: B	**Years trading:** 4
Estimated annual bad debt rate: 2.3%	**Term:** 60 months
Business nature: Healthcare	**Loan purpose:** Working capital
Region: South West	**Director guarantee:** Loan guaranteed
Type: Limited Company	**Asset security:** No asset security

Figure 2.4: Excerpt from a loan auction (sic, redacted).

An excerpt from the loan auction page can be seen in Figure 2.4 and a snapshot from the secondary market in Figure 2.5. The main auction page contains information about the requested



Loan details	Risk band	Repayments left	Parts available	Rates available
Construction Co Working Capital	C-	56	3	13.1-13.8%
Textile Business Looking To Expand	A	8	1	4.6-4.6%
Office Expansion Loan	B	60	47	9.8-10.8%
Expansion Loan	C-	59	7	12.7-13.6%

Figure 2.5: Snapshot of the secondary market. Each row represents a loan, showing the risk band, number of repayments left, number of loan parts on offer, and the rate distribution of those loan parts

amount by the borrower, the term length of the loan, 60 months in the excerpt, how long the borrower has been trading in years, i.e. 4 years, and the target rate at which they would like to borrow, which is tied to the risk band, "B" in the example. Additional data visible in the excerpt includes the nature of the business (Healthcare), which roughly corresponds to industry, the geographical region the business operates in (South West of the United Kingdom), the type of business (limited company), the risk band which includes the estimated annual rate of bad debt (2.3%), the purpose of the loan (Working capital), and whether there is asset security (no) or a director's guarantee (yes).

The borrower is subject to credit scoring, which takes the form of a year of monthly measurements by external agencies. The current credit score is also calculated relative to all businesses, relative to the industry the business is in, relative to businesses with similar size and with similar age as the borrower's business. Apart from the credit history, a detailed profit and loss statement of the applicant is provided. This is the annual filed management accounts of the company for the last few years. Additional information includes the amount of outstanding loans, the amount of overdraft, and the payment performance for the company and the industry.

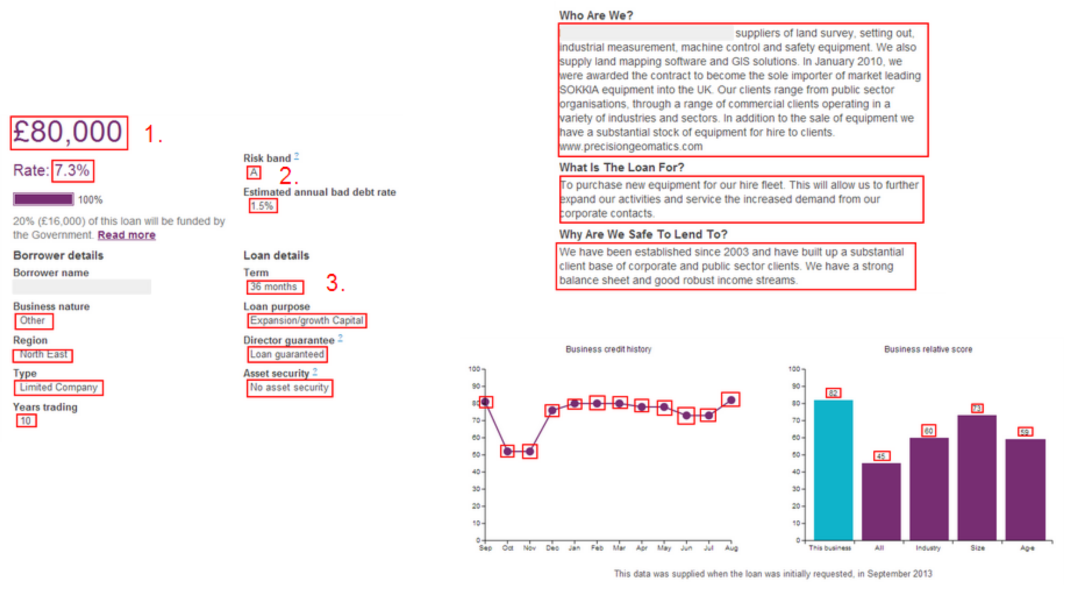
Each loan application contains textual information on the purpose of the loan and the borrower's business. This is covered by the "Who Are We," the "What Is The Loan For," and the "Why Are We Safe To Lend To" sections, as shown in Figure 2.4. This information is filled out by the borrower when they make a loan application. Lenders and borrowers can also communicate with each other through a Questions and Answers section where lenders can ask questions, which the borrower can answer.

After bidding finishes, the closing rate and the accepted bid spread of the loan is available.[4] The closing rate is the weighted average rate of all the accepted bids. Finally, we record time

[4]Funding Circle has recently introduced fixed rate auctions where there is no bid spread. This work was carried out prior to that.

dependent data, such as when the auction closed at, as well as the date we scraped the secondary market prices, and the time difference between these two. We have gathered around 3,500 loan applications between 2013 and 2014. There are around 1,000 - 1,500 words in 50-70 sentences with an average of around 6 question and answer pairs and 3 additional text fields per loan request.

Textual features may include the name of the borrower, the nature of his business, the geographical region he operates in, the type of business he runs, the risk band he has been assigned to, what the purpose of the loan is, whether there is asset security and director's guarantee with it. These features can also be used as categorical variables with one less degrees of freedom than the number of possible values. For a summary of these features, see Figure 2.6.



£80,000 1.

Rate: 7.3%

100%

20% (£16,000) of this loan will be funded by the Government. Read more

Risk band ?

A 2.

Estimated annual bad debt rate

1.5%

Borrower details

Borrower name

Business nature

Other

Region

North East

Type

Limited Company

Years trading

10

Loan details

Term

36 months 3.

Loan purpose

Expansion/growth Capital

Director guarantee ?

Loan guaranteed

Asset security ?

No asset security

Who Are We?

suppliers of land survey, setting out, industrial measurement, machine control and safety equipment. We also supply land mapping software and GIS solutions. In January 2010, we were awarded the contract to become the sole importer of market leading SOKKIA equipment into the UK. Our clients range from public sector organisations, through a range of commercial clients operating in a variety of industries and sectors. In addition to the sale of equipment we have a substantial stock of equipment for hire to clients. www.precisiongeomatics.com

What Is The Loan For?

To purchase new equipment for our hire fleet. This will allow us to further expand our activities and service the increased demand from our corporate contacts.

Why Are We Safe To Lend To?

We have been established since 2003 and have built up a substantial client base of corporate and public sector clients. We have a strong balance sheet and good robust income streams.

This data was supplied when the loan was initially requested, in September 2013

Figure 2.6: Example key loan information, description and business credit history.

We can increase the sophistication of the modelling by uncovering related information from several loans in the same industry and model how trustworthy businesses coming from different industries as a whole are. This is a form of multi task learning. Figure 2.7 shows that we can bias the learning algorithm towards industry specific information before making a prediction for a loan. For example, loans coming from the healthcare sector might be safer with lower likelihood of bankruptcy than loans in finance or construction.

Plain textual features that contain paragraphs and sentences are the "who we are" section, the "what this loan is for" section, and "why we are safe to lend to". The borrower needs to fill

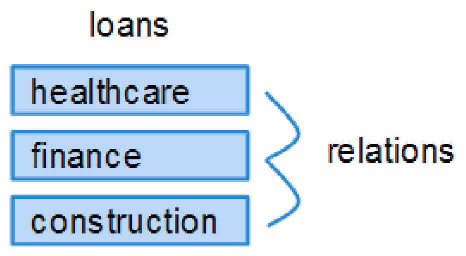
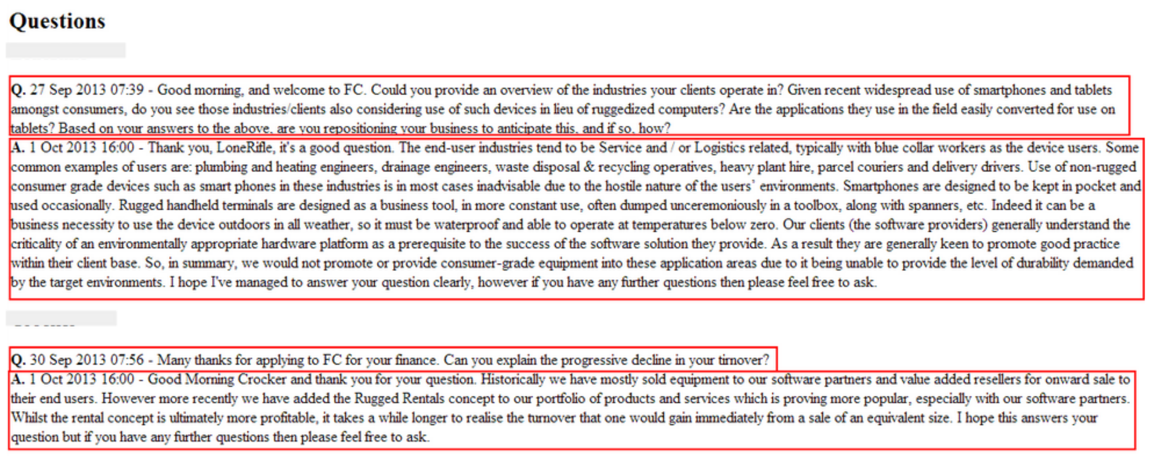


Figure 2.7: Uncovering latent relationships between industries when making predictions for individual loan applications. Pairwise relationships can be extracted between healthcare, finance, construction and other industries.

these out with at least 500 words. There is also a Q&A section where lenders can ask questions and the borrower can answer them which facilitates a conversation. This is a form of crowd due diligence, the purpose of which is to provide additional assessment of the loan application by the wisdom of the crowds. In addition to the Q&A section, crowd due diligence is also performed on other websites with regard to borrowers, where ongoing information with regard to loan applications can be acquired. A snapshot of a few comments can be seen in Figure 2.8. The content of this textual information can be processed with natural language processing techniques in order to uncover term meanings and semantic knowledge regarding the riskiness of these investments.



Questions

Q. 27 Sep 2013 07:39 - Good morning, and welcome to FC. Could you provide an overview of the industries your clients operate in? Given recent widespread use of smartphones and tablets amongst consumers, do you see those industries/clients also considering use of such devices in lieu of ruggedized computers? Are the applications they use in the field easily converted for use on tablets? Based on your answers to the above, are you repositioning your business to anticipate this, and if so, how?

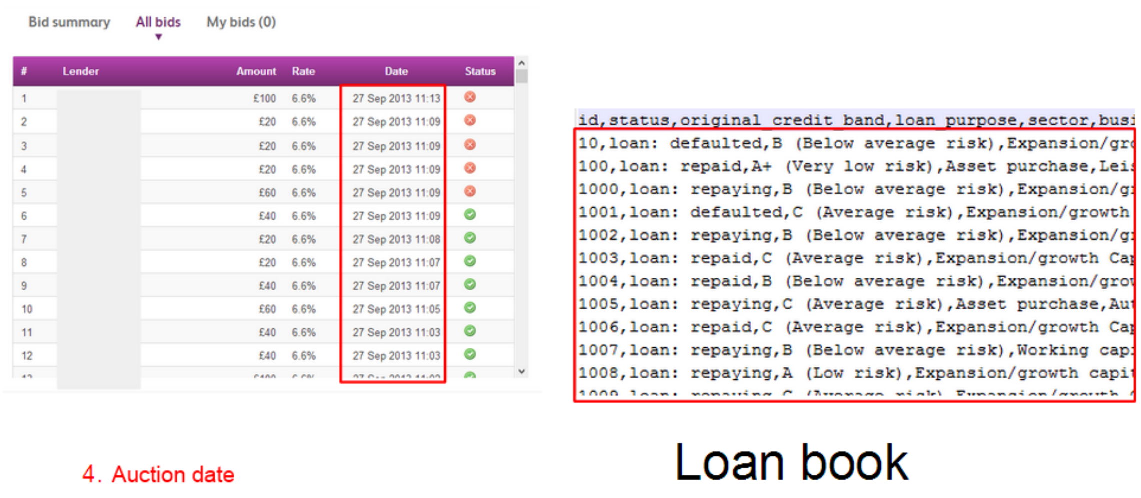
A. 1 Oct 2013 16:00 - Thank you, LoneRifle, it's a good question. The end-user industries tend to be Service and / or Logistics related, typically with blue collar workers as the device users. Some common examples of users are: plumbing and heating engineers, drainage engineers, waste disposal & recycling operatives, heavy plant hire, parcel couriers and delivery drivers. Use of non-rugged consumer grade devices such as smart phones in these industries is in most cases inadvisable due to the hostile nature of the users' environments. Smartphones are designed to be kept in pocket and used occasionally. Rugged handheld terminals are designed as a business tool, in more constant use, often dumped unceremoniously in a toolbox, along with spanners, etc. Indeed it can be a business necessity to use the device outdoors in all weather, so it must be waterproof and able to operate at temperatures below zero. Our clients (the software providers) generally understand the criticality of an environmentally appropriate hardware platform as a prerequisite to the success of the software solution they provide. As a result they are generally keen to promote good practice within their client base. So, in summary, we would not promote or provide consumer-grade equipment into these application areas due to it being unable to provide the level of durability demanded by the target environments. I hope I've managed to answer your question clearly, however if you have any further questions then please feel free to ask.

Q. 30 Sep 2013 07:56 - Many thanks for applying to FC for your finance. Can you explain the progressive decline in your tirnover?

A. 1 Oct 2013 16:00 - Good Morning Crocker and thank you for your question. Historically we have mostly sold equipment to our software partners and value added resellers for onward sale to their end users. However more recently we have added the Rugged Rentals concept to our portfolio of products and services which is proving more popular, especially with our software partners. Whilst the rental concept is ultimately more profitable, it takes a while longer to realise the turnover that one would gain immediately from a sale of an equivalent size. I hope this answers your question but if you have any further questions then please feel free to ask.

Figure 2.8: Example of comments on a loan, a form of crowd due diligence.

Time series features include the monthly credit score of the business, the relative score to the industry as a whole, and to the whole market, the score of the industry, and a score relative to other business with the same size and age, visible in Figure 2.6. However, temporal information can also be included from the date the auction was released on the primary market, which enables us to model changing non-stationary market dynamics. This date information is available from the bidding log of individual users, as seen in Figure 2.9.

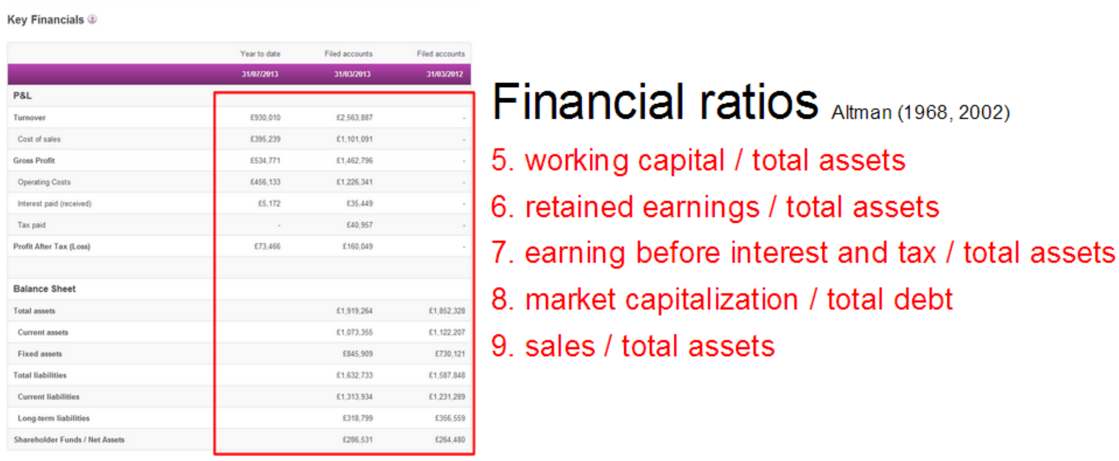


#	Lender	Amount	Rate	Date	Status
1		£100	6.6%	27 Sep 2013 11:13	✗
2		£20	6.6%	27 Sep 2013 11:09	✗
3		£20	6.6%	27 Sep 2013 11:09	✗
4		£20	6.6%	27 Sep 2013 11:09	✗
5		£60	6.6%	27 Sep 2013 11:09	✗
6		£40	6.6%	27 Sep 2013 11:09	✓
7		£20	6.6%	27 Sep 2013 11:08	✓
8		£20	6.6%	27 Sep 2013 11:07	✓
9		£40	6.6%	27 Sep 2013 11:07	✓
10		£60	6.6%	27 Sep 2013 11:05	✓
11		£40	6.6%	27 Sep 2013 11:03	✓
12		£40	6.6%	27 Sep 2013 11:03	✓

```
id,status,original_credit_band,loan_purpose,sector,busi
10,loan: defaulted,B (Below average risk),Expansion/gro
100,loan: repaid,A+ (Very low risk),Asset purchase,Lei
1000,loan: repaying,B (Below average risk),Expansion/g
1001,loan: defaulted,C (Average risk),Expansion/growth
1002,loan: repaying,B (Below average risk),Expansion/g
1003,loan: repaid,C (Average risk),Expansion/growth Ca
1004,loan: repaid,B (Below average risk),Expansion/gro
1005,loan: repaying,C (Average risk),Asset purchase,Au
1006,loan: repaid,C (Average risk),Expansion/growth Ca
1007,loan: repaying,B (Below average risk),Working cap
1008,loan: repaying,A (Low risk),Expansion/growth capi
```

Figure 2.9: Log of individual bidding history of users and aggregate loan book information.

A number of crucial financial indicators can be extracted from the annual filed management accounts of the companies. Plain indicators include the amount of outstanding loans, the amount of overdraft, the payment performance for the company, industry and what the trend is in this regard. For a snapshot of this information, see Figure 2.10. Other loan properties can be extracted from the loan book directly, as seen in Figure 2.9


Key Financials

	Year to date	Filed accounts	Filed accounts
	31/07/2013	31/03/2013	31/03/2012
P&L			
Turnover	£930,010	£2,563,887	-
Cost of sales	£395,239	£1,101,091	-
Gross Profit	£534,771	£1,462,796	-
Operating Costs	£456,133	£1,226,341	-
Interest paid (received)	£5,172	£35,449	-
Tax paid	-	£40,957	-
Profit After Tax (Loss)	£73,466	£160,049	-
Balance Sheet			
Total assets		£1,919,264	£1,852,328
Current assets		£1,073,355	£1,122,207
Fixed assets		£845,909	£730,121
Total liabilities		£1,632,733	£1,587,848
Current liabilities		£1,313,934	£1,231,289
Long-term liabilities		£318,799	£356,559
Shareholder Funds / Net Assets		£286,531	£264,480

Financial ratios Altman (1968, 2002)

5. working capital / total assets
6. retained earnings / total assets
7. earning before interest and tax / total assets
8. market capitalization / total debt
9. sales / total assets

Figure 2.10: Key financial statement information and calculated financial ratios.

The target of the model is to predict the secondary market ongoing buyer rate of loans, which is defined as the top of the ask order book, i.e. the highest rate currently on offer for the loan among all the offers. As Funding Circle does not provide a bid side of the order book, or historical trade data, this is the closest indicator of market judgment, where a lower rate

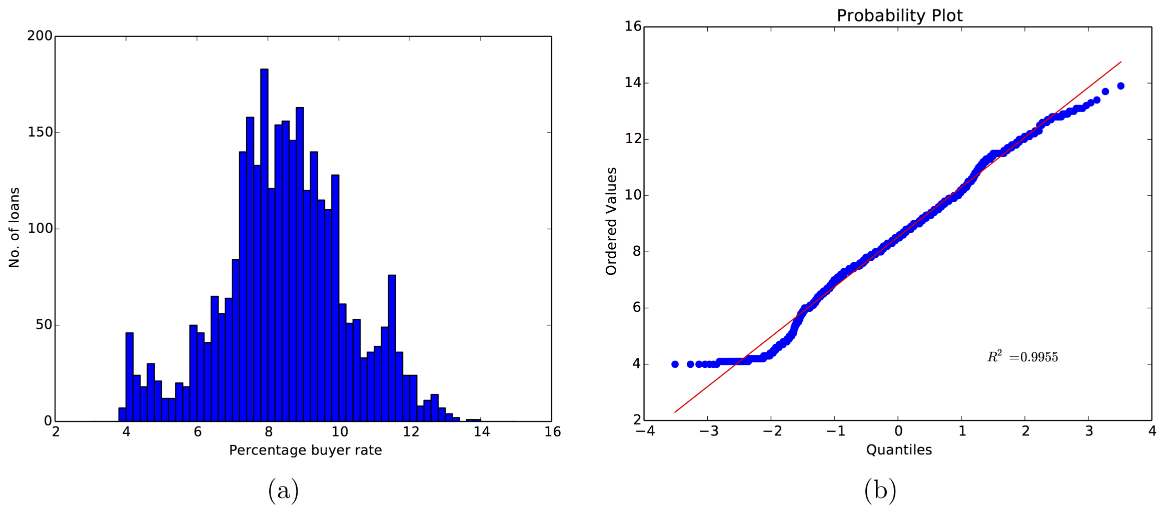


Figure 2.11: Distribution of buyer rates, showing a histogram over the rates (a), and a normal probability plot (b). The red line in (b) illustrates perfect normality, and the data points are plotted in blue.

means less risky. This can deviate significantly from the rate the loan closed at, as many market participants only buy exclusively from the secondary market and do not participate in the primary market. In addition, as the loan matures, it generally becomes less risky with most defaults happening towards the beginning-mid section of the loan duration. Another target can be the markup (premium or discount) applied to loans.

Examining the distribution of secondary market buyer rates in Figure 2.11 we see that it passes the nearly normal condition. The dips in the -2 and +3 quantiles give evidence of risk aversion and risk seeking behavioral biases (Kahneman and Tversky, 1979), where loans with the lowest and highest rates are unusually in high demand. Below quantile -2, we see the effect of a hard minimum limit imposed on the market place at 4% with some values below that due to rounding error on the market server. Additional spikes can be observed at the minimum bid rates, i.e. quantile 1.5 - 11.5%, where a market service called "autobid" automatically buys loan parts with a 0% markup. The target ranges between 4-15%.

One of the most well known methods of bankruptcy prediction for small businesses is the Altman Z-score (Altman, 1968). Altman applied multiple discriminant analysis to several financial ratios to provide a metric to evaluate small loan applications. He then performed bankruptcy likelihood prediction with a form of multivariate linear regression. Altman created the z-score which shows that it is possible to predict the bankruptcy of a firm with 72% confidence two years prior to the actual bankruptcy. He validated his results with t and F scores and rejected his null hypothesis. He noted that after a two year horizon, the model does not perform well, while for the present time, it can predict by up to 94% accuracy. The Altman z-score plus is an updated version of the original formula. With a neural network prediction model, the accuracy improves to 85%(Atiya, 2001). These models can be used to predict default rate for companies, and credit risk in small scale loan applications, like in the Funding Circle peer-to-peer online debt market, which is one of the datasets we examine in this thesis.

For the likelihood of default of a small business, the following fundamental analysis features or indicators have been constructed in previous work (Atiya, 2001). These are called the Altman ratios:

- Working capital / Total assets
- Retained earnings / Total assets
- Earnings before interest and taxes / Total assets
- Market capitalization / Total debt
- Sales / Total assets

Additional ratios mentioned in other literature include the following (Ravi Kumar and Ravi, 2007).

- Book value / Total assets
- Cashflow / Total assets
- Rate of change of cashflow per share
- Gross operating income / Total assets
- Return on assets

Extra information can be also extracted from the equity markets (Atiya, 2001).

- Rate of change of stock price
- Rate of change of cashflow per share
- Stock price volatility

In the Funding Circle data, the Altman terminology corresponds to the following:

- Sales = Net sales = Turnover
- Total assets = Total assets
- Working capital = Current assets - Current liabilities

- Retained earnings = Profit after tax
- Earnings before interest and tax = Gross profit - Operating costs
- Book value of total liabilities = Total liabilities
- Market value of equity = Turnover * BizStats data or Net assets
- Shareholders funds / Equity = Net assets

One peculiarity of analysing whether it is worth lending capital to small and medium sized businesses is that these businesses are not listed on the stock market, therefore it is difficult to determine the worth of the company since the market does not make a judgement. There exist several rule of thumb methods however for specific industries in order to provide private company valuations.[5] Alternatively, the worth of business can be computed from its net shareholders' assets.

We use the Funding Circle dataset to study fundamental analysis of small medium sized businesses and determine how much predictive power this has on interest rates and credit risk at which the borrowers can obtain funding and at which loans are traded on the secondary debt market. We aim to show that this data contains predictive power for forecasting interest rates and excess risk free returns can be realised with statistical arbitrage. For further information about the these experiments please refer to Section 5.2. In addition. Section 3.2 presents a review of the probabilistic Gaussian Process framework used in this study.

As we have seem. in peer to peer lending, rich data is presented to investors in order to allow them to make informed decisions. Having a way to combine several disparate features is important because in our modelling problems, in common with many others, we have rich structured data in addition to unstructured text. For example, (Joshi et al., 2010) predicted movie revenues not just from the text of critic reviews, but also used data such as genre and rating of the movie, showing that the combination of both inputs provides the best performance. The complementarity of structured and text data was also observed in financial market modelling (Lerman et al., 2008) and predicting Ebay auction outcomes (Resnick and Zeckhauser, 2002). For this reason, we seek to combine text with structured data in modelling P2P loan rates, proposing several kernel combination methods combined with Bayesian model selection.

Besides unstructured text and structured numerical data, additional information can be included by examining the time dependency of data. Financial market dynamics are known to

[5]BizStats - `http://www.bizstats.com/reports/valuation-rule-thumb.php`

change considerably with time, such as for Ebay auctions in (Ghani and Simmons, 2004). Previous work has adopted sliding window or time bucketing for training in order to prevent past data from unduly influencing future predictions (Kogan et al., 2009; Lampos et al., 2013; Wang and Hua, 2014). In contrast we take a more principled approach by explicitly modelling time using several kernels over temporal features, such as the date and the time left on a loan. These allow modelling of smoothly varying changes with time, while learning the importance and rate of change of each temporal factor.

2.2.6 Complex Systems

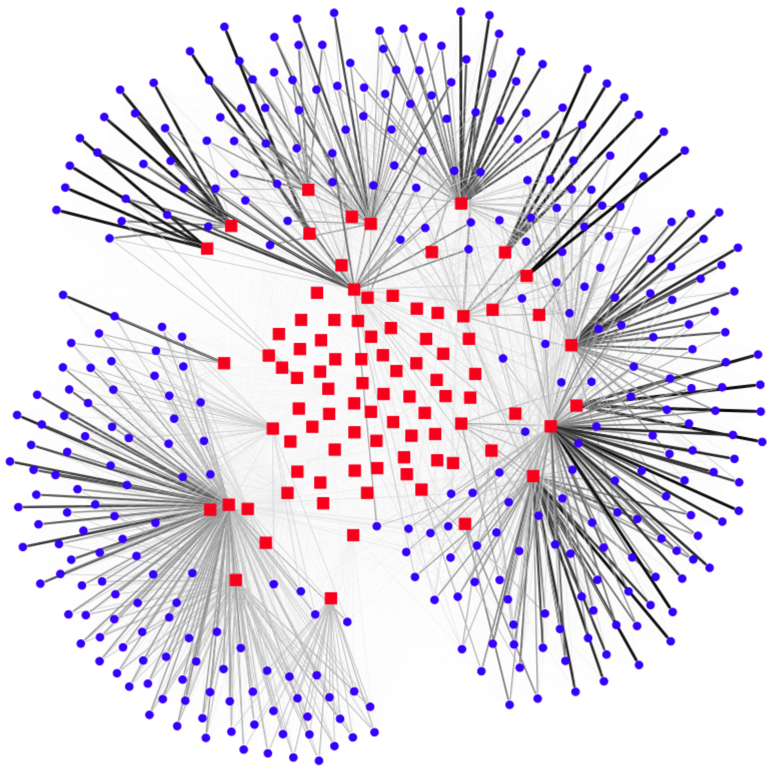


Figure 2.12: One way to model complex systems is to simulate local behaviour of agents with different initial conditions, leading to the emergence of highly non-linear outcomes.

Sornette (Sornette, 2009) examines a specific type of market inefficiency related to large crashes in price. He regards the stock market as a complex system that exhibits highly non-linear behaviour. Although it may be robust to certain variations in the environment, in some dimensions it may be extremely sensitive to a change in conditions. At any point in time, slight perturbation in the system can cause a collapse, which can be characterised as a singularity event. He argues that these sensitive aspects of the system is what leads to crashes.

Sornette claims that following the dynamics of complex systems, markets exhibit fundamentally different behaviour when they are forming a bubble, or when they are trading sideways. While prices may normally follow a random walk, during bubble forming activity they follow what he describes as log periodic oscillating pattern. This pattern keeps amplifying itself via a self reinforcing behaviour, and may even accelerate faster than exponential in some cases, moving towards a critical time when the system collapses. At this point, the underlying process is no longer sustainable and divergence occurs as a finite time singularity is reached.

These patterns may explain some psychological biases in stock market participants, such as herding. This happens when limited information exchange of market agents is propagated via local connections and communications, which enables traders to influence their neighbours in their social network at large. Other crowd effects include panic selling, imitative trading and further behavioural cues. He notes that some of these behaviours may in fact be selfish and rational, but still predictable to a high degree.

He provides evidence for the existence of such patterns via statistical hypothesis testing. He takes some historical crashes, such as the 1987 crash and the 1929 crash, and fits various log periodic functions on the historical price data. He also experiments with modelling reverse crashes where a bubble deflates following a log periodic pattern. In other experiments, back testing and forward testing is performed on price data from multiple markets. In many cases, he was able to predict a crash accurately.

Sornette provides multiple examples of additional historical bubbles followed by crashes, such as the tulip mania or the South Sea bubble. He notes that these type of patterns are also present in other fields, for example in fractals or in nature. Finally, he alludes to an upcoming global crash occurring around the year 2050, by which date larger economic and population growth will halt.

Sornette's work is a good example of how relatively simple univariate statistical model can predict stock market evolution. His model involves non-linearity and is fitted on price history. On the other hand, a general purpose machine learning learning algorithm, such as a deep neural network or Gaussian process, can model arbitrarily complex patterns with arbitrarily many input and output variables due to the Universal Approximation Theorem (Cybenko, 1989). This system is likely more flexible than the above univariate statistical model, and likewise could be successful in forecasting markets. In Gaussian processes, various periodic and log periodic patterns can be explicitly encoded with kernels. With neural networks, recurrent and convolutional structures may be of use. Although the large capacity of such models may be of concern, in practice overfitting can be effectively avoided (Kawaguchi et al., 2017).

While Sornette applied a log periodic model to predict deflation of bubbles, he did not apply it to predict abrupt upwards jumps in prices, observed in some markets. Another downside of his technique is that he has to first manually categorize crashes by selecting the degree of price movement. Though the author alludes to the possibility of having this hyper parameter tuned automatically. In addition, another way of fitting his model is to identify anchor points in the price movements by eye which requires manual labour. A fully automated machine learning approach in this case would be more general, as these models can be trained to automatically identify changes of regimes in price movements. We note that some machine learning models deal explicitly with change point detection (Adams and MacKay, 2007).

This concludes our review of technical and fundamental analysis, with a detour to complex systems. We have seen evidence that both of these techniques are useful for predicting financial markets. In the next chapter, we progress to a review of machine learning frameworks, some of their recent achievements in various domains, and how they may be adapted to our financial prediction task. However, before that we address some potential concerns related to the relevance of EMH in today's world.

2.2.7 Is EMH Still Relevant Today?

As the Efficient Market Hypothesis was proposed several decades ago and there has been extensive research in behavioral economics in recent years to identify market inefficiencies, it would appear that the fact that EMH is wrong has been settled and the matter is considered trivial.

This is supported by the existence of the multi billion dollar hedge fund industry, with firms such as Winton Capital and the Man group. These firms make huge profits by relying on the fact that the EMH is wrong. On the other end of the spectrum, other firms such as Vanguard have also grown substantially in recent years, and they propose the use of index funds or passive management that relies on the EMH being true. If markets were inefficient, then it would be irrational to invest in index funds, whereas if markets were efficient, no one would want to invest in active portfolio management. Therefore, how can both of these seemingly contradictory facts about markets be true?

It is evident that some of the top performing hedge funds in the world turn out to be the worst performing ones in the future, as evidenced by commonly used disclaimers stating that "past performance is not indicative of future returns". It is very common that certain money making strategies abruptly stop working in many hedge funds, while the funds continue to take fixed fees. In addition, some top hedge funds also turn out to be mere facades for elaborate Ponzi schemes, i.e. Bernard L. Madoff Investment Securities LLC. A study measuring fund performance of top hedge funds in the second part of the 20th century showed that many of them actually underperformed the market by their added expenses (Jensen, 1968). Just because hedge funds exists, that is not a proof that they beat the market, similar to the way that just because casinos exist, that is not a proof that gambling is a reliable way to make more money. On the other hand, investing in index funds may have appeared to be a good strategy historically, since on average it resulted in double digit yearly returns for much of the 20th century. Despite this, many index funds actually lost money during the first decade of the 21st century and even underperformed many hedge funds during the market crashes of 2003 and 2008. So what should the sophisticated concerned investor do in this case? Should they pursue an active or a passive investment strategy? Unfortunately, where there is any amount risk, there is the possibility for gains and losses, and any single historical realization of events will result in one or the other. Although the EMH was discovered many years ago, and recent research has shifted away from it, its basic tenets are still relevant today, which can be summarized as follows:

If markets were predictable, then traders would exploit this predictability, which would then make the markets unpredictable. At the equilibrium, markets will converge to unpredictability, while when they are disturbed, they may exhibit predictable patterns that can be exploited by certain participants that are quick enough for the duration until other participants catch up and make the potential profits erode. Therefore, the only way to beat the market is to discover and exclusively exploit a signal that is not exploited by anybody else, i.e. secure a monopoly over a segment of the market.

The discovery and validation of such signal is dependent on the cost of doing the necessary analysis. Therefore, to people where the cost is larger than the potential expected gain acquired from the signal, the markets appear efficient, whereas to others where the cost is smaller, the markets appear inefficient. This is how markets can be both efficient and inefficient at the same time. Costs may include personal time, labour, transaction costs or management fees among others. In this thesis, we attempt to identify signals for inefficiency by using advanced machine learning models that decrease the cost of analysis, as it would have been prohibitively expensive to carry out such work many years ago with less sophisticated hardware and computational resources. We aim to make use of our best tools to claim that these signals have indeed predictive power with a certain degree of statistical significance, but we should note that nothing substitutes for a real long term trading system and past performance may not necessarily hold true for the future. Markets are extremely stochastic and noisy in nature, combined with constantly evolving non-stationary dynamics, therefore, apparent signals may or may not work reliably in practice.

Note that this statement of fact above is nothing special, only pure economics, where competition prompts technological advancement in the pursuit of increased marginal returns, which over time erode as competition catches up, and makes the market more efficient. A similar phenomena occurs when one attempts to scale up the exploitation of inefficiencies but encounters reduced return on investment. This decrease in available exploitable inefficiency and profits is an example of diminishing returns. On the other hand, having a way to protect said profits via exclusivity can suppress the effect of free market and can create lasting advantage via temporary monopolies. In this thesis, by researching new ways to identify signals, we discover new markets for inefficiencies and lower the barrier to entry for potential new participants.

What this means is that the reader should not regard this thesis as a work on financial theory and solving the Efficient Market Hypothesis debate. Instead, this thesis should be regarded as a contribution to the computer science data mining field that aims to push the boundaries of signal processing and pattern discovery, thereby uncovering potentially previously unseen inefficiencies in financial markets that may be converted to economical gains. This way, the thesis aims to give evidence for where inefficiencies may exist and which under the right circumstances, could potentially be exploited to make excess profits without excess risk. The financial theory covered in this chapter is used as a way to set the context of the study, but it is not the purpose of it. It is not recommended that the work in this thesis is used out of the box as a get rich quick scheme, likewise it not recommended that the work in this thesis be dismissed out of hand.

Instead, we advocate critical thinking and urge the reader to observe the context of each study, weigh the evidence presented carefully, and make their own judgements, using the thesis as a blueprint to guide their thinking.

2.3 Experiment Setups

In this section, we give an overview on the datasets used in the thesis and the various modeling objectives and evaluation methods employed in our experiments.

2.3.1 Datasets

In this section, we give a brief introduction of all the datasets used in this thesis. This thesis makes use of several datasets in order to investigate various ways of uncovering inefficiencies in financial markets. We focus on both established developed financial markets, such as major stock markets in the UK and USA, as well as emerging markets and alternative investments, such as the Funding Circle peer-to-peer lending marketplace. In addition, in order to further investigate the effectiveness of text based modelling, we also make use of movie review dataset that can be used to predict box office revenue of movies for exploitation in prediction markets.

We first make use of historical dataset consisting of open high low close and volume information of companies that were part of the FTSE 100 index between 2000-2013. This covers more than a decade of data, and it is used to test and evaluate the effectiveness of our first proposed trading algorithm that aims to predict future profits based purely on price history of companies. This dataset includes slightly less than 100 companies due to some companies having left the index in the mean time, and the evaluation is performed on all companies spanning more than a decade of test data, in a sliding window manner. A recent 2-year section of this dataset is also used to test the effectiveness of Bayesian treatment of learning model and to quantify uncertainties in the modelling and predictions. However, due to the fact that this step makes computations a lot more expensive and requiring more resources, only a recent subset of the FTSE 100 dataset is used for evaluating these family of modelling techniques.

Second, we make use a different historical stock market dataset that was collected between 1994-2012 and consists of the largest USA based companies listed on the Nasdaq and New

York stock market exchanges. This include historical constituents of the S&P 500 index, with only a few companies missing from the dataset. This dataset is used to test the effectiveness of evaluating an end-to-end trading and learning agent in the last chapter. In addition to this dataset, we have additional datasources that includes historical news articles released by seven major international English language publishers. This dataset includes prevailing market sentiment of the public at the time. In addition, we also have historical economics data published by the government of the United States and the Federal Reserve Bank that includes information regarding consumer price index and various term bond yields. We make use of all this information, including news data, economics data and historical price and volume data to simulate investment decisions in a real time manner.

For the alternative investment market, we evaluate on the Funding Circle dataset. This includes historical loan description on the peer-to-peer lending platform and has information about the borrower, their credit rating history, credit risk analysis and a personal statement about what they plan to do with the loan. We also scrape a snapshot of the secondary market of Funding Circle that contains information related to how various maturing loans trade, at what rate the best loan parts are offered, and whether any loans lent to small and medium sized companies or for property development have defaulted, repaid early or matured. The purpose of this dataset is to show that fundamental analysis works and thus we can prove that we can accurately forecast the trading rates of various unseen loan purely based on fundamental data. To account for changing market condition, we also make use of temporal information as to how the loans evolve over time. As an addition to fundamental and temporal data, we scrape social media information, such as comments made by investors on the loan, questions asked by investors and answers submitted by the borrowers displaying crowd due diligence, and the aforementioned personal statement of the borrower. We aim to quantify uncertainty in predicting loan rates using Gaussian Processes. All together, there are several thousands of loans scraped for this experiment.

Finally, we make use of a movie review dataset. This contains articles released by major movie review sites, written by movie critiques privy to a private screening of the movies before they are released to the public at large. The aim of this dataset is to measure that it is possible to forecast the opening box office revenue of films released in the USA and elsewhere in the world based on the initial impressions of these critiques. The challenge is that most information is available in unstructured text format. Successfully extracting information from this source of data could make additional expensive manual processing of the data into structured format redundant. It can also be used to measure the sentiment of reviewer which can be used as a proxy of public sentiment as our experiments show. Many of the reviews are long text documents containing several paragraphs and thousands of words. In addition to the large amounts of text, there is also meta data available about the movies or reviews such as on which site the review was released and what famous actors play in a movie, or what the movie genre is. This helps with the prediction task of forecasting movie revenues as well as it enables us to measure how effective the text based algorithm is compared to one based on meta data. Finally, we can measure interaction of both meta and text data. Our dataset is split temporally in that we use recent movies for testing the effectiveness of the algorithm, and train on earlier ones. Therefore, our results accurately show how well our algorithm could perform on predicting future box of revenue of movies released in the future based on preliminary reviews, which makes the prediction task realistic and potentially exploitable in real financial prediction markets.

Of course, there are many more financial markets where we can measure the effectiveness of our proposed solutions, and the goal of this thesis is to provide a sample of diverse set and type of markets that have non-trivial amount of trading volume to showcase the existence of significant risk free profits in these markets. We also aim to measure the effectiveness of our algorithms on different types of data, such as temporal, structured / meta or unstructured / text, which we again achieve by focusing on these datasets mentioned above. Our final goal is to cover a diverse set of periods over time. This may contain several years or several decades of historical data where that is available, i.e. for the stock prediction experiments.

Once we have selected the datasets, we need to define suitable mathematical objectives to extract patterns from the data, reviewed next.

2.3.2 Model Objectives

As we have detailed above in the section on datasets, we aim to give a diverse overview of markets in order to evaluate the effectiveness of our algorithms. We aim for diversity in the datasets and also in defining the objectives of our models. There are many ways to formulate artificial intelligence prediction tasks that offer various trade off. For overview of trade-offs of various frameworks, please refer to section 3.4.

The first model we propose is a linear model. Linear models can be very robust to noise and their modelling architecture is simple. The output is simply a weighted combination of the inputs. However, in linearity lies their weakness as well. This means the inputs must already capture meaningful predictors of the output on their own, and modeling interactions of input variables is quite difficult. The latter is achieved by non-linear models. In proposing our first linear model, our goal is to align the modeling objective with the prediction task we really care about. It might be evident to predict the price change directly as a regression task or as a classification task relative to the last price of an asset, however this may not always result in the highest possible profit that can be achieved on the market. That is because every model is incomplete and has errors. Therefore, we aim to structure our model in a way that actually maximises the profit generated over time, and thus we propose to optimise for this profit measure directly. This way, our model aims to maximise the right metric we care about and hopefully make errors where this metric is not that significant, i.e. the profit potential is not that large. We need to do this because every model has limited computation power available to it with limited amount of data, and thus finding the right optimisation metric influences results a great deal.

However, as we saw above, although we have successfully changed the objective of the model to profitability, away from regression of classification, we still only have a linear model. If we were to model the entire underlying structure of developed financial stock markets, then likely a linear model is not powerful enough. Therefore, in the next step, we aim to incorporate more fine grained modeling techniques. For example, rather than training one big linear model for the entire market, we split the market into smaller segments and we fit a piecewise linear model individually on each segment. This way, our model only needs to explain the dynamics on a small segment of the market, which may be much simpler than the full market as a whole. We define our segments as per companies, i.e. we fit a linear model per company, and also over time periods, which is one month. This way, we do not need to worry about temporal dynamics changing over time. However, we hypothesize that there is a relationship between these segments, so we enable our model to allow for information flow between these tasks using multi-task learning. For the companies, we hypothesize that each company has its own dynamics but they all broadly follow an overall average market process. For the temporal buckets, we hypothesize that although the dynamics may change over time, they likely do that smoothly so subsequent buckets in time will be broadly similar.

A final objective of the models is to show that technical analysis effectively can preprocess the data and contain useful information regarding the prediction task. That means, we can summarize our datasets better with technical analysis than with just feeding plain historical returns into our model.

In the next chapter of the thesis we focus on uncertainty in our models. That is because while in frequentist models, all uncertainty is ignored and the models always make a certain prediction, incorporating uncertainty and balance of viewpoints likely improves the results. We first do that by training our linear model not just with one set of weights, but with an entire probability distribution of possibly good weights. This way, we can sample from this probability distribution when making predictions, which should make our predictions more robust to noise.

Next, we make our models non-linear by incorporating and extending the Gaussian Process framework. GPs allow for non-linear prediction via kernels, such as the Radial Basis Function kernel. This means, we can cluster datapoints in a semantic space and use neighbouring datapoint to predict for unseen data. This makes the learning algorithm much more powerful as it is capable of following the complex structure and manifold of our data that describes the real world relationships we are trying to model. GPs can also model noise very effectively due to defining a probability distribution of possible dynamics or functions that are modified or updated as training data is observed. We can use this uncertainty to quantify expected profits on various investment opportunities and selects the ones to invest in among multiple ones, i.e. maximising profit from arbitrage opportunities. In GPs we can also incorporate multiple data sources by defining a kernel over each of them and then combining these kernels together. We aim to predict the secondary market rates directly with regression.

In the final section, we explore another way to extract interaction of various datasources and information in the data, by composing hierarchical deep representations that progressively become more complex until they can predict the task very accurately. First, we explore this family of techniques on predicting movie revenue from critique reviews and meta data. We train a convolutional neural network to extract local textual information in the documents with a few neighbouring words and assemble higher levels of semantic meaning of the review progressively, i.e. to capture the entire information content of the document. Then we predict the movie revenue directly with a linear layer, and optimize for minimising the deviation from the predicted revenue compared to the actual revenue. This captures interaction of meta and textual data.

To take this further, in the last chapter we feed textual news articles data, economic indicators and past price and volume history of the entire market into a neural network. This can evolve over time and output limit order trades directly on the market with predicted prices and amount for each asset traded. Then the objective becomes to simply maximise the net worth of the simulated trader over the entire simulation period. This actually directly simulates the entire trading process of a human trader, from reading raw data coming from various sources, reasoning over it and updating beliefs over time, keeping track of a balance and stock inventory and placing trades on the market directly. The reason we say that this is an end-to-end trader is because the entire process of trading is encompassed by the machine learning model and learned from data. There is no human engineered rules involved. In previous models, machine learning only consisted of a small part of the trading problem, with additional human defined heuristics augmenting the rest to create a working trading software. We aim to avoid heuristics.

Defining the training objective is a critical component of any machine learning system, which is tightly tied into the evaluation of the system, which is reviewed next.

2.3.3 Model Evaluation

So far we have seen a sample of datasets used and the training objectives constructed on a high level. Next, we detail how we can evaluate on a test dataset the system.

Our evaluation metrics of all our models are simple. In the first stock market experiment, we simulate the balance of the human trader by taking the prediction and using the signal to shift the portfolio to either long or short position. We keep a separate balance for each market we trade on and assume we start each day with a fresh £1 balance. We evaluate our models over long periods of time and also simulate a 0.6% transaction fee. We record the overall accumulated balance at the end of the simulation, which if greater than £1, means that our algorithm has made a profit. We evaluate on future movements of price (i.e. relative price changes) compared to the last observed closing price. We assume we can trade at the closing price and our trades do not have material impact on the market, which for large markets and small budgets is a reasonable assumption. For further evaluation, we also inspect what data sources contribute to our predictions the most, i.e. which technical analysis indicators contain the most power, and how our models change over time or across markets, plotted in a graph of learned weights. All trading simulation calculates conventional fund performance metrics, such as beta, alpha, and Sharpe ratio, which aim to quantify how much risk free excess profits have been acquired by our algorithm.

For the Funding Circle dataset, we evaluate our algorithm by measuring the root mean squared error of the predicted rate on a previously unseen loan and the actual market rate of the loan on the secondary market at the point in time when the loan was scraped. For this evaluation we ignore the uncertainty around the predictions, but we take that into account when measuring the expected profits. Here we assume that the loan rates on the market are distributed according to our model posterior and therefore we simulate what would happen if we bought a loan at the observed rate that it is available, waited a little bit for the market to move and resell the loan at a new rate. The new market rate is sampled from the model posterior and our profit is quantified by the rate difference between the rate we bought and the rate we sold at. This is an example of arbitrage over a short time. In our evaluation, we first buy all loan parts at the observed rate and calculate a new price for the loan such that it maximises the expected profit we can obtain. Doing this results in significant compound profits. However, if we priced the loan parts randomly according to the model posterior, our profits would be minimal or even negative. This shows that maximising for the expected profits increases final trading profits in the simulation.

For the loan review dataset, we evaluate the prediction accuracy by measuring the mean absolute error between predicted and actual movie revenues for unseen test reviews. We also evaluate the importance of certain terms in the movie reviews which is a form of text analysis that inspects the deep black box nature of convolutional neural networks. We highlight phrases that have large impact on the prediction and phrases that actually change the prediction outcome depending on the context they appear in. This shows the importance of uncovering non-linear relationships in the data coming from the interaction of various data points or sources.

Finally, we evaluate our end-to-end trading agent relatively simply. The reason being that we aim to simulate the experience of a real world trader. We start with a unit $1 balance and at every day our trader makes limit order trades on the market for trading assets. In case the trades are filled, we record the transaction which can be either a purchase or a sale satisfying all constraints. That means for example we disallow short selling, the account cash balance cannot be negative, and the purchase prices cannot be larger than what is permitted by the current cash balance. We simulate this trading behaviour until we reach the end of our dataset. We then measure the amount of cash possessed and convert all stocks acquired at their last traded price to cash. This gives us the net worth of the trader. We aim to maximise this net worth. When we evaluate the algorithm, we plot the stocks it bought, and compare this to a baseline buy and hold algorithm or an algorithm that always buys the latest largest growth stock from the previous period. We also show the variance of the returns and measure whether our algorithm is statistically significantly different to any of our baselines.

Note, measuring trading in real life is made more difficult by issues of market impact, transaction costs, slippage, implementation shortfall, software bugs, or simply bad luck. Therefore, where possible, we aim to measure statistical significance in our results and evaluate on test data spanning long stretches of time. We also assume that for large markets and small budgets, our trades have little impact, and we conduct simulations of our strategies on this test data. This way, we can measure our algorithms on a multitude of markets using realistic evaluation metrics. In the next section, we give a theoretical review of machine learning frameworks used.

Chapter 3

Machine Learning Frameworks

In the previous chapter we have reviewed a theory of markets and some techniques scholars in the past used to analyse these markets, and potential experiment setups. In this chapter, we review different classes of machine learning models that may be useful for forecasting markets with a data driven approach. For a discussion about the advantages of this technique, please refer to Section 2.1. We also explain how we can include multiple modalities of data into these systems, and what other fields machine learning has had great successes in. We first start with linear models and investigate multi-task regularization techniques. Then we move on to Gaussian Process that allows us to model non-linearities via kernels and also incorporate uncertainty into the predictions which we can exploit. Finally, we review the successes of representation learning and deep neural networks, and show how we can construct arbitrary differentiable directed acyclic graphs to learn good trading strategies. We also pay particular attention to textual data and show ways in which text is used in forecasting.

Machine learning, a branch of artificial intelligence, concerns the construction of systems that are able to learn from data. The aim of a machine learning algorithm is to uncover a hidden pattern or function that can explain how data can be used to make predictions about some unknown variable. That assumes the existence of a function that is able to map the inputs to the outputs. In supervised machine learning, data point examples are shown to the algorithm where the desired output is known in advance. This dataset is used for training the algorithm so that it learns the function of interest. The learning process involves defining and measuring the prediction error the algorithm makes on the training data, and iteratively adjusting the model parameters, also called weights, in such a way that with each iteration the prediction error becomes smaller. However, care has to be taken to ensure that the algorithm does not overfit the training data, or else when the system sees unseen datapoints, the predictions will be largely incorrect. Therefore, another dataset is used for validation. The purpose of validation is to check that the algorithm does not overfit the training data. On this dataset, additional parameters of the learning function can be tuned. In the end, the trained model is evaluated on unseen datapoints in a test dataset, and the prediction error is recorded in order to gain a realistic measure of the predictive power of the model. The loss function measures the error the algorithm makes when it produces predictions and these are compared to some gold standard targets or labels.

In machine learning, a feature is an individual measurable heuristic property of a phenomenon being observed. It is important to determine the features used for a machine learning algorithm. A good choice of derived and raw input features can greatly aid the algorithm to converge to a solution. We must ensure that we have adequate data and high variance models that can extract the relationships between the input features and the targets. Otherwise, adding more data to a model may not be always beneficial especially if the model used is high bias and underfits. High variance refers to systems that have a large capacity to explain the underlying relationships in the data, whereas high bias means that the model is too rough or simple for the task at hand. In the next section we cover linear models.

3.1 Linear Models

Given a data set of some statistical units, a linear regression model assumes that the relationship between the dependent variable y and the regressors X is linear. This relationship is modelled through an error variable ϵ, which adds noise to the model. Given M the number of datapoints and N the number of dimensions, $\mathbf{y} \in \mathbb{R}^{M\times 1}, \mathbf{X} \in \mathbb{R}^{M\times N}, \mathbf{w} \in \mathbb{R}^{N\times 1}, \epsilon \in \mathbb{R}^{M\times 1}$ linear regression predicts the following way

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon. \tag{3.1}$$

One way to solve for $\mathbf{w}$ and ϵ is to minimise the ordinary least squares loss function. Here it is assumed that a bias term X_0, which takes the value 1, is appended to X which has its own weight w_0. An analytical solution can be found as follows,

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \, ||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 \tag{3.2}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}. \tag{3.3}$$

Predictions of the model can be squashed between 0 and 1 by applying the logistic function, which results in the logistic regression algorithm. This is widely used for binary classification with a defined threshold, and also for ranking datapoints as follows

$$\mathbf{y} = \frac{1}{1 + e^{-\mathbf{X}\mathbf{w}}}. \tag{3.4}$$

Often real world problems are ill-posed, as solutions may be under or over-determined. One way to circumvent this issue is to use Tikhonov regularisation (Tihonov, 1963). Occam's Razor states that between two equivalent models, the one that is simpler is preferred. Therefore, to keep the weights of the model low and to prevent overfitting, a regulariser term is added to the loss function

$$\hat{\mathbf{w}} = \underset{w}{\operatorname{argmin}} \, ||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 + \alpha \Lambda(\mathbf{w}) \tag{3.5}$$

$$\Lambda(\mathbf{w}) = \|\mathbf{w}\|^2. \tag{3.6}$$

The function of the regularisers is to prevent overfitting the dataset. Overfitting occurs when the model describes the dataset with very small error, but for new datapoints the prediction errors are very large. Hence, the model cannot adequately generalise. A regulariser parameter imposes a penalty on learning too complex and well-fitting models. In the training process, some amount is added to the loss function which is minimised by the optimisation procedure. In the end, the optimisation arrives to a compromise where the model does not have high variance, ie. fitting the training set well without adequate generalisation, nor does it have high bias, fitting the training set roughly without adequate generalisation.

3.1.1 Utility Maximisation

We have seen an overview of linear models. Next we investigate how we can optimise linear models directly for a metric we care about. In a financial context, this may be trading profit or other types of utility functions we may wish to construct.

Optimising for the right objective is critical aspect of machine learning and can profoundly affect the results. Therefore, a remaining question is the choice of training objective for fitting models to financial data. The most common form of minimising the error in a prediction model is to minimise the squared loss. However, in a stock market setting, often people do not care about prediction accuracy, rather they care about maximising the profit or minimising the negative profit. Under such circumstances, the loss function needs to be modified to take this into account. Minimising squared loss is also inappropriate for the reason that profit and prediction accuracy are often not significantly correlated in a financial context.

Previous work has explored the possibility of replacing prediction criterion in training, ie. minimising squared loss, with a financial criterion (Bengio, 1997a). Bengio applied the back-propagation algorithm through time to identify optimal weights to determine the share of each stock or asset in the portfolio at a given time step which was one month. In his recurrent neural network configuration, 5 macro and micro economic variables were identified as external inputs.

In contrast to (Bengio, 1997a), in our experiments we would like to optimise for profit which is a financial criterion suited to our setting, instead of forecast accuracy of stock price movement. The model trained is linear which is akin to a one layer neural network. In addition, separate models for different companies should be related to one another during training via multi-task learning. Furthermore, the training is done on daily, not monthly, market data. Last, time-wise multi-task learning could be exploited to take temporal changes into account with a fixed time window that is not varied.

Given that y, i.e. the targets, represent the relative price movement or returns of some asset and the prediction means how confident we are that we should either buy or sell that asset, with +1 meaning we should definitely buy and -1 meaning we should definitely sell, we can write a profit (or utility) maximisation objective as follows:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \left\{ -\mathbf{y} \tanh \frac{\mathbf{Xw}}{2} \right\} \tag{3.7}$$

$$\tanh \frac{\mathbf{Xw}}{2} = \frac{2}{1+e^{-\mathbf{Xw}}} - 1 \tag{3.8}$$

The scaled sigmoid produces a prediction between -1 and +1. This is multiplied by the relative price movement of the asset. This results in the profit over a day. Then the negative profit is minimised across all examples.

Equation 3.8 is one possible formulation of profits that we can incorporate into our model. Formulating the profit this way instead of predicting shares of stocks in a portfolio as seen in (Bengio, 1997a) may have the benefit of being able to learn a separate trading model for each company and for each trading period, which can be more robust to overfitting. In the next section, we look at how we can further uncover signal from a cluster of related models by allowing knowledge transfer via multi task regularisation.

3.1.2 Multi Task Regularisation

We have seen how the objective of a linear model can be modified to take into account a financial criterion. Next, we investigate ways to make linear models more complex so as to better fit the distribution of our data, which may come from a collection of related domains or tasks.

Supervised learning is the computational task of discovering an arbitrary function from labelled examples. The central tenet of supervised machine learning is finding good ways of generalisation. In other words, we are interested of mapping inputs to outputs and discovering relations by training on our data that results in good prediction performance on some unseen test data in terms of the loss function of our choice. Many times in supervised learning tasks, the assumption is made that the training and the testing distributions are identical. This allows for generalisation to test data instances from training examples. However, often times this assumption does not hold in real life. Humans solve this problem by recalling the knowledge they acquired on previous similar tasks in the past and slightly adapting it when applied to the unseen challenge they face. In this spirit, multi-task or transfer learning has emerged as a way to solve multiple related tasks and facilitate the transfer of previous knowledge (Skolidis, 2012).

Multi task learning is a special kind of supervised machine learning algorithm, where the input dataset is divided into smaller sub-datasets, called tasks, and separate models are trained for each task. Then these parametric models are related to one another through the use of multi task regularisers in order to facilitate the sharing of data statistics across models, but at the same time allow for variations in the models. This way, it is hoped that a multi task learning algorithm can produce better predictions than a single task learning algorithm, because it allows for structured variations in the data. In multi task learning, there are separate set of weights for each task. One way to enable the models to share information is through the use of multi task regularisers. In the rest of this section, we review some forms of multi task learning encountered in the literature and also propose new ones that may be suitable for our problem of financial trading, where potential related tasks could be divided along companies and trading periods.

Mean regularised multi task learning (Evgeniou and Pontil, 2004) shows a method to learn multiple related tasks together instead of separately, which can improve accuracy for the primary task learned. It is a special case of graph regularised problems. This regularisation can also be encoded using the structure matrix $\mathbf{R}$ (Mairal, 2012). In his paper, Evegniou describes regularised multi task learning and the mean regulariser as follows,

$$\min_{\mathbf{w}_t,\xi_{it}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \rho_1 \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 + \rho_2 \sum_{t=1}^{T} \left\| \mathbf{w}_t - \frac{1}{T} \sum_{s=1}^{T} \mathbf{w}_s \right\|^2 \right\}. \tag{3.9}$$

There are T number of tasks with m elements in each task. The objective is to minimise the weights $\mathbf{w}$ and the slack variables or error terms ξ. The first section sums all the prediction errors. The second section is the L2 regulariser that keeps the weights to 0 in view of Occam's razor. The third term is the mean regulariser, that measures the distance between the weights and the average of all weights, and tries to minimise that. This has the effect that all weights tend to the average, but some variations are allowed. Each regularisation is weighted by a coefficient ρ to find the optimum amount of regularisation that results in the smallest prediction errors on the validation data.

In case the model evolves through time, multi task time regularisation can be used. The equation is similar to the mean regularisation setting, but now the distance between adjacent tasks is measured, instead between the mean of all tasks. This is related to other temporal modelling methods such as the Fused Lasso (Tibshirani et al., 2005), which penalizes absolute changes in weights between adjacent time intervals using the L_1 norm. Here instead we elect to use the L_2 norm for smoothness as follows,

$$\min_{\mathbf{w}_t,\xi_{it}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \rho_1 \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 + \rho_3 \sum_{t=2}^{T} \|\mathbf{w}_t - \mathbf{w}_{t-1}\|^2 \right\}. \tag{3.10}$$

The first two terms are identical as above, but the third term here instead allows the model weights to vary smoothly across the tasks, assuming that task divisions were performed with a time series dataset. Similarly to the mean regulariser, a coefficient tunes the amount of regularisation that is needed for optimum performance on the validation data. It is also possible to combine mean and time regularisation.

Multi task learning can also be used for temporal data. (Caruana, 1997) explored time series prediction where he fitted a neural network with shared parameters to produce outputs for multiple simultaneous tasks. He noted that this kind of learning is especially useful for harder longer term predictions where best results can be obtained when the middle task is predicted from previous and later tasks. To take temporal changes into account (Bengio, 1997a) trained on a window of data, which he shifted through time. This is a naive way for accounting for time as it assumes non stationarity but sacrifices information sharing that goes beyond the window. In another experiment, Bengio used a recurrent neural network, with five macro and micro economic variables as external inputs.

In our experiments, we consider a linear model, with its outputs mapped to trading actions via a non-linear sigmoid function. This is akin to a one layer neural network, although the Gaussian Process method can support many non-linear models. Here we account for time using a regularization method which ensures smoothness between adjacent time periods, where model parameters were stratified by month. Our experimental setup focuses on extrapolation a month into the future, rather than the easier problem of interpolation, as done in (Caruana, 1997).

Another common form of regularisation is the elastic net (Zou and Hastie, 2005; Lampos et al., 2013). Bilinear Elastic Net and Bilinear Group Elastic Net have shown good results with regard to political opinion forecasting based on the Twitter dataset. This regulariser can help define groups of features that may correlate with each other. It also results in sparse solutions and it is especially useful when the number of predictors is larger than the number of data observations. It has been shown that it outperforms the LASSO regularisation method. The elastic net provides a transition between the L2 and L1 regularisation. It can be written as follows,

$$\min_{\mathbf{w}_t,\xi_{it}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \rho_4 \sum_{t=1}^{T} \frac{1-\alpha}{2} \|\mathbf{w}_t\|_2^2 + \alpha \|\mathbf{w}_t\|_1 \right\}. \tag{3.11}$$

The α parameter tunes how much the L2 vs the L1 regulariser takes effect. When the parameter is 1 that is equivalent to LASSO whereas when it is 0 that is ridge regression.

Often prediction problems have very high dimensional feature sets, as in the case of text input, and therefore they can take a long time to process computationally. One way to reduce the dimensionality of the data is through the use of joint L1/L2 regularisation (Obozinski et al., 2010) that selects the appropriate features in a shared space across tasks for multi task learning. This algorithm learns a projection to lower dimensional space by a blockwise path-following scheme (Obozinski et al., 2010) using random projections, and it is implemented by the SPAMS package (Mairal, 2012).

The above examples of multi task regularizers enable the transfer of data statistics across domains, which allows for more complex modelling of the problem. In the following, we review some additional works that made use of multi task learning in a financial context.

Multi task learning has been investigated as an effective way of improving predictions of machine learning models in finance. (Ghosn and Bengio, 1997) studied whether sharing hidden layers of neural networks between companies could improve the selection of high return stocks. He trained neural net parameters on one company and used them to produce predictions for other stocks. He also examined whether selective parameter sharing of various neural net layers could aid prediction. In his paper, he achieved significant market returns with a trading system that was based on this model. (Bengio, 1997a) experimented with portfolio optimisation over 35 stocks and used one model for all companies. He concluded that with sharing neural network parameters across companies, better results could be obtained. To take temporal changes into account, he used different size values for the training window on the data. Last, (Cha and

Chan, 2000) explored domain adaptation by which he trained his network on all stocks but one, and tested on the held-out stock. These papers show that relationships between companies contain valuable information that can be mined for trading. Therefore, in our experiments separate models for different companies are related to one another during training via multi-task learning, considering investment in various stocks taken from the FTSE 100 index over a decade of historical trading data. Our working assumption is the mechanisms for predicting individual stock movements will be highly correlated, through model regularisation towards a common shared component.

Multi task learning can also be used for temporal data like financial markets. (Caruana, 1997) explored time series prediction where he fitted a neural network with shared parameters to produce outputs for multiple simultaneous tasks. He noted that this kind of learning is especially useful for harder longer term predictions where best results can be obtained when the middle task is predicted from previous and later tasks. To take temporal changes into account (Bengio, 1997a) trained on a window of data, which he shifted through time. This is a naive way for accounting for time as it assumes non stationarity but loses any information sharing that goes beyond the window. In another experiment, he applied the backpropagation algorithm through time. In his recurrent neural net configuration, five macro and micro economic variables were identified as external inputs. In our experiments, we do time series regularisation to ensure smoothness between adjacent times, discretising time by month, so we have per-month models. Our experimental setup is also more by testing extrapolation well into the future, rather than interpolation in (Caruana, 1997).

These models have numerous applications, including modelling financial markets. In stock markets, the FTSE 100 index is made up of companies with the highest market capitalization. It could be assumed that by examining the performance of these companies, one can make deductions with regard to the economy as a whole. Therefore, these companies follow an average market growth, but there might be variations in their individual cases. Another aspect of the London Stock Exchange is that the underlying market conditions change across time therefore it is unlikely that one model could describe a stock movement accurately all the time. For example, with the stock market data, one could create a task for each company and mean regularise against all companies, as well as one could divide each company time series dataset time wise, for example create a task every new month of data in a larger dataset of 10 years of stock market trading. In the Funding Circle dataset, tasks can be divided by industry and loan parts may evolve across time in terms of riskiness and premiums on markup. The primary lending market may also be time dependent, since economic conditions change and alternative borrowing opportunities may become available for potential businesses looking for funding.

This ends our discussion of multi task learning. Next we review validation in machine learning which has the purpose of avoiding overfitting. We note that using specific forms of validations we can account for the temporal nature of our problem and make our training and testing distributions more similar, which is a problem also addressed by multi-task learning.

3.1.3 Training and Validation

It is important to validate the model on a heldout dataset in order to prevent it overfitting or underfitting the training dataset as a result of which the prediction errors are going to be high on the unseen test dataset. In the following, we discuss ways to train the hyperparameters of our models with a specific focus on a time series setting. Hyperparameters are distinct from model parameters in that they refer to the training procedure of the learning algorithm. In a Bayesian setting, hyperparameters may define a prior distribution of the data, to which the learning algorithm fits a posterior distribution.

The standard way of tuning model hyper parameters is through grid search, which constructs a grid of possible hyper parameter values, trains the model for each combination of values and picks the one with the best performance on the validation set. However, this process can be very resource and time intensive and people have proposed alternative models that may find better solutions in reduced time under certain circumstances. Therefore, another way to tune the hyper parameters of the learning algorithm is through random search. Due to the fact that random search can cover spaces which grid search leaves out, it can provide optimal solutions for the optimisation problem after potentially fewer iterations than grid search (Bergstra et al., 2011).

Alternatively, people have experimented with Sequential Bayesian Optimisation (Snoek et al., 2012) which aims to tune the hyperpamaters of the model by constructing a secondary machine learning model, often probabilistic in nature such as a Gaussian Process, and using that to optimise the performance of the primary model on a held out set. In this case, the optimising variables may not be differentiable and therefore cannot be tuned using conventional gradient based optimisers. Although SBO is more efficient than brute force search such as grid search, one disadvantage of it is that it is not embarrassingly parallelizable. Even though the secondary machine learning model may have fewer hyper parameters than the primary, those need to be tuned as well. In a Bayesian setting this can be done by maximising for the marginal probability of the data, which is one way to address this issue.

A further way to tune the parameters on the validation set is through the coordinate descent algorithm (Friedman et al., 2010). In this framework, first some potential values for the parameters are selected just like in grid search. However, instead of iterating through each value combination as in grid search, only one parameter is searched at a time, while all other parameters are kept constant. Next, another parameter is searched while everything else is constant. When no parameter values change anymore, the optimisation procedure terminates. The direction as well as the stepsize of the search is determined by line search.

Apart from hyper parameter settings, validation can also distinguish between different experiment setups, which is a case of model selection. For example, the validation performance of a model that is run with technical analysis features and a model that is run with a window of historical returns as input to the learning function can be compared. Validation can tell how much principal component analysis reduces performance as opposed to when no dimensionality reduction is performed, or what kernel functions are best suited for the learning problem at hand in a Gaussian Process. With neural networks, validation can select the optimal network architecture, including number of hidden layers and hidden units in each layer. In a Bayesian setting, both model selection and hyperparameter tuning can be performed with respect to the marginal likelihood of the training data, which makes validation technically unnecessary.

For validation to work effectively, we first need to hold out a portion of data set. There are multiple ways of making a selection for which examples to hold out. Typically in stratified crossvalidation a random subset of the data is selected, on which the validation performance is measured. Then a different subset is selected with the training and evaluation repeated. This procedure is repeated many times, denoted by the letter K in K-fold crossvalidation terminology. In the end, all validation evaluations are averaged to obtain a final metric of performance. Even though K-fold crossvalidation is widely used, in a time series prediction setting it is not recommended. In time series prediction problems that are characteristic of financial markets, sliding window validation and training is preferred instead. A sequential training and testing is executed in order to simulate a time series prediction model, ie. extrapolation rather than interpolation. A stratified training and testing model would allow for the knowledge of future information when making predictions for today. This is due to the fact that training and testing observations are randomly sampled from the dataset, therefore a testing datapoint may occur with an earlier timestamp than a training datapoint. This is unrealistic in a real prediction scenario because by training on datapoints that come after the testing datapoints, knowledge about the future can be acquired.

Alternatively, online and/or active learning can also be explored as an training method. In this case, we dynamically train on examples and test on the following example in time, extrapolating only one step into the future. Then the test example is added to the training set and a parameter update is performed. For speed, the learning update, i.e. a gradient step, can be done only on the newly added example. Then a further extrapolation is performed on the next time step, which is then also added to the training set, and this training testing alternation is continued until the end of the entire dataset is reached. More fine grained time-wise division of the tasks is possible. For example, one time step could be represented either by a month or a day. In addition, retraining on new examples can be done in batch mode or online, taking the entire training dataset or only on the newly added example, the former of which is more computationally expensive to perform. As for how time is represented, note that other methods such as kernel approaches can efficiently deal with fine grained or even continuous representations of time. It is also important to note that training should be done on a large time frame with many datapoints. It is hypothesized that certain patterns can be uncovered in the dataset with a sufficiently flexible model but these patterns may not be visible in too small datasets (Banko and Brill, 2001).

As noted above, online learning means only one example is trained on at a time and a small fixed gradient step is made towards minimising the cost. In this setting, the tasks should not be partitioned time-wise since online learning automatically adapts to temporal variations in the model, also known as non-stationarity or concept drift in the data (Gama et al., 2014). By remembering the previous parameter values before initiating a gradient update, we effectively perform a warm optimisation start, and adapt to the drifting concept gradually. Nevertheless, online learning never actually finds the optimal weights, rather it oscillates around them due to noise in the data observations. One challenge is to adequately tune the learning rate parameter that controls the tradeoff between speed and accuracy of convergence. The disadvantage of the batch training approach is that it evaluates on an entire time block before updating the weights to a more current model. For example, testing towards the end of the time block would benefit from training on observations towards the beginning of the test time block in terms of prediction accuracy as there would be less lag between trained data points and test data points. It may be the case that the longer the prediction horizon is into the future, the weaker the prediction signal becomes, as there is a larger risk of concept drift taking effect in the test predictions.

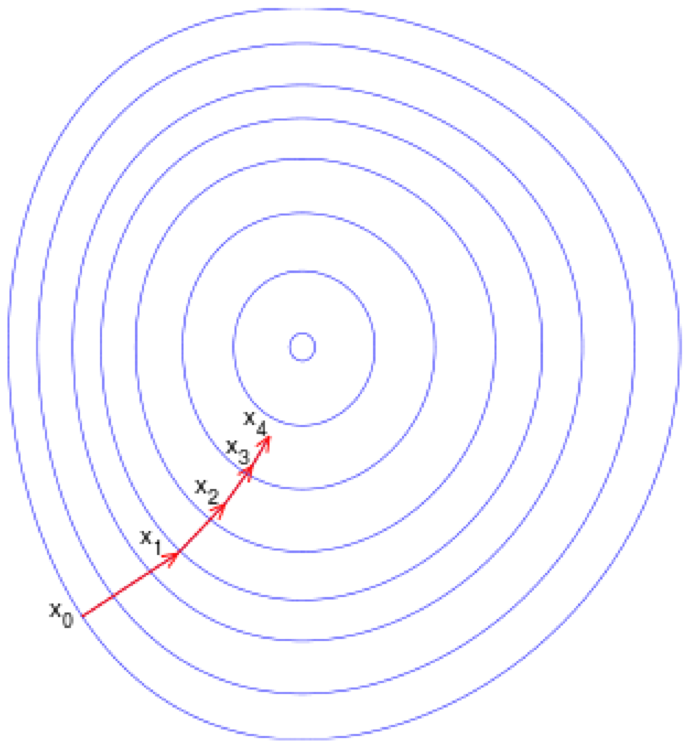


Figure 3.1: Gradient descent iterative optimisation. X indicates function evaluations, red arrows show gradient step, blue contours show cost function value.

There are many ways to optimise a learning algorithm iteratively. Gradient based optimisation methods generally outperform evolutionary, genetic or particle swarm optimisation algorithms. An illustration of convergence can be seen in Figure 3.1. For the gradient descent optimisation a momentum term can be added, which can have better performance results than conjugate gradient descent (Qian, 1999). A particular type of momentum is the Nestrov momentum (Nesterov et al., 1994) which aims to correct overshooting in gradient updates. An adaptive learning rate can be combined with the momentum for each weight dimension (Yu and Liu, 2002). The Rprop algorithm (Riedmiller and Braun, 1993) or RmsProp (Dauphin et al., 2015) could result in just as fast optimisation procedure. Gradient descent can be performed by randomly shuffling the dataset and taking one example at a time, which gives the stochastic gradient descent algorithm. Shuffling ensures the gradient is well estimated and does not reflect idiosyncratic patterns of consecutive datapoints. In order to make efficient use of linear algebra and vector math implementation of numerical libraries, a small batch can be taken on each iteration, which results in the mini-batch gradient descent. The limited version of the BFGS minimisation algorithm based on hashing and second order curve approximation of the objective function is also preferred for many optimisation tasks (Byrd et al., 1995). This is an iterative method for solving unconstrained nonlinear optimisation problems. Even though it assumes convexity in the loss function, local minima can often be escaped effectively. Variants of gradient descent are the AdaDelta (Zeiler, 2012), AdaGrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2014) optimisation procedures. These accumulate a running history of gradients and parameter updates and aim to adapt the learning rate for each parameter individually. This way, they aim to achieve faster convergence and also attempt one-shot learning, i.e. learning from rare examples. There is no golden rule as to which optimisation procedures performs best on a given task, as it is difficult to visualise the high dimensional error surface of the problem, so it is advisable to always try a few of them and compare the results.

This concludes our review of training, validation and testing techniques for learning. In the next section, we examine how linear models were previously used for text based prediction tasks.

3.1.4 Text Regression

In this section, we investigate ways in which text can be incorporated into predictive modelling. Since most previous work focused on linear models, our review also mainly covers this class of models. For non-linear text based predictive systems that is proposed in this thesis, please refer to chapters 5 and 6.

Text regression is the problem of predicting a quantifiable real world phenomenon from unstructured text inputs. This has seen diverse applications including the prediction of stock volatility from company reviews (Kogan et al., 2009), market movements from news (Lerman et al., 2008), movie revenues from critic reviews (Joshi et al., 2010) and political opinion polls from social media posts (Lampos et al., 2013). Almost exclusively, these approaches have used linear predictive models, which aids model interpretability, nonetheless these approaches are inherently unable to handle non-linearities (one notable exception is (Wang and Hua, 2014), who modelled stock volatity using a non-linear copula regression model). In particular, this is likely to be problematic for financial market data, which is known to be noisy and highly non-linear (Atiya, 2001). In this thesis, we propose, among other frameworks, a Gaussian Process model with non-linear kernels for modelling P2P lending rates, which we show improves significantly over linear modelling. We retain model interpretability through automatic relevance determination (ARD) and learned kernel importance weightings, which identify the relative importance of features and feature groups.

A key question is how to represent text. Many models make a "bag-of-words" assumption, a simplification which facilitates fast training and scaling to large datasets. However, text has much deeper semantic meaning which incorporates its context in the document and the corpus. There have been mixed results in prior work on text regression when attempting to use deeper representations for text. (Kogan et al., 2009) found that TF/IDF alone does fairly well even without additional text processing. Similarly, (Joshi et al., 2010) found that part of speech tagging and dependency relations did not contribute to performance improvement. In our work, we use TF/IDF scores and extract semantic meaning with LDA topic modelling (Blei et al., 2003), which can automatically capture latent semantic information in the document that would not be accessible directly from the bag-of-words. Although TF/IDF scores have been used for text regression in the past, in our model we also add the LDA topic distribution to the feature set. In the following, we review key papers in text regression in detail.

Box-office revenues for movies, spikes in book sales, Academy Award winners and national elections are just some of the many examples that can be predicted with social media text. The bag of words approach is currently the standard way to approach text regression problems. This involves preprocessing the text and representing the document in the vector space model. Next, we review some works making use of text as an input for forecasting real world quantities, with many examples coming from the financial domain.

Text regression has been applied to volatility prediction (Kogan et al., 2009). It has been shown that support vector regression(Drucker et al., 1997) is able to predict stock volatility of companies close the performance of GARCH (Kogan et al., 2009). The optimisation objective is formulated as the following:

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} \max(0, |v_i - f(\mathbf{d_i}; \mathbf{w})| - \epsilon) \right\} \tag{3.12}$$

$$f(\mathbf{d}; \mathbf{w}) = h(\mathbf{d})^{\mathrm{T}}\mathbf{w} = \sum_{i=1}^{N} \alpha_i K(\mathbf{d}, \mathbf{d_i}) \tag{3.13}$$

where C is the regularisation constant, ϵ controls the training error margin, v_i is the target volatility, h maps the documents to a vector space representation, K is a kernel or similarity measure, and α_i is a learnt weight per data point. For interpretability of results, a linear kernel was used. The authors trained a model based on the text content of financial annual reports. Combining the two methods certainly improved the baseline performance. Documents were represented with TF, TFIDF or LOP1P:

$$\mathrm{TF} : h_j(\mathbf{d}) = \frac{1}{|\mathbf{d}|} freq(h_j, \mathbf{d}), \forall j \in \{1, ..., M\} \tag{3.14}$$

$$\mathrm{TFID} : h_j(\mathbf{d}) = \frac{1}{|\mathbf{d}|} freq(h_j, \mathbf{d}) \times \log(N/|\{\mathbf{d} : freq(x_j, \mathbf{d}) > 0\}|) \tag{3.15}$$

$$\mathrm{LOG1P} : h_j(\mathbf{d}) = \log(1 + freq(x_j, \mathbf{d})) \tag{3.16}$$

and the mean squared error was measured. An interesting outcome was that the training data was highly time dependent, and thus it was unclear whether adding more past data helped prediction as the training window was kept constant at 5 years. They found that some term weights changed over time, for example, "higher margin" and "lower margin" swapped signs. In 2002, a law was introduced that made noticeable changes in the learned weights. Some terms such as "estimates" and "accounting policy" increased and were associated with lower volatility after the passage of the law. They also tried to model delisting of a company but did not have enough data. This paper highlights the fact that markets change over time which is reflected in the influential phrases in the text document too.

Another paper attempted to forecast movie revenues based on the text content of movie reviews found on the Internet (Joshi et al., 2010). They used a bag of word approach to create a vector space representation of words in each review, and in other experiments they performed multi task learning by denoting each review by its source, i.e. Varsity or LA times. In addition, they explored whether adding meta data to the model, such as the genre, rating and budget of the movie helped the predictions. They found that just by using the movie reviews text they could get very high accuracy, so they were willing to substitute the numerical data for the text data. The model they used was linear regression with elastic net regularisation. They explored whether adding additional text processing increased the model performance, such as in addition to the n-gram bag of words, up until trigrams, after removing stopwords. These additional features included part of speech tags and dependency relations, but they found that these did not contribute significantly to the model. Some interesting findings include that "independent", "documentary" and "rate_r" phrases had a negative weight, which meant decreased revenue. In addition, "will_smith" and "this_series" had high weights which meant increased revenue. In Chapter 6 we propose a nonlinear convolutional neural network model that achieves 40 % improvement over the results achieved in this paper which highlights the need for modelling non-linearity. In addition, we obtain our results without any pre-processing of the text, apart from tokenisation, while also retaining model interpretability via a novel phrase impact measurement technique we propose.

Besides blog reviews and financial documents, people have also used social media, such as twitter to forecast markets and election results. A user centric model of voting intention (Lampos et al., 2013) examined whether it is possible to predict voting intention from social media, such as Twitter. Users were sampled from UK and Austria and filtered by geographic location. Another experiment was about predicting stock market prices with the same dataset. The model they used was an elastic net with a bilinear objective. In a separate experiment, the regularisation was performed by a mixed l1/l2 regulariser. There were separate set of weights for words and users and the model was optimised in turns for each set of weights. It was found that the model accuracy had a low root mean squared error on the test set. An interesting user was JobsBhamAdmn which was spamming job ads, which is normally ignored by users, but this account had high weight in the model, which means that at the time Birmingham had high unemployment rate and was labour friendly, therefore its tweets actually had high influence. This paper highlights the fact that it is not just text itself that is important for predictions but also metadata such as the author of the text or geographic region.

It has been also shown that combining objective news forecasting with learning financial market trends produce good results with ensemble learning (Lerman et al., 2008). They performed ridge regression to classify past market data in an online manner to predict future market movement and measure the profit proportional to the price change. They combined this system with a bag of words approach for ngram modelling of news stories and also performed dependency parsing on it, i.e. determining subject and object of words for relations and entities. They chose the better system each day based on a 3 day window of past prediction accuracy. They found good results, where the news forecasting alone did not perform well, but combining that with past market data improved prediction accuracy. This paper again shows that while designing text driven forecasting methods, it is important to include available structured data, which in this case was historical market data. However, we must note that they trained the market and text based models separately and combined them with ensemble learning. A better approach would be to train the models jointly in order to allow for interactions of features.

To further support the idea of combined modelling of multi-modal datasets, a recent paper on Kickstarter has shown that by extracting text mining and structured information from a Kickstarter project, a simple decision tree was able to predict with 68 % accuracy if a project was going to be successfully funded (Greenberg et al., 2013). The authors scraped the information themselves. They planned to create a tool that could aid new applicants to tailor their profiles more successfully to the investors.

Another paper attempted to predict the end price of Ebay auctions (Ghani and Simmons, 2004), based on data extracted from the auction page. The reputation and comments by previous users could also be included in the model (Resnick and Zeckhauser, 2002). The price of the auction was put into price buckets and the model was framed as a multinomial classification task. It was found that the model had temporal variations, ie. the history of the item in previous auctions was important. With a binary classification of whether an auction is successful, a neural networks obtained 96% accuracy. The mean squared error for the regression task was around 6. Potential application would be a listing optimiser or price insurance to the seller.

A related field to text regression is sentiment analysis, which is a popular technique for financial datasets. However, it has the disadvantage of inserting an intermediate problem into the prediction task, where the problem becomes first to predict the sentiment of the text and next to pipe that output into a market prediction algorithm. If there is an error in the sentiment prediction part, then that will propagate into the downstream models. Therefore, text regression has the advantage of skipping the sentiment analysis part and directly forecasting the target variable we are interested in from the text. This supports an integrated modelling approach, where we aim to achieve superior results by jointly modelling multi-modal datasets for a common task, rather than separately. Next we review a few sentiment analysis papers.

Gloor measured the number of times "fear" "worry" and "hope" was mentioned in Twitter for six months in 2009. He used a feed containing a randomized sample of 1% of all tweets. He found that instead of hope being positively and fear and worry being negatively correlated with stock market movement, they were all correlated with the volatility of the stock price, i.e. either moving up or down. This means when economic times are uncertain and the stock prices change a lot, people use more emotional words on Twitter (Zhang et al., 2011).

Skiena has shown that it is possible to create a trading strategy by analysing online sentiment from various news sources and predicting stock market returns (Zhang and Skiena, 2010). He found that the predictive power is indicative to the same day, i.e. news sources in the morning can predict the outcome of afternoon, and best returns can be achieved with a portfolio of 3 stocks. After that average sentiment across stocks drops which negatively affects the returns. He found the polarity with returns provides the best correlation with an R of 0.47. In addition, a holding day of 1 days or less works the best. Yesterdays news has little effect on the outcome, and similarly predicting tomorrow does not work. Further research is needed to make more fine grained predictions. Skiena states that Twitter has the best performance because of its mood consistently affects the market, whereas with normal news the correlation is not that significant. Polarity is calculated by the number of positive words minus the number of negative words divided by the number of all words.

Gilbert showed that our emotional state influences our choices (Gilbert and Karahalios, 2010). He demonstrated that estimating emotions from web-blogs provides novel information about future stock market prices, which is not in the stock market data. He estimated anxiety, worry and fear from a dataset of over 20 million posts made on the site LiveJournal. Using a Granger-causal framework, he found that increases in expressions of anxiety, evidenced by computationally-identified linguistic features, predict downward pressure on the S&P 500 index. He confirmed the results via Monte Carlo simulations.

The last paper related to sentiment analysis that we review is by Bollen who predicted the stock market with Twitter sentiment. In his paper, an autoregressive neural network and self organizing fuzzy network were combined with sentiment analysis to predict DIJA closing prices with a 15 minutes horizon (Bollen et al., 2011). He found up to 87% correlation, though this estimate may be biased upward due to the small dataset size and insufficient validation. Experiments that attempted to replicate his approach produced 60-70% accuracy.(Chakoumakos et al., 2012; Chen and Lazer, 2012; Chyan et al., 2012; Debbini et al., 2012; Hsu et al., 2012; Kuleshov, 2012; Lee, 2012; Mittal and Goel, 2012; Sprenger and Welpe, 2012) According to Bollen, the stock market is a complex non linear system. General mood states do not affect it but rather specific mood states. For example, Opinion finder did not have significant correlation, but GPOMS with extended vocabulary did. Namely, for Calm and Happy word types in the vocabulary. Calm was the main indicator, while Happy seemed to aid accuracy. Nevertheless, by themselves these indicators were not significant enough, but only when combined with the self organizing fuzzy neural network. The result was analysed by Granger causality analysis, which produced an F-statistic p-value. The overall results were measured by prediction trend accuracy.

Overall, what these papers show is that text often contains useful information with regard to predicting a real world phenomenon, and this information if often complementary to other sources, such as structured data, that may be available. In addition, on certain occasions text can be non-stationary where the meaning of phrases can change over time. Last, we should strive to create end-to-end integrated systems and optimize for the right metric that we seek to evaluate as this may improve performance. In summary, it is useful to include text as an additional medium for forecasting financial markets. Text may be available on social media, such as Twitter, in financial news sites or blogs, or other social media outlets, such as Funding Circle. Text may give indication about the emotional states of the crowds, or simply contain additional facts and information not captured by structured meta data features in the input. These in turn may influence the behaviour of markets at large. However, care must be taken when performing sentiment analysis in a modelling pipeline, because it can lead to error propagation to the final market prediction task. For example, in chapter 6 we demonstrate that sentiment analysis and text regression does not always correlate with each other in terms of prediction accuracy. We do experiments with text input in sections 5.2, 6.1 and 6.2. In the next section, we review some target metrics that we can either predict directly or incorporate into a custom objective measuring profit.

3.1.5 Target Returns

Various loan metrics can be predicted with the use of multiple output multi task learning models. These include the default likelihood of a loan, the end interest rate on the loan, the end interest spread, the secondary market markup, asset returns or even projected revenues of movies. These can be either directly regressed against, or they can form part of a more complex optimisation objective measuring trading profit or net worth, which we would then maximise. This choice would also indicate the kind of output our models could produce, which may be the revenue predicted, or specific trading actions.

For stocks, the dataset can be labelled in such a way that the target of the prediction is the relative closing price movement of the stocks from the present closing price to the following day closing price,

$$y(t) = \frac{s_{close}(t+1) - s_{close}(t)}{s_{close}(t)}. \tag{3.17}$$

$y(t)$	: target of prediction for time step t
s_{close}	: closing price of stock at time step t+1 and t

This kind of labelling is useful when we aim to predict the profit or returns that could be achieved by trading daily on the stock market. This is usually the standard way to predict market movements, as traders are typically interested in the amount of profit that could be made on the market. For example, if £1 is invested into a company, and the relative price movement is 100%, ie. the price goes up, then when there is a sell the next time step, the profit is 100%. On the other hand, if the price went down, then there would be a loss.

On a market where loans are bought and sold, returns define the percentage profit that is achieved on an investment over a period of time, usually one year. When investing in the Funding Circle marketplace, this means that given a time and amount that is invested in a particular loan part, when that loan part is sold to other investors on the secondary market, how much profit can be generated at the time of sale (markup), or until the loan matures and expires naturally (interest). The definition is identical to the stock market trading scenario, the only difference being that the marketplace is different as well as the traded security, which is a loan part in this case instead of a stock of a company.

When calculating the gold standard returns, it is recommended that they are adjusted by the risk of the investment. For example, a loan with really high return may be more likely to default, which wipes any combined returns on other investments as well. Therefore, from each returns calculation, the risk of defaulting must be subtracted to obtain the net or risk adjusted returns. Trading fees should be similarly removed from the profits. Last, the risk free return, or average market return, can also be removed in order to determine the excess returns that can be made. For details on how to calculate the excess return, please refer to section 3.1.7.

A particular interesting research question is whether it is possible to achieve higher returns by trading on the secondary market, than by holding on to the loan and waiting for them to mature, ie. the borrower pays back the money owed. In other words, is there any difference between how the primary and secondary markets judge worthiness of an investment.

Instead of predicting returns, another option is to predict the price of a security or asset at specific intervals or time points. This information can be used for day trading (Wang, 2002). Cha (Cha and Chan, 2000) has proposed a system that output buy, sell and hold trading signals and worked on stocks that were not part of the training set. In his system he extracted local maxima and minima from the dataset and trained a neural network to predict these points. He used a dataset of around 800 trading days with three companies on the Hong Kong Stock Exchange. He considered investing proportional to the strength of the trading signal but left the implementation of such profit maximisation objective to future work. He also explored domain adaptation by which he trained his network on all stocks but one, tested on the held-out stock and got satisfactory results. In our experiments, we aim to determine the trading signal by a learning algorithm that invests based on the strength of the signal. In addition, we train on several thousand data points across almost a hundred companies on the London Stock Exchange, which is a much larger dataset than Cha used. This way, our algorithm gives trading actions directly instead of regressing against the return variable. In sections 2.2.4 we review more papers dealing with market forecasting, in sections 4, 5.1 we present models that predict trading actions, while in section 6.2, our model actually outputs limit order trades. In section 5.2 we perform regression of loan rates, but on top of this output we can optimise for trading actions.

Prediction markets are speculative markets created for the purpose of making predictions for the outcome of future real life events. It is also possible to trade on prediction markets when one tries to predict the outcome of an event. The current market price of the event reflects the investor confidence as to what the outcome is likely going to be. In this case if the prediction for the likelihood of realisation of the event is higher than the market price, we can make a profit by buying. For each realised event we get a positive revenue depending on the accuracy of our prediction and for each not realised event we get nothing. For an experiment regarding forecasting movie revenues, a problem which can be exploited in prediction markets, please refer to 6.1. Another metric commonly predicted in financial markets is volatility, which we investigate next.

3.1.6 Volatility

Apart from returns, volatility also plays an important role in forecasting financial markets. In finance, volatility is a measure for the variation of price of a financial instrument over time. As such, it is closely related to risk, which is defined as the standard deviation of expected returns. Historical volatility is derived from time series of past market prices. An implied volatility is derived from the market price of a market traded derivative (in particular an option). The symbol σ is used for volatility, and corresponds to standard deviation, which should not be confused with the similarly named variance, which is instead the square, σ^2. Predicting volatility is useful for trading options. In finance, an option is a contract which gives the buyer (the owner) the right, but not the obligation, to buy (call) or sell (put) an underlying asset or instrument at a specified strike price on, as with European style options, or before, as with American style options, a specified date.

For example, in an experiment we can predict the volatility of a security, after querying a list of options on offer on the market. Then with the Black Scholes formula (Black and Scholes, 1973) with inputs such as the strike price, interest rate, current price, and waiting time, the implied volatility of each option can be calculated. Black Scholes describes a financial market with derivative investments. If the forecasted volatility is greater than the implied volatility, a buy opportunity is present as the option is underpriced. The option can be bought, and kept until maturity to realise a profit. This strategy is not dependent on the type of option, i.e. call or put, but might work with only European style options, which can be only realized at their expiry date. One the other hand, for American style options, which can be realized at any time, one needs to have continuous forecasting values for the volatility, which could be obtained with a Gaussian Process, see in section 3.2.

An interesting problem is correlating implied volatility of option prices to financial news stories, i.e. using implied volatility as a gold standard and finding source of fear or speculation on the web as text. Generally, forecasting volatility is an easier task in finance than forecasting returns, as alluded to in (Kogan et al., 2009).

The Implied Volatility of a stock or index is the volatility implied by an option price observed in the market (Black and Scholes, 1973). Because there are many options on a stock with different strike prices and expiration dates, each option can yield a different volatility implicit in an option's premium. Even options with the same number of days remaining until expiration, but with different strike prices, will have different values of implied volatility. Thus to use implied volatility in volatility analysis, it is necessary to calculate a representative implied volatility for a stock. There are many ways to calculate such a type of representative value. For example, it can be calculated as average implied volatility of the at-the-money options only or at-the-money and out-the-money options.

All else being equal, the more violent and rapidly moving the market (i.e. the higher implied volatility), the more expensive the option contracts are. If one stock has IV Index of 25% and another 50%, it can be said that options of the second stock are more expensive than those of the first stock (Christensen and Prabhala, 1998).[1] The IV Index of major market indices is one of the many tools of sentiment analysis, i.e. studying the prevailing market psychology and therefore can be used as a gold standard for text regression applications which attempt to gauge market sentiment.

Once we know what target metrics to predict with the trained model, we need to evaluate performance. In the next section, we review some common methods of evaluating the performance of a financial trading algorithm.

3.1.7 Fund Performance

In this section, we review common methods of evaluation for measuring a trading algorithm's performance on stock markets. The model can be evaluated the standard ways, such as by

[1] `http://www.ivolatility.com/`

measuring the loss function output on the test data without the regularisers, which gives the prediction accuracy, error or the overall profit to be made. In addition, it is possible to run simulations on the market, also known as backtesting, in order to get a more or less accurate measure of fund performance on historical data through which conclusions can be drawn about future performance. Though care must be taken with regard to Black Swan events, which are rare events that are not present in historical data, but might occur in the future. In a real life scenario, stress testing could give an adequate indication of fund performance in these cases (Taleb, 2007).

Modern Portfolio Theory gives five measures to evaluate the performance of a trading strategy. According to the Capital Asset Pricing model (Frencha, 2003), see section 2.2, in order to evaluate whether an investment is worth the capital, the investor must be compensated for the time value of his capital and for the risk of losing the investment, also known as risk premium. Jensen's alpha (Jensen, 1968) measures how much a fund outperforms a baseline investment strategy, whereas beta measures how volatile or risky an investment is compared to the baseline investment strategy. These measures are all adjusted by a risk free interest rate, which is a tunable parameter.

Notation	Explanation
α_a	alpha, how much the strategy outperforms the baseline, risk adjusted
r_a	returns of portfolio or own strategy
r_f	risk free returns, 3 month UK treasury bond yields in sterling
r_m	market or baseline returns, always buy or buy and hold
β_a	beta, how much riskier the strategy is compared to the baseline
S	sharpe, how much excess return is not attributed to excess risk
σ	covariance function

Table 3.1: Fund performance metrics summary

$$\alpha_a = r_a - (r_f + \beta_a(r_m - r_f)) \tag{3.18}$$

$$\beta_a = \frac{\sigma(r_a, r_m)}{\sigma^2(r_m)} \tag{3.19}$$

$$S = \frac{r_a - r_f}{\sigma(r_a)} \tag{3.20}$$

In Table 3.1.7 and Equations 3.18–3.20, we list a few metrics measuring various aspects of fund performance. Alpha measures the return on investment in percentage of a fund and subtracts from it the return on investment of a baseline strategy. The excess returns α_a denotes the fund performance, r_a the expected or residual return, r_f the risk free interest rate or time value of money, and $\beta_a(r_m - r_f)$ the risk premium. This can be further broken down into β_a the sensitivity of the excess market rate of return over the risk free interest rate, $r_m - r_f$.

The average yearly return measures the fund's performance throughout a period and averages it on an annual basis. The Sharpe ratio (Sharpe, 1998) shows excess return adjusted by risk. The r-squared metric measures how much two return series move together. The standard deviation indicates the variability of values in a given return series. There is no single gold measure that gives a good generalisation performance for the future, rather a combination of these measures are often used to gauge the effectiveness of the trading strategy.

In this section, we have covered linear models and given an introduction as to how a trading algorithm may be constructed and evaluated using this framework. In the next two sections, we focus on non-linear machine learning models, in particular Gaussian Processes and Neural Networks. As financial markets are highly dynamic and non-linear, these machine learning frameworks could greatly boost the performance of the system, though care must be taken not to overfit the training data with these powerful high dimensional models.

3.2 Gaussian Processes

In the previous section, we reviewed linear models and showed how they could be adapted and applied to financial trading. In this section, we explore Bayesian models, in particular Gaussian Processes (GP). This framework allows us to not only learn inherent non-linearity in the data, but also incorporate uncertainty, where we could make more robust predictions by exercising multiple probable hypotheses at the same time in the model, as opposed to just taking the single most likely solution.

In frequentist statistics, an unknown parameter is assumed to have one true value and out of some number of experiments, at least a subset of these will contain the true value. On the other hand, in Bayesian statistics the unknown parameter is assumed to be generated from a hidden distribution by some process. While frequentists try to learn some fundamental property of the universe, Bayesian statisticians regard probability distributions as degrees of subjective belief, which can even be in conflict with each other. As such, Bayesian statistics often assumes a prior belief about a random variable or event, and given some evidence, it updates that belief as a rational agent. This is summarized by Bayes theorem, which states that the posterior probability of our belief (B) in an event given the evidence (E) is the product of a prior belief and the likelihood of evidence given our belief divided by the marginal likelihood of the evidence.

$$P(B|E) = \frac{P(E|B)P(B)}{P(E)} \tag{3.21}$$

Since markets are highly complex systems, often laws are uncovered by empirical data analysis (Gopikrishnan et al., 1999). Besides incorporating subjective beliefs, Gaussian Processes can also model non-linear systems, such as financial markets. This is done by the kernel method, where similarity between data points is defined by a kernel, such as the Radial Basis Function kernel (RBF). Another characteristic of markets is thought to be non-stationarity, which Gaussian Process can model well too via the same kernels. Interpolation and limited extrapolation of data at any point in time is also possible. Gaussian Process can determine feature relevance, which is highly useful in financial markets in order to decide which indicators are important. By the central limit theorem, the random walk of i.i.d. stock prices assumes a Gaussian distribution, which GPs can model, even though this may be a simplification of actual price movements. A Levy process may be also suitable, and can be simulated with a non-smooth Brownian kernel Gaussian Process. This allows for abrupt jumps in prices often seen in markets, rather than smooth changes. Special cases of Gaussian Process are the Weiener and Kalman filters that could give linear complexity is time via spatio-temporal modelling of market prices.

A Gaussian Process is a stochastic process. Each realisation in this process is considered a random variable with a normal distribution. GPs can model both spatial and temporal datasets. In addition, any finite linear combination of the random variables or samples of the GP is also jointly Gaussian distributed (Alvarez et al., 2011). GPs can be viewed as an interconnected probabilistic graphical model shown in Figure 3.2, and they can be employed for non-parametric regression problems. In order to define a GP, we must define the covariance

k and mean m functions as follows,

$$m(x) = \mathbb{E}[f(x)] \tag{3.22}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{3.23}$$

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{3.24}$$

In this framework, we assume that the prior of the latent functions f is given by a zero mean GP with $\mathbf{K}_{N\times N}$ covariance matrix. This is calculated by applying a given valid covariance function and evaluating it between pairs of datapoints. Given data observations, the noise model or the likelihood can be expressed as,

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2 I_N) \tag{3.25}$$

which applies an identical noise variance to each function realisation. Integrating over the latent function realizations $\mathbf{f}$, the evidence or the marginal likelihood can be written as.

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})\mathrm{d}\mathbf{f} = \mathcal{N}(0, \mathbf{K} + \sigma^2 I_N). \tag{3.26}$$

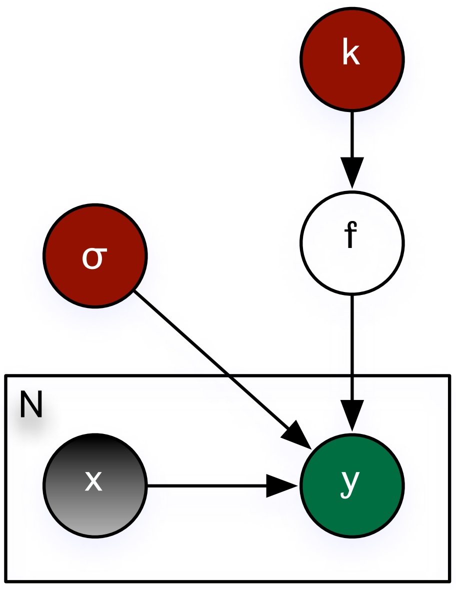


Figure 3.2: Gaussian Process plate diagram, where $\mathbf{x}$ is the training data, y the target, both replicated N times for each data point pair and f is a latent function. The targets are generated by evaluating $f(\mathbf{x})$ and adding Gaussian noise with variance σ^2. Finally, k is the covariance function, which together with a zero mean, specifies the $\mathcal{GP}$ prior distribution over f.

3.2.1 Predictive Distribution

Once we defined a GP, we need to make predictions from the training data. In order to generalise and perform prediction at new examples x_* given the training set, we need to compute the density of the latent functions at the test point, $p(f_*|\mathbf{y}, \mathbf{X}, x_*)$. We note that the training set $\mathbf{f}$ and the predictive latent functions f_* are jointly Gaussian $[\mathbf{f}, f_*] \sim \mathcal{N}(0, \mathbf{K}_*)$, with

$$\mathbf{K}_* = \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\intercal & k_{**} \end{bmatrix} \tag{3.27}$$

where the covariance vector between the test point x_* and the training data X is given by $\mathbf{k}_* = k(\mathbf{X}, x_*)$. The marginal variance at the test data can be written as $k_{**} = k(x_*, x_*)$. In case we omit noise, we can then write the predictive distribution as $p(f_*|\mathbf{y}) = \mathcal{N}(\mu_*; \Sigma_*)$ with,

$$\mu_* = \mathbf{k}_*^\intercal(\mathbf{K} + \sigma^2 I)^{-1}\mathbf{y} \tag{3.28}$$

$$\Sigma_* = k_{**} - \mathbf{k}_*^\intercal(\mathbf{K} + \sigma^2 I)^{-1}\mathbf{k}_*. \tag{3.29}$$

We can add noise to the predictive distribution $p(f_*|\mathbf{y}) = \mathcal{N}(\mu_*, \Sigma_*)$ or $p(y_*|\mathbf{y})$ by simply adding σ^2 to the variance Σ_*. As a result, the predictive mean becomes

$$\mu_* = \sum_{i=1}^{N} a_i k(x_i, x_*) \tag{3.30}$$

where a_i is the i^{th} component of $\mathbf{a} = (\mathbf{K} + \sigma^2 I)^{-1}\mathbf{y}$. This demonstrates that the prediction of $\mathbf{y}$ in the GP involves the linear combination of N number of basis functions, where each function is centred around a training example. However, what this means is that optimising the hyperparameters of the GP requires the inversion of the $N \times N$ covariance matrix. This operation has complexity $O(N^3)$, and for large scale learning problems may be impractical to do. Though there are ways to reduce this complexity, down to $N \times M$ where N is the number of data points and M is the number of inducing variables (Csató and Opper, 2002).

3.2.2 Covariance Functions

A cornerstone of GPs is the covariance function applied to the data. In kernel based nearest neighbour methods like the GP, similar inputs should yield similar outputs. Similarity is hereby defined by the covariance function, with some conditions.

In the GP framework, it is assumed that the functions $\mathbf{f}$ follow a multivariate Normal distribution. This dependency is defined by the covariance matrix $\mathbf{K}$. Therefore, one of the most important components of GPs is the covariance matrix because it defines the correlation of the random variable at the inputs x. We can create multiple covariance functions by parametrizing and evaluating over pairs of training data instances $k(x, x')$. A covariance functions must satisfy the constraint of being symmetric positive semi definite (Williams and Rasmussen, 2006),

$$\mathbf{v}^{\intercal}\mathbf{K}\mathbf{v} \geq 0, \forall \mathbf{v}. \tag{3.31}$$

Parametric covariance functions give us the ability to infer these parameters from the data. In a Bayesian linear regression model, these parameters are called hyperparameters θ, and they are responsible for controlling the distribution of the model. Unfortunately, it is not possible to directly infer the optimal value of the parameters, but there are alternative methods. For example, one approach is to set a prior over the parameters and infer the posterior via sampling methods (Gelman et al., 2014). Alternatively, the prior can be directly optimized by maximizing the log posterior probability distribution. This is called Maxium a Posteriori approximation, which can be executed by approximating the posterior $p(\theta|\mathbf{y}, X) \propto p(\mathbf{y}|\theta, X)p(\theta)$, with a distribution that is centered at the mode. At this point $p(\theta|\mathbf{y}, X)$ is maximal,

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|\mathbf{y}, X) = \arg\max_{\theta} p(\mathbf{y}|\theta, X)p(\theta) \tag{3.32}$$

By following this procedure, we can find the optimal parameter value by making a point estimate, and maximising the log of the right hand side of Equation 3.32. One way to do this is to ignore any dependency on the prior and maximise the log marginal likelihood as shown in Equation 3.26. By doing this approximation and maximising the marginal likelihood, we obtain the type II Maximum Likelihood (ML).

How to choose the covariance function largely depends on the specific application and the features of the dataset. There are two classes of covariance functions: stationary and non-stationary. Stationary covariance functions are not affected by input space translations and they are of the form $k(|x - x'|)$. On the other hand, non-stationary covariance function are affected by it (Williams and Rasmussen, 2006). A commonly used covariance function is the squared exponential seen in Figure 3.3,

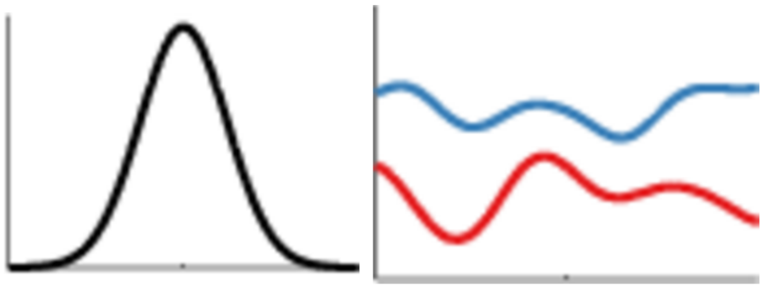


Figure 3.3: Visualization of the Squared Exponential covariance function and random realisations from a GP.

$$k(x_i, x_j) = \theta_0^2 exp\left\{-\sum_{k=1}^{d}\frac{(x_{ik} - x_{jk})^2}{2\theta_1^2}\right\} \tag{3.33}$$

where the parameters θ_0 and θ_1 are the amplitude and the characteristic lengthscale respectively. The SE can be modified by allowing a different lengthscale parameter per input dimension, which allows for Automatic Relevance Determination (Neal, 1995). ARD is useful for the identification of important features in the dataset and automatically upweighing or downweighing them. Non-stationary covariance functions are of the dot product form $k(x \cdot x')$. One example of this is the linear kernel,

$$k(x, x') = \theta_0^2 + \theta_1^2\sum_{k=1}^{d} x_{ik}x_{jk}. \tag{3.34}$$

New covariance functions can be derived by adding, multiplying or convolving existing valid covariance functions (Williams and Rasmussen, 2006).

A key advantage of using a Gaussian Process for predictions is that it does not require the input features to be i.i.d. since it can model correlation through joint Gaussian distributions. In addition, with a Gaussian Process we can make inference at any point in time. It can also model many other machine learning algorithms and their Bayesian interpretations, such as Linear regression, Kalman filters, Wiener filters, or even neural networks (Neal, 1995). With a Gaussian Process, we can declare a prior distribution over the data which can be overridden during training if there is sufficient amount of evidence for it in the training data. GPs are also very good at modelling uncertainty at any stage and since the complexity of the GP is not dependent on the dimensionality of the input, it is ideal to work with high dimensional datasets (Skolidis, 2012).

Some disadvantages of GP are that it requires long training time when the dataset is large, and there are many datapoints, since the covariance matrix can be quite large, which denotes the complexity of inference. In addition, there may be occasional numerical instabilities in the Cholesky decomposition of the positive semi definite covariance matrix, $\mathbf{K}$. This is typically solved by the addition of jitter. There are Matlab and python implementations of the algorithm.[2] GPy provides functionality to declare custom kernels that can be added and multiplied together provided that they satisfy the requirement of being positive semi definite. In addition, custom likelihoods with link functions can also be implemented. In chapter 5 we provide examples and experiments to integrate custom kernels and likelihoods in the GP framework in order to predict financial markets.

3.2.3 Multi Task Gaussian Process

In section 3.1.2, we saw a way to incorporate multi task learning into linear models via regularisation techniques. In this section, we examine how we can transfer knowledge from related tasks in the GP framework.

Considering a multi-task scenario, we would like to learn M functions f_j, from data $X_j = [x_{1j}, ..., x_{nj}], \mathbf{X} = [X_1, ..., X_M]$, and targets $\mathbf{y}_j = [y_{1j}, ..., y_{nj}]$ with $j = 1, ..., M$ and $n_1 + ... + n_M = N$. Similarly to Single Task Learning, we assume the following noise model, where $y_{ij}(x_{ij})$ denotes the i^{th} task and ϵ_j is task dependent noise,

$$y_{ij} = f_j(x_{ij}) + \epsilon_j, \epsilon_j \sim \mathcal{N}(0, \sigma_j^2). \tag{3.35}$$

The key idea behind multi-task learning is that by simultaneously training all M models, we can facilitate intelligent information transfer between tasks. To obtain this outcome, we can allow some structure of the parameter space to be shared across tasks. It is likely the case that

[2]GPy - `http://sheffieldml.github.io/GPy/`

for tasks where training data is scarce, inferring the common parameters of the models provides more benefits than for tasks where data is plentiful. That is because with lots of training data, even a single learning task has sufficient information to infer the optimal model parameters on its own. Another important issue to consider is how related the tasks are. Lots of approaches take this for granted but in practice little relations may exist. However, it has been shown that even by randomly subdividing a STL problem into a MTL one, performance improvements can be made.

Multi-task learning can be cast as a multi-response or mutli-output problem. In this special case of MTL, the inputs for all tasks are shared, while the outputs are not (Chen et al., 2012). In multi-response problems, the output on a data point can be seen as a vector valued function where each dimension is the output for each task, $\mathcal{Y} = \mathbb{R}^M$. This problem has been extensively studied before in the geostatistics community where it is often referred to as co-kriging. An alternative way to explore this problem is to say that we are interested in the distribution over the latent functions as follows $\mathbf{F} = [\mathbf{f}_1 ... \mathbf{f}_M], \mathbf{F} \in \mathbb{R}^{N \times M}$, and subsequently the matrix of the outputs are $\mathbf{Y} = [\mathbf{y}_1 ... \mathbf{y}_M], \mathbf{Y} \in \mathbb{R}^{N \times M}$. It is possible to derive this matrix variate Normal distribution via the Intrinsic Model of Coregionalisation (IMC), where a more general formulation of this method is known as the Linear Model of Coregionalisation (LMC) (Goovaerts, 1997). In LMC, we assume that each task output is a linear combination of independent random functions. The relationships between the various tasks are then captured by a separate positive definite matrix. A further approach to tackle the multi-response problem is done with the use of Convolution Processes (CP), where it is possible to construct multi-output covariance functions (Ver Hoef and Barry, 1998). In this case, the outputs are derived as a convolution integral between the latent function and a smoothing kernel. In all of these approaches, missing outputs for some tasks can be straightforwardly handled by ignoring the likelihood terms for the unobserved outputs. This way, both the IMC and CP methods can be applied to multi task learning (Alvarez et al., 2011; Skolidis, 2012).

As we have seen, Gaussian Process can be used for multi task learning as well as to produce multiple simultaneous outputs (Skolidis, 2012; Bonilla et al., 2007). One way to do this is through the use of coregionalisation kernels. The automatic relevance determination kernel can be used to automatically determine the lengthscale of each input feature. Then we can multiply the coregionalisation kernel with an ARD kernel, and the resulting kernel will have the dimensionality of the input space. The inputs can be either shared or not shared between tasks.

Manifold relevance determination assumes that there is a hidden process that generates the data from some unknown distribution. The multi task setting means that from the same hidden distribution, the data is generated in different ways. Some parts of the data is related and shared among tasks, which are called shared latent space variables, whereas other features are private for each task instance. MRD could be used to generate inputs from targets and vice versa (Damianou and Lawrence, 2012; Damianou et al., 2012).

One application of multi task GPs is modelling annotator bias (Cohn and Specia, 2013) which shows that Gaussian Processes using a multi task kernel have a better accuracy in predicting annotator time in machine translation. Task division was performed on a per user basis. We assume that this kernel can be applied to modelling financial markets where task relatedness could be automatically determined by industry or sector. In the following sections, we describe some ways we can include custom objectives (pseudo likelihoods) in the GP via approximations that may enable us to encode the profit in a financial context.

3.2.4 Model Selection

L2 regularisation that aims to keep the weights low and prevent overfitting is already incorporated in the Gaussian Process context through the Bayesian interpretation of Tikhonov regularisation (Tikhonov, 1943). Standard GP regression with a Gaussian prior is equivalent to minimising the mean squared error with the squared L2 norm of the weights as penalty. In addition, imposing a Laplacian prior yields regularised regression with an L1 penalty in a similar fashion. During maximum likelihood estimation, the coefficients of regularisation are determined by the GP hyperparameters as a form of Bayesian model selection. For example, if the GP were to overfit the data by assigning low variance to it and making the lengthscales also very small, the marginal likelihood will be low even though the model perfectly fits the dataset. That is because in this setting, the GP may explain a subset of the data very well, but completely fail to explain some other datapoints. Similarly, if the GP chose to fit a wide variety of data, then due to the likelihood terms spreading out too much with high variance, the resulting marginal likelihood will be again small. This is illustrated in Figure 3.4. In the

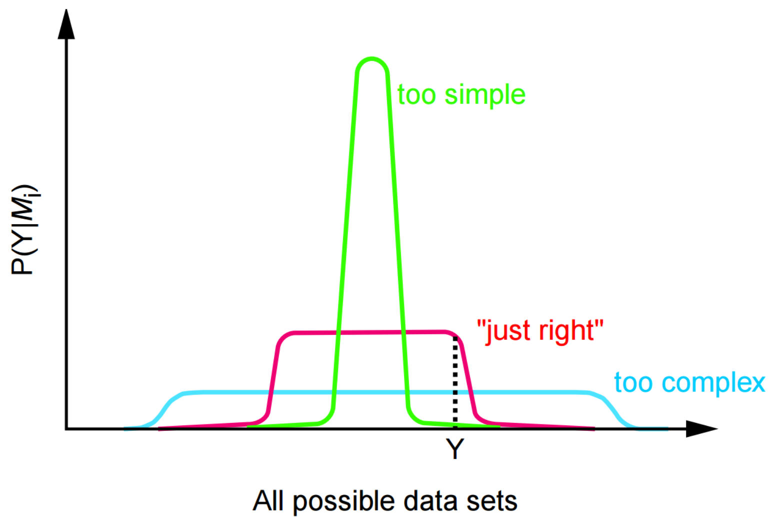


Figure 3.4: Gaussian Process marginal likelihood. Both the too simple and too complex models assign incorrect likelihood to the data observed at Y, where the right value is in between.

linear context, this would result in a really small cost function output, so the L2 regularisation needs to be tuned by crossvalidation to obtain a good tradeoff between fitting the dataset and generalisation. Crossvalidation for GPs can still be used for higher level model selection problems, such as deciding on kernels and likelihoods. Alternatively, the marginal likelihood of the model considered can be employed to pick optimal hyperparameters without the need of a separate crossvalidation dataset. The GP marginal likelihood also contains a term that penalizes model complexity, thereby favouring simpler models implicitly. The hyperparameters of the GP can be optimised using L-BFGS or Stochastic Gradient Descent. This results in a fitted posterior around the data points, as seen in Figure 3.5. In the figure we can see random draws from the GP prior, which cluster around the datapoints in the posterior. Fitting the model hyperparameters results in a close fit around the data. The marginal likelihood can be written as,

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^{\intercal}(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}+\sigma^2 I| - \frac{N}{2}\log 2\pi + \log p(\theta) \tag{3.36}$$

where the first term attempts the maximise the fit of the data, the second term limits the volume of the determinant, having the effect of limiting model complexity, and the rest of the terms are normalizing constants and the prior likelihood. This shows that GPs implicitly trade off model complexity with data fit, facilitating Bayesian model selection.

On another practical note, often models still require parameters to tune their configuration, now aptly called hyperparameters, once the original weights or parameters are marginalised.

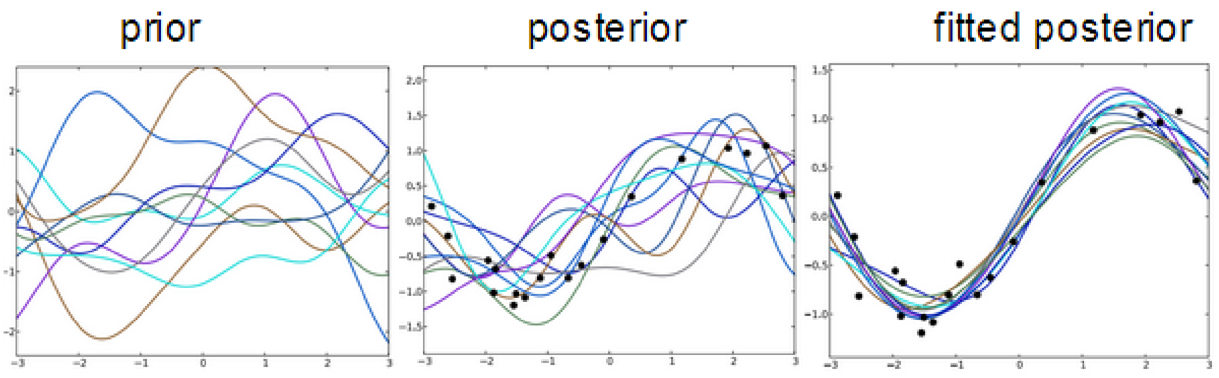


Figure 3.5: Gaussian Process regression before data is observed (prior), after data is observed (posterior), after maximising the marginal likelihood (fitted posterior.)

Due the fact that the original parameters are no longer present in the final model, these family of models are called non-parametrics. We can repeat the process of marginalisation for the hyperparameters as well by introducing new hyper-hyper-parameters. It is dependent on the user how many layers deep they want to marginalise the parameters. In this thesis, we only deal with one level of non-parametric models.

3.2.5 Sparse Gaussian Process

When the dataset is too large, it is infeasible to compute the full covariance matrix and in this case sparse approximations can be used instead. This is done via a number of inducing points that are representative of the dataset. The location of these inducing points then can be optimised such that the likelihood is approximated by a lower bound using Jensen's inequality, or alternatively the Kullback-Leibler divergence of the true distribution and the approximation can be minimised. Hence, the complexity of the computation is reduced from n^3 to nm^2 where n is the number of datapoints and m is the number of inducing variables.

$$\mathbf{K_{nn}} \approx \mathbf{K_{nm}} \mathbf{K_{mm}^{-1}} \mathbf{K_{mn}} \tag{3.37}$$

The likelihood can then be written as

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) \tag{3.38}$$

$$\ln p(\mathbf{y}|\mathbf{u}) \geq \mathrm{E}_{p(\mathbf{f}|\mathbf{u},\mathbf{X})}[\ln p(\mathbf{y}|\mathbf{f})] \tag{3.39}$$

where $\mathbf{u}$ is the functional at the inducing points. The location of the inducing points can be initialised either by choosing m datapoints from a random permutation of the input data, or by K-means clustering. If $m = n$ then we should retrieve the original dense Gaussian Process model. To take the concept of Sparse GP further, Stochastic Variational Inference can be performed with GPs using natural gradient to make approximation of the covariance matrix with m^3 complexity (Hensman et al., 2013a).

Sparse Gaussian Process requires the input of some inducing variables. These can be provided by K means clustering on the input data for each task. Then the Gaussian Process tries to approximate the full covariance matrix from the inducing points. In this case, the GP has degenerate covariance matrix with the inducing points being the bottleneck. This is useful in settings where there are a lot of datapoints in the training set, and therefore the Gaussian Process training algorithm could run out of memory when the full covariance matrix is computed. This way, the Gaussian Process can be scaled to big data problems (Hensman et al., 2013a). GPs can also perform online learning (Csató and Opper, 2002) and classification is possible with Gaussian Process through the use of link functions and Gaussian approximations of the Bernoulli likelihood. Effective approximations include the Laplace Approximation, which expands a local Taylor series approximation of the target distribution around the Maximum-a-Posteriori estimate of the parameters. Alternatively, Expectation Propagation can be used that matches various moments of the target distribution. Many of these techniques are compatible with Sparse GPs.

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining (Hartigan and Wong, 1979). K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. This algorithm can be used to initialise the inducing points in sparse GPs.

In this section we have covered sparse approximation of the covariance matrix to scale the GP to large datasets, such as ones found in financial markets. In the next section we review approximations for various likelihoods in order to incorporate custom objectives into the GP framework, such as profit maximisation.

3.2.6 Likelihood Approximation

In section 3.1 we have introduced linear models that work by taking the Maximum a Posterior parameter estimate of the weights distribution. However, given the MAP point estimate solution for the profit maximisation objective, it is possible to sample the full range of possible weights values from a Gaussian approximation of the entire weights distribution. Due to the fact that the MAP solution corresponds to the Maximum Likelihood solution given enough data, once almost all prior information has been overridden by new evidence, we can make predictions based on only one "optimum" point estimate value of the parameters. However, at this point by taking a single point estimate we are discarding information that might be useful for predictions. One example is where the parameter distribution is bi-modal as seen in Figure 3.6. To counter that, predictions should be made by integrating out the entire distribution of the weights, also called marginalisation of the parameters. In practice, in case the integral is intractable, we can still draw samples from an approximation of the true weights distribution that could give us useful information vis-a-vis the underlying process that generated the data, which then can be averaged to derive a single prediction per data instance.

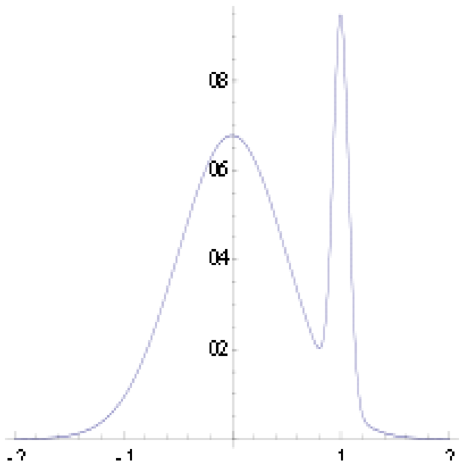


Figure 3.6: Example bimodal parameter distribution where MAP solution is not perfect.

To limit information loss, given the cost function and the MAP solution, we can perform a local Taylor series expansion of the weights distribution function in the Laplace approximation method (Rogers and Girolami, 2011) shown in Figure 3.7. In mathematics, a Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. Hence, the Taylor series approximates the curvature of any function by a polynomial. The higher order the polynomial, the better the approximation is. In practice, a 3rd order polynomial can give fairly accurate local expansion of the function we are interested in. The Taylor series approximates the function the following way

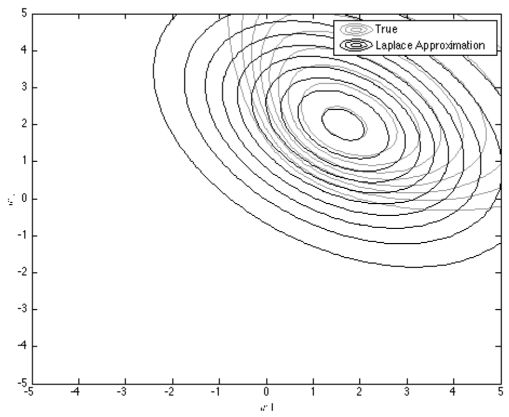


Figure 3.7: Laplace approximation of a non-Gaussian distribution.

$$f(\mathbf{w}) = \sum_{n=0}^{\infty} \frac{(\mathbf{w} - \hat{\mathbf{w}})^n}{n!} \frac{\partial^n f(\mathbf{w})}{\partial \mathbf{w}^n}\bigg|_{\hat{\mathbf{w}}} \tag{3.40}$$

where $\frac{\partial^n f(\mathbf{w})}{\partial \mathbf{w}^n}\Big|_{\hat{\mathbf{w}}}$ is the nth derivative of $f(\mathbf{w})$ with respect to $\mathbf{w}$, evaluated at $\hat{\mathbf{w}}$. Similarly, if we discard all terms above order 3, we get the approximation of the posterior weights distribution which is a multivariate Gaussian probability distribution, with mean μ and variance $\mathbf{\Sigma}$,

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^{\mathbf{2}}) \approx \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{3.41}$$

where μ is set to $\hat{\mathbf{w}}$ and $\mathbf{\Sigma}$ is the negative of the inverse Hessian 2nd order partial derivatives matrix:

$$\mu = \hat{\mathbf{w}} \tag{3.42}$$

$$\mathbf{\Sigma^{-1}} = -\left(\frac{\partial^2 \ln \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y})}{\partial \mathbf{w} \partial \mathbf{w}^{\intercal}}\right)\bigg|_{\hat{\mathbf{w}}} \tag{3.43}$$

where $\mathcal{L}$ is the likelihood function, or alternatively the cost function can be used too.

An alternative to Laplace approximation is Expectation Propagation (EP). EP in a Sparse Gaussian Process setting works by local moment matching (Minka, 2001). If the link function is non-Gaussian then the posterior distribution will not be Gaussian either. Therefore, both the Laplace approximation and the Expectation Propagation algorithm try to approximate the non-Gaussian posterior with a Gaussian distribution. However, Expectation Propagation works in a sparse setting too. In EP, the Kullback-Leibler divergence between the approximate and true distribution is minimised where

$$q(\mathbf{f}_*|\mathbf{y}, \mathbf{y}_*, \mathbf{X}, \mathbf{x}_*) = \underset{q(\mathbf{f}_*|\mathbf{y}, \mathbf{y}_*, \mathbf{X}, \mathbf{x}_*)}{\operatorname{argmin}} \mathbf{KL}(p(\mathbf{f}_*|\mathbf{y}, \mathbf{y}_*, \mathbf{X}, \mathbf{x}_*) \| q(\mathbf{f}_*|\mathbf{y}, \mathbf{y}_*, \mathbf{X}, \mathbf{x}_*)) \tag{3.44}$$

where q is the approximate distribution of functionals. To find the minimum of the above approximation, we eventually need to replace the likelihood term with a Gaussian distribution,

$$p(\mathbf{y}_*|\mathbf{f}_*) = \mathcal{N}(\mathbf{m}_*|\mathbf{f}_*, \beta_{\mathbf{m}}^{-\mathbf{1}}). \tag{3.45}$$

where $\mathbf{m}_*$ and $\beta_{\mathbf{m}}^{-\mathbf{1}}$ and the first and second order moments of the distribution that are matched. EP approximates the belief states by only retaining expectations, such as mean and variance, and iterates until these expectations are consistent throughout the network. In general, Expectation Propagation unifies assumed-density filtering, an extension of the Kalman filter, and loopy belief propagation, an extension of belief propagation in Bayesian networks (Minka, 2001).

3.2.7 Bayesian Utility Maximisation

In this section, we outline a potential way to incorporate arbitrary cost functions inside the Bayesian learning framework. For example, in the stock market trading scenario, we can capture the MAP solution of a profit maximisation objective, and calculate the Hessian at that point. Given the negative of the inverse Hessian and the MAP, we can approximate the posterior weights distribution locally, from which we can sample to produce predictions. Performing Monte Carlo simulation around 10,000 times for each prediction, and then averaging the predictions takes into account more information about our learning problem and thus results in better forecast accuracy when training data is limited. In chapter 5.1 using the Laplace approximation, we show experiments to obtain better predictive performance on the stock dataset, where we predict performance on the last trading month by training the model on previous months across companies.

To take the Bayesian treatment of the objective further, we can implement the stock market profit maximisation objective in the context of Gaussian Processes, which is an interconnected probabilistic graphical model on the latent functions $\mathbf{f}$ and the targets are only dependent on

$\mathbf{f}$ instead of the input $\mathbf{X}$. We convert the cost function to a pseudo likelihood function, which has the restriction of being positive. Due to the fact that there is the possibility of obtaining negative profit during training, one way to circumvent this is to shift our profit calculation such that in the worst case scenario the limit of the profit goes to 0. This can be done by calculating the profit by the product of balance changes on each day which is $1 + r$ where r is the returns for that day, denoted by $\mathcal{L}_2$. An alternative way to convert the cost function to a likelihood is to treat the sum of the daily returns as the log likelihood and take the exponential of that in order to obtain a positive profit function $\mathcal{L}_1$. Therefore, the log likelihood of the profit maximisation function can be written as follows

$$\ln \mathcal{L}_1(\mathbf{y}|\mathbf{f}) = \sum^{N}(l(\mathbf{f})\mathbf{y}) \tag{3.46}$$

$$\ln \mathcal{L}_2(\mathbf{y}|\mathbf{f}) = \ln \prod^{N}(l(\mathbf{f})\mathbf{y} + 1) \tag{3.47}$$

where N are the number of trading days that collectively make up all the data points in the dataset, $\mathbf{f}$ is a functional that can be evaluated at a given datapoint and $\mathbf{y}$ are the target returns for a given trading day. By using a Radial Basis Function kernel to create the covariance matrix of the GP, the $\mathbf{w}^\intercal\mathbf{x}$ in a linear model can be replaced by the GP kernel functional. l is the link function that is obtained by reversing the Hyperbolic tangent activation function, $l(x) = tanh(x)$, denoting investment (+1) or divestment (-1). The derivatives of both the link function and the likelihood function can be either derived manually, or via automatic symbolic differentiation, up to the 3rd derivatives. When Automatic Relevance Determination is enabled, it allows for the learning of a different lengthscale for each input dimension, which allows for the isolation of influential trading patterns. In the GP setting there is one model weight for each datapoint as opposed to one model weight for each input feature, however, both the GP and a linear model attempt to find the linear combination of datapoints and weights that gives the hyperplane that fits the data the best. The tanh activation function derivatives used for Laplace approximation are as follows

$$\frac{\partial \mathbf{f}}{\partial} \tanh(\mathbf{f}) = \text{sech}^2(\mathbf{f}) \tag{3.48}$$

$$\frac{\partial \mathbf{f}^2}{\partial^2} \tanh(\mathbf{f}) = -2\tanh(\mathbf{f})(\text{sech}^2(\mathbf{f})) \tag{3.49}$$

$$\frac{\partial \mathbf{f}^3}{\partial^3}\tanh(\mathbf{f}) = 2(\cosh(2\mathbf{f}) - 2)(\operatorname{sech}^4(\mathbf{f})) \tag{3.50}$$

In the linear model, we can take non-stationary into account via a multi-task time regulariser. In the Gaussian Process setting, time can be taken into account as another input dimension of the RBF ARD kernel. Therefore, time points that are close to one another will have high covariance, whereas time points that are far away, will have low covariance, as determined by a local Gaussian distribution with a given lengthscale and variance. In the RBF kernel, each input dimension covariance is multiplied together to obtain the overall covariance of two data points. This justifies the implementation of non-stantionarity as just as another input in the RBF kernel.

To take relationships between multiple companies into account, a multi task mean regulariser is used in the linear model. In the Gaussian Process context, we can apply the coregionalisation kernel for each company, by taking the Kronecker product of the RBF kernel and the coregionalisation matrix, using a block design method. This is also called the intrinsic model of coregionalisation. The design matrix can be factorised by Cholesky decomposition which yields the Gram matrix (**W**). In the simplest case, the Gram matrix has only one column, by increasing the rank of the decomposition matrix, we can increase the complexity of the factorisation, and hence more complex relationships can be obtained between companies. It is not recommended to increase the rank above the number of tasks in the dataset as this could lead to an ill-posed problem. Task independencies can be taken into account by combining the Gram matrix with an identity matrix with diagonal coefficients κ.

The Kronecker product of two matrices can be written as,

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \tag{3.51}$$

The IMC covariance matrix accross companies, weighted by temporal market changes can be described as follows,

$$\mathbf{K} = \mathbf{B} \otimes (RBF(\mathbf{x}, \mathbf{x}') \times RBF(t, t')) + white(\mathbf{x}, \mathbf{x}') \tag{3.52}$$

$$RBF(r) = \sigma^2 \exp\left(-\frac{1}{2}r^2\right) \tag{3.53}$$

$$\mathbf{B} = \mathbf{W}\mathbf{W}^\intercal + \mathrm{diag}(\kappa) \tag{3.54}$$

$$white = \mathrm{diag}(\sigma) \tag{3.55}$$

where r is the scaled distance between two datapoints, $\mathbf{B}$ is the coregionalisation matrix across companies, $white$ is diagonal noise, x is the input datapoint, i.e. stock prices, and t is time. This has the effect of down weighing the influence of trades that happen far apart in time $RBF(t, t')$, multiplied by similarity of the trading pattern observed $RBF(x, x')$, and this is weighted by cross-company correlations across the entire market $\mathbf{B}$, while accounting for some independent processes in each company $\mathrm{diag}(\kappa)$, with some added noise in trading behaviour $white$.

3.2.8 Comparison to Discriminative Models

Next, we give a brief comparison of GPs and Neural Networks (NNs) that complements the comparison in the end of section 3.3.

While linear models are considered discriminative seeking to model the conditional distribution of targets given our data, $p(y|X)$ with some parameter estimates θ, Bayesian frameworks are generative, and instead model the joint distribution of the targets and the data together $p(y, X)$ while integrating over a distribution of possible parameter values. This makes them a non-parametric class of models. The latter approach is considered more robust as it also includes the calculation of a (usually intractable, thus requiring approximation) term for the marginal likelihood, $p(X)$, which counteracts overconfident predictions far from the manifold of the observed data. Although one of the cited advantages of discriminative models, i.e. neural networks seen in 3.3, is non-local generalization, this property can under specific circumstances backfire and give highly confident predictions for incomprehensible data (Nguyen et al., 2015). Therefore, in this case, Bayesian models remain an attractive choice. GPs also have a unique way of solving the intractable problem of calculating $p(X)$ by assuming a Gaussian distribution where analytical solution exists for integrating over all possible hypotheses. On the other hand, NNs make no such assumption.

Another difference between GPs and NNs is the fact that with Neural Networks, the training is relatively fast, but predictions can be slow as it requires the propagation of the signal throughout the entire network. With Gaussian Processes, the training is slow given the calculation and inversion of a similarity (covariance) matrix, often with complexity $O(n^3)$, but predictions are fast as being a nearest neighbour type model, it just involves the lookup of the nearest neighbours from the precomputed covariance matrix and interpolating between nearby labels. It is also generally the case that storing all the data points in nearest neighbour type models is very memory intensive, while neural networks are more computationally expensive instead as they require multiple subsequent projections of data. That said, GPs can be equivalent to a one infinitely wide hidden layer neural network with a specific kernel. In other words, a neural network with a single hidden layer and infinetely many hidden units with a special activation function in fact corresponds to a Gaussian Process with a specific covariance kernel in case there is a probability distribution over the neural network weights (i.e. Gaussian) (Neal, 2012).

This concludes our overview of Bayesian non-parametric models. For experiments making use of these models in a financial setting, please refer to chapter 5. In the next section, we move on to reviewing Neural Networks that have achieved great breakthroughs in machine learning in recent years.

3.3 Neural Networks

So far we have seen Bayesian and frequentist models when we reviewed linear models and Gaussian Processes. In this chapter, we review Neural Networks that are able to construct multiple deep non-linear projections of the data.

Artificial neural networks are nature inspired computational models. They resemble neurons found in the brain and are capable of pattern recognition by feeding information throughout the network. For example, hand writing can be recognized by activating the neuron with raw pixel data of a digit. After the information propagates throughout a custom designed architecture, the final output neuron can determine the type of digit. Neural networks, just like other machine learning methods, excel at problems that are traditionally hard to solve using rule-based programming. For example, computer vision, speech recognition or natural language understanding.

Deep learning is used to denote multi layer neural networks that can learn hierarchical concepts and representations of data (LeCun et al., 2015). It has achieved good results in speech recognition, object detection, natural language processing, drug discovery and genomics. Backpropagation is used to train deep models in order to adjust its parameters layer by layer and make prediction errors smaller. Further, convolutional and recurrent nets have both improved the state of the art in image recognition and natural language processing. Convolutional nets allow the network to be invariant to shifts in the image, while recurrent nets can process sequential data such as text and time series easily.

Neural Networks are ubiquitous in society, and can be found in web searches, social networks, content filters, e-commerce, and even in consumer products such as smartphones and cameras. For example, it may transcribe speech to text, identify objects, recommend products or news items, and return relevant search results. Deep learning has gained traction in all these areas. Traditionally, machine learning techniques require feature engineering in order to convert raw data into a form that can be fed into the learning system. This often requires domain expertise and it is very difficult to perform. Then the system would perform regression or classification on this feature representation. In contrast, representation learning allows for the automatic learning of features with respect to the end task. In deep learning, we learn multiple levels of representations, where higher level concepts are composed of lower level ones. As a result, quite complex abstract functions can be modelled. On the highest level, discriminating features of the data can be amplified, while other feature variations suppressed. For example, for an image the first layer may represent raw pixels, which are transformed to extract edges at different locations and orientations. Then these edges can compose motifs and larger more familiar objects. At the end, objects could be detected by the network.

In deep learning, the layer representations are not designed by humans, but instead learned by a general purpose procedure. This makes tasks solvable that had been hard to tackle previously, as it can uncover intricate patterns in high dimensional data. This makes deep learning applicable to many domains in business, science and government. It has beat records in image recognition (Krizhevsky et al., 2012; Szegedy et al., 2015), speech recognition (Hinton et al., 2012), monitoring drugs (Ma et al., 2015), particle accelerators, brain circuits (Ciodaro et al., 2012), and disease mapping. In natural language processing (Collobert et al., 2011), breakthroughs include topic classification, question answering, sentiment analysis and machine translation (Sutskever et al., 2014). It is hypothesized that deep learning is also applicable to financial data where it can uncover patterns previously not seen by simpler learning algorithms.

3.3.1 Deep Learning

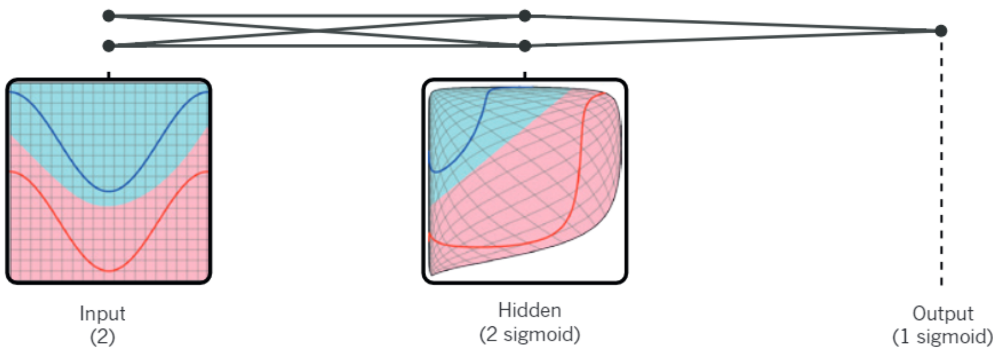


Figure 3.8: Input-output mapping with 2 sigmoid hidden units.

One common form of machine learning is supervised learning. Here we may be interested in classifying images as house, car or pet. First, we collect a large number of images with already labelled category. Then the machine produces a score for each category given how likely that image contains that category. Before training, it is unlikely that the correct category will have the highest score. We define an objective function that encodes our error between what the network predicts and what we want it to predict. Then we iteratively minimize the error by tuning the adjustable parameters of the network little by little. A deep learning system may have millions of these parameters, and likewise millions of labelled examples to learn from. Figure 3.8 illustrates the power of depth in a prediction scenario where the complex input data is first untangled via a non-linear projection onto a hidden layer, after which a simple final linear prediction can be made.

One of the most popular ways to train deep learning models is via gradient descent. In short, to perform the learning, the gradient has to be computed with respect to the parameters of the network. This tells us that by modifying the weights a little, how the error changes in the predictions. To minimize the error, we want to update the weights in the negative direction of the gradient. That is because the objective function can be regarded as a high dimensional hilly landscape and our goal is to find the minimum of it, averaged over many training examples. This is called the steepest descent optimization. A variant of steepest descent learning is stochastic steepest descent. Here, instead of evaluating the gradient on the entire training set, we evaluate on a single or a small number of examples. Then we pick a different sample of examples, and repeat the process until we see the objective function decrease on average. SGD usually yields faster convergence than full batch training, but it gives a noisier estimate of the gradient too (Bousquet and Bottou, 2008). Once training finishes, the generelization power of the system is measured on a separate test set. If there are some additional training parameters of the network, those are tuned on a third validation set. The test set serves the purpose of obtaining an accurate estimate of how the network may perform on examples it has not seen previously.

Previously, many systems consisted of linear classifiers or regressors on top of human engineered features. For example, for a two class problem, the classifier would compute a weighted sum of the input features plus bias, and if that sum is above a threshold, it would assign one of the categories to the input, if below, then assign the other one. Linear classifiers are only able to carve the input space into half regions separated by a linear hyperplane (Ro and Pe, 1973). However, many real world problems are more complex than that, and the learning algorithm needs to be robust to irrelevant variations of the inputs, for example in location, or orientation, or pitch of the speaker. At the same time, it needs to be sensitive enough to distinguish for example between a white wolf-like dog and a white wolf. At the pixel level, two white wolfs in two completely different environments may only marginally resemble each other other, whereas a white dog and a wolf in similar positions and backgrounds may be difficult to distinguish. A shallow classifier may not able to distinguish the latter two images, and it would require a good feature extractor to become selectively variant to certain changes of the input. At the same time, the classifier should remain invariant to other changes such as in the background. One way to make classifiers more powerful is to use non-linear features. These can be generated automatically via kernel methods (Scholkopf and Smola, 2001), such as with a Gaussian kernel as seen in section 3.2. However, one disadvantage of this is that generalizability is limited to the local neighbourhood region of the training data. Other options include considerable feature engineering with domain knowledge, which is the preferred method for linear models seen in section 3.1. However, with deep learning good features can be automatically learned from the data, making feature engineering or kernel method less relied upon. A deep learning architecture is simply a stack of simple non-linear modules (layers) with input and output mappings. Each module is responsible for only part of the overall transformation of the data.

With a depth of 5 or more, a deep learning system can represent extremely intricate functions that are selectively invariant to certain aspects of the inputs. For example, it could distinguish between two species of quadrapeds, but be invariant to pose or lightning. Training such deep networks had been a big challenge in the 90s when Neural Networks were first introduced. One algorithm for training is backpropagation, which we cover next.

3.3.2 Backpropagation

Replacing hand-engineered features with automatically extracted ones has been a long term aim of artificial intelligence research. However, until the 1980s, the solution had remained elusive, despite its relative simplicity. As it turns out, multi-layer architectures can be trained using stochastic gradient descent, as long as the gradients with respect to the weights can be computed and the error surface of the learning problem is relatively smooth (Rumelhart et al., 1988). The backpropagation procedure at its core is simply the application of the chain rule of differentiation. As such, the gradient with respect of the input of a computation module can be derived from the gradient of the output of that module. Then the chain rule can be repeatedly applied from the output prediction module all the way to the initial inputs of the network. Once these gradients are calculated, the gradients with the respect to the weights themselves can be obtained.

Many deep networks learn a fixed size input to a fixed size output. To calculate intermediate layers, a weighted sum of the inputs from the previous layer is retrieved and passed through a non-linearity. The most popular non-linearity at the moment is the Rectified Linear Unit, or ReLU $f(z) = \max(z, 0)$, but previously researchers have used smooth non-linearities, such the tanh $f(z) = \tanh(z)$ or the sigmoid $f(z) = 1/(1 + exp(-z))$. The advantage of the ReLU is that it allows the gradient calculation to be done faster, thus it results in faster training in deep supervised networks, without any necessary unsupervised pre-training (Glorot et al., 2011). Units that are in the middle layers of the network are called hidden units and their purpose is to distort the input in such a way that it becomes linear with respect the output that is being predicted, as shown in Figure 3.8. This may be achieved through a series of non-linear transformations. The challenge of this problem is that many learning procedures may get stuck in poor local minima, where changing the weights a little bit only results in an increase in the error. In addition, extracting good features was thought to be only possible through significant prior knowledge. In practice, local minima are not a serious issue as it can be shown that many of the local minima are in fact saddle points and they are very similar in quality to each other. Even though the gradient is near zero at these points, the network can still have good generalization properties. Therefore, it does not matter which saddle point the optimization achieves (Dauphin et al., 2014).

Unsupervised pre-training was originally useful to train deep networks without adequate training data. This enabled layer-wise pre-training of the network, by attempting to reconstruct the input. This way, more complex feature detectors could be trained and the weights initialised to sensible values. Then a final output layer could be added to the system and all the parameters fine tuned using standard backpropagation (Bengio et al., 2007). Such a system was initially used for recognizing handwritten digits or detecting pedestrians. Another application of unsupervised pre-training was in speech recognition. Such networks were feasible to train with the advent of GPUs, which sped up training considerably (Hinton et al., 2006). In one experiment, short time frames in the speech were mapped to human utterances and the probability was maximised. This achieved very good results and by 2012 many Android phones ran one flavour of this algorithm to recognize speech. For small datasets, unsupervised pre-training helps avoid overfitting, resulting in better generalization. It can also facilitate transfer learning from a large source domain to a smaller target domain. So far, unsupervised pre-training has mostly been useful for small datasets. Another popular type of feed forward neural network is the convolutional neural network, which has been popular in the computer vision community. However, this kind of architecture can also be applied to sequential data such as time series and text, where they were previously known as autoregressive neural networks.

3.3.3 Convolutional Neural Networks

ConvNets can process data in multiple channels, for example the three two dimensional colour channels of images containing the pixel intensities (Le Cun et al., 1990). 1D signals include sequences, such as for languages, 2D signals include images or audio spectrograms, and 3D signals are for video or volumetric images. In convolutional networks, we emphasize local connections with shared weights, pooling, and multiple hidden layers. A typical ConvNet architecture can be seen in Figure 3.9.

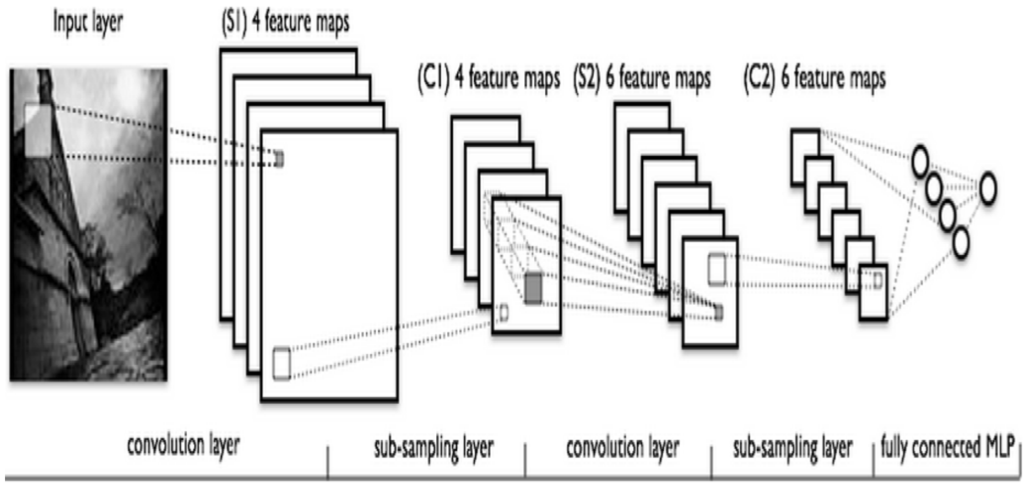


Figure 3.9: A convolutional network for image processing

A convolutional network architecture has several stages. First, there are two types of layers: convolutions and pooling. In convolutional layers, local patches of the input are mapped to feature maps. The mapping weights are called filter banks and the output can be further pushed through a non-linearity such as ReLU. This mapping is then performed in a series on subsequent patches of the data using the same filter banks. Part of the justification for this architecture is that in images, local motifs are often correlated in the neighbourhood of the input, which can be detected this way. Second, the motifs can appear anywhere in the image, therefore the neural network should be invariant to the location of the motif. This is achieved with weight sharing. Mathematically speaking, the filtering operation is a discrete convolution, hence where the name comes from. While convolution detects local patterns, the role of pooling is to merge semantically similar features. Because the location of each feature can vary somewhat, pooling merges coarse-grained locations of the motifs. In a typical pooling scenario, we compute the maximum value of a local patch of feature maps. This way, we are merging the local motifs in the dataset, making them location invariant to slight shifts or distortions. We typically perform one or three layers of convolutions coupled with pooling and non-linearities, followed by fully connected layers. Backpropagation in a convolutional net is possible and all the network weights can be trained (LeCun et al., 1998).

Deep learning makes use of the property that many natural phenomena are compositional hierarchies, in which lower level features can compose higher level ones. For example, in images local edges can form motifs, which then can assemble parts and objects. Similarly in speech and text, sounds can form phones and phonemes, or syllables can make up words and sentences. Pooling makes the representation invariant when lower level representations can change in appearance or position. Convolutional and pooling layers may be inspired by simple and complex cells in the vision system and neuroscience (Hubel and Wiesel, 1962). The architecture is reminiscent of the LGN-V1-V2-V4-IT pathway in the visual cortex (Felleman and Van Essen, 1991). In addition, half of the variance in the neural activation of monkeys can be explained by the high level units of a conv net (Cadieu et al., 2014). ConvNets originate from the neocognitron (Fukushima and Miyake, 1982), which had a similar architecture but was not trained with backpropagation. A 1D time-delay neural network can be used to recognize phonemes and words (Waibel et al., 1989). Time-delay Convnets were also used for speech recognition and document reading, which could be combined with a probabilistic language models. This system proved to be popular for cheque reading in the USA in the early 1990s. Microsoft also deployed multiple handwriting recognition software (Simard et al., 2003). Last, conv nets were applied to object detection, such as faces and hands (Lawrence et al., 1997), and image understanding covered next.

3.3.4 Multimodality

Conv Nets can be applied to various modalities of data, such as text or images. Since 2000, Conv Nets have been applied with success to object and region detection and segmentation. These were tasks with lots of label information. Example application include traffic sign recognition, biological image segmentation, detection of faces and body parts in natural images (Cireşan et al., 2012; Ning et al., 2005). Sometimes, images were labelled at the pixel level, which has application in autonomous robots and self-driving cars (Hadsell et al., 2009). Industry partners such as NVidia were also interested in this technology. Conv Nets were also useful for natural language understanding and speech recognition, where they are seen as alternatives or complementary to recurrent neural networks. The main success of Conv Nets came with the 2012 ImageNet challenge, where they achieved very good results in recognizing objects from a 1,000 different categories. Some of the key outcomes of the competition was the use of GPUs, the use of dropout regularization (Srivastava et al., 2014), and data augmentation by generating additional noisy examples. Conv Nets have since become the state-of-the-art in many computer vision tasks, and approach human performance. In recent years, they have been also be combined with recurrent nets for caption generation (Vinyals et al., 2015).

Conv Nets can have tens of layers of ReLUs, millions of weights and billions of connections. Previously, the training of these networks may have taken years, now it is feasible to do within hours thanks to better hardware parallelization. Conv Nets are used in many tech companies, including Google and Facebook, as well as startups. Conv Nets, along with feed forward networks, can be implemented on field-programmable gate arrays, which can enable real time image processing (Boser et al., 1991). In the next section, we will talk about distributed representations which are especially useful as a form of pre-training for natural language data.

3.3.5 Distributed Representations

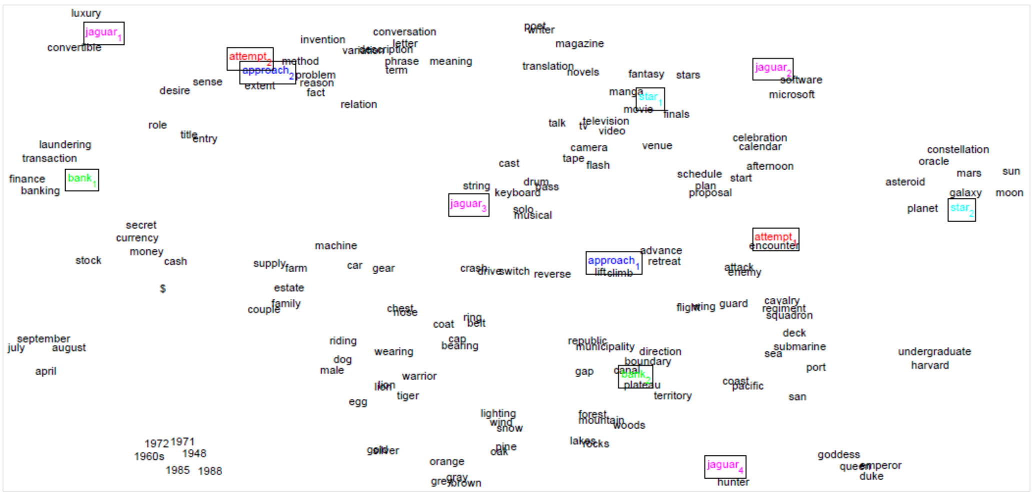


Figure 3.10: Word embedding visualizations with t-SNE (Van der Maaten and Hinton, 2008).

Deep learning has a few advantages over conventional machine learning techniques. The use of distributed representations allows for the exploitation of the compositional structure of the data. For instance, one can generalize to new composition of the data beyond what has been seen during training, as 2^n combinations are possible with n binary features (Bengio, 2009). Second, in depth the learning can become exponentially more efficient.

The hidden layers in the neural net learn an input representation that is predictive of the output target. For example, predicting the next word in a sequence of text from local context (Bengio et al., 2006). Words are typically represented with one-hot encodings, and in the first layer each word generates a pattern of activation. Then subsequent layers learn to predict the probability of the next word that is available in the vocabulary, and maximise the correct word probability. Figure 3.10 shows a sample of word representations projected onto a 2D space.

The learned word vector dimensions can be interpreted as different aspects of the words. This was first demonstrated for distributed symbols. These semantic relationships were not hand engineered into the inputs, but rather were discovered through the learning procedure which factorised the input and output symbols into multiple active rules. Word vectors also work well when the representations are learned from a large unstructured real word corpus. This way, the network can learn based on context that the words Tuesday and Wednesday are similar, as well the words Sweden and Norway. The features of these words are not exclusive but can be shared via variations in the observed dataset. In addition, no expert coded these rules into the vectors, but were discovered automatically by the network.

Word vector representations are now widely used in natural language processing (Bahdanau et al., 2014; Socher et al., 2011; Schwenk, 2007; Mikolov et al., 2013b). The issue of representation is critical part of the debate between logic and neural network inspired models of cognition. In logic, an instance of a symbol can be either identical or not to another symbol. It has no structure on its own. For reasoning, these instances must be bound by rules for inference, which must be carefully chosen. In neural networks however, reasoning is done over weight vectors and non-linearities, and common sense is more intuitive under this paradigm as there are no rigid rules. Before distributed representations, many NLP tasks were based on statistical counting of frequencies of words in local N-gram contexts. With V the number of vocabulary items, the number of possible N-grams is VN. To take a large context into account, this would require a large training corpus to scale well. N-grams treat words as atomic units and therefore they cannot generalize to semantically similar words. By associating each N-gram to a continuous real valued vectors, semantically similar words can end up close to each other in the vector space. In the next section we will cover recurrent neural network, which also has wide applicability in finance and text processing.

3.3.6 Recurrent Neural Networks

With the introduction of backpropagation, the feasibility of training recurrent neural networks have become more manageable. RNNs are more suited for tasks with sequential inputs such as speech and language. RNNs process examples one at a time and maintain a hidden state vector with all the history of the past elements it has seen. Provided that we consider the state vectors as outputs of the RNN, then we can apply standard backpropagation to train

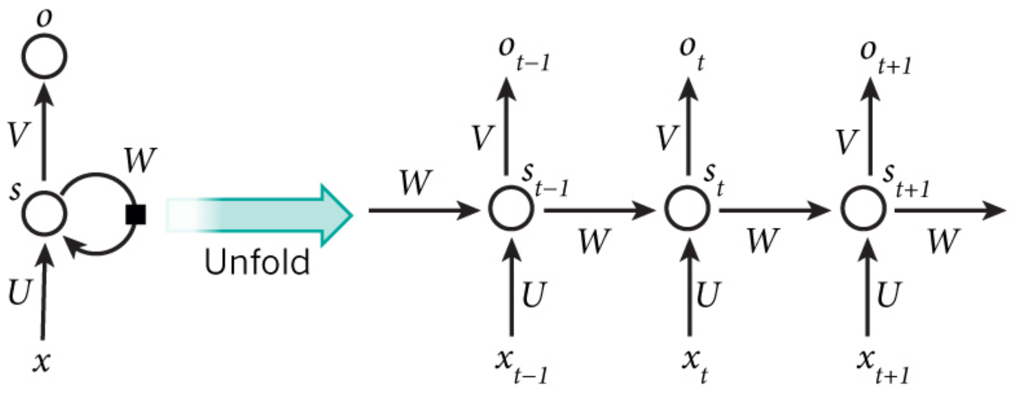


Figure 3.11: Unfolding a recurrent network

the parameters of the networks. An unfolded RNN can be seen in Figure 3.11, where x is the input at time step t, U are the encoding, V the decoding, and W the recurrent weights, s is the hidden state and o is the output. Even though RNNs are very powerful, training them effectively has remained elusive, because the backpropagated gradients either tend to explode or vanish over multiple time steps (Bengio et al., 1994; Pascanu et al., 2012). However, with advancements in their architectures and better training procedures, such as the introduction of Long Short Term Memory unit (Hochreiter and Schmidhuber, 1997), they have been good at predicting the next character or word in a text, but they are also used for more complicated tasks (Sutskever et al., 2011). For instance, an English sentence can be encoded by a RNN in order to produce a thought vector for that sentence one word at a time. Then this vector can be fed into another RNN for decoding, which can generate words one at a time using a probability distribution, until a special full stop symbol is encountered. The new French tokens are then conditioned on the tokens previously generated so far, as well as the English sentence, using a special attention mechanism (Bahdanau et al., 2014) that allows the network to focus on different parts of the input sentence while generating the translated output sentence one token at a time. This procedure can generate a French sequence according to a probability distribution by only conditioning on the English sentence. This is a rather naive way of doing machine translation, but it has quickly become the state of the art. This also questions the premise whether intricate symbolic expressions need to be manipulated to do translation, or whether neural common sense reasoning with multiple analogies is sufficient (McClelland and Rogers, 2003).

Instead of doing translation between two languages, the RNN can also perform translation between an image and a caption. The encoder network can be a ConvNet that encodes the pixel information into a hidden vector. Then the decoder network takes this presentation and generates an English sentence as seen before in machine translation and language modelling. Once unfolded in time, such a RNN can be seen as a deep feed forward neural network that shares the same weights in each time step. Even though they can store long range dependencies, in practice this may be challenging due to the exploding or vanishing gradient problem. This is related to unstable eigenvalues of the recurrent weights that are multiplied together in each step (Bengio et al., 1994). One solution for this problem is to introduce an explicit memory cell in the network. One example is the long short-term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997), which has a special hidden unit to remember the input over long periods of time. The memory cell accumulates information over time and it also has a connection to itself. Access to the cell is modulated by leaky gates that manage the input, the output from the cell, and how much information should be erased from memory. LSTMs have proved to be more effective than conventional RNNs, especially when they have deep architectures coupled with them. This could allow the raw processing of audio signal to the generation of characters in transcription. Machine translation also makes extensive use of gated networks. A recently proposed alternative to LSTMs is Gated Recurrent Units which attempts to simplify the carrying over of long distance information in the memory cell (Chung et al., 2014). Last, some good results have been achieved by carefully initializing the recurrent weights in the network so that they retain their eigen values of 1 and propagate the gradients effectively (Le et al., 2015). The LSTM is defined as follows. The proposed memory cell $\tilde{\mathbf{c}}_\mathbf{t}$ is modulated by the input $\mathbf{i_t}$, the forget $\mathbf{f_t}$, and the output $\mathbf{o_t}$ gates. The hidden unit outputs the next memory cell $\mathbf{c_t}$ and hidden state $\mathbf{h_t}$.

$$\tilde{\mathbf{c}}_\mathbf{t} = \tanh(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_c}\rangle + \mathbf{b_c}) \tag{3.56}$$

$$\mathbf{i_t} = \sigma(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_i}\rangle + \mathbf{b_i}) + \mathbf{c_{t-1}}\tilde{\mathbf{w}}_\mathbf{i} \tag{3.57}$$

$$\mathbf{f_t} = \sigma(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_f}\rangle + \mathbf{b_f}) + \mathbf{c_{t-1}}\tilde{\mathbf{w}}_\mathbf{f} \tag{3.58}$$

$$\mathbf{o_t} = \sigma(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_o}\rangle + \mathbf{b_o}) + \mathbf{c_{t-1}}\tilde{\mathbf{w}}_\mathbf{o} \tag{3.59}$$

$$\mathbf{c_t} = \tilde{\mathbf{c}}_\mathbf{t}\mathbf{i_t} + \mathbf{c_{t-1}}\mathbf{f_t} \tag{3.60}$$

$$\mathbf{h_t} = \tanh(\mathbf{c_t})\mathbf{o_t} \tag{3.61}$$

We can also add peekback weights as defined in (Graves, 2013). The alternative simplified

GRU is defined as

$$\mathbf{u_t} = \sigma(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_u}\rangle + \mathbf{b_u}) \tag{3.62}$$

$$\mathbf{r_t} = \sigma(\langle(\mathbf{x_t}, \mathbf{h_{t-1}}), \mathbf{W_r}\rangle + \mathbf{b_r}) \tag{3.63}$$

$$\tilde{\mathbf{h}}_\mathbf{t} = \tanh(\langle(\mathbf{x_t}, \mathbf{r_f}\mathbf{h_t}), \mathbf{W_g}\rangle + \mathbf{b_g}) \tag{3.64}$$

$$\mathbf{h_t} = (\mathbf{1} - \mathbf{u_t})\mathbf{h_{t-1}} + \mathbf{u_t}\tilde{\mathbf{h}}_\mathbf{t} \tag{3.65}$$

where $\mathbf{x_t}$, $\mathbf{h_t}$, $\tilde{\mathbf{c}}_\mathbf{t}$ and $\mathbf{c_t}$ are the inputs, new hidden states and proposed and actual memory cells; $\mathbf{i_t}$, $\mathbf{f_t}$, and $\mathbf{o_t}$ are the input, forget and output gates; $\tilde{\mathbf{w}}$ are peekback weights; and $(.,.)$ denotes concatenation.

Other ways of augmenting RNNs with a memory module include the Neural Turing Machine which includes a tape-like memory that can be written to or read from, and memory networks with a kind of associative memory (Graves et al., 2014; Weston et al., 2014). One application of such networks is question answering where the system is read a story one line at a time and needs to produce answers to some question. Other tasks beside memorization include reasoning and symbol manipulation. In addition, these networks can be taught algorithms, for example related to the sorting of an unsorted list of symbols. They can also keep track of the state of the world after reading a story and answer questions requiring complex inference. One example is after feeding in the story of the Lord of the Rings in 15 sentences, it correctly answered "where is Frodo now?"

3.3.7 Future Directions

Even though unsupervised learning was important in reviving interest in deep learning, it has been overshadowed by purely supervised methods. One example where unsupervised learning can shine is by observing that many things in the world do not have a label attached to them. A future direction for deep learning can come from the way human vision is processed, with a large low resolution surrounding and a high-resolution fovea, focusing on specific tasks intelligently. Such systems can be trained end-to-end combining RNNs with ConvNets, and use reinforcement learning to pick where to focus their attention. These systems have already outperformed passive vision systems and learned to play different video games (Mnih et al., 2015). Another big improvement can come from natural language processing, where systems can learn to focus their attention selectively to different parts of a document or sentence. Finally, combining deep learning with reasoning is another avenue to explore. This would make the manipulation of rule-based symbolic systems succeeded by large vector representation and operations (Bottou, 2014).

3.3.8 Benefits of Depth

There is some evidence that the brain is a modular framework with separate parts responsible for specific functions, such as vision or hearing. However, for these modules to communicate with each other, there needs to be a shared representation of information for effective interaction. In the Stripey cat experiment (Hubel and Wiesel, 1959), it was shown that cats brought up in vertical world were blind to horizontal lines. This suggest that specific parts of the cat's brain are responsible to detecting horizontal edges. Similarly, in the Split brain experiments (Sperry, 1968), people with disconnected brains were able to recognize some objects but could not name them. This suggests that certain areas of the brain are responsible for object detection, while others have linguistic functions. In deep representation learning, we aim to emulate this aspect of the brain by composing hierarchical and modular representations of the data that can interact with other components of the system in order to produce a prediction.

Deep learning architectures have resulted in massive improvements in visual recognition tasks (Krizhevsky et al., 2012). In natural language processing however relatively shallow linear representations have dominated recent advances, such as the shallow but efficient word embedding model trained by (Mikolov et al., 2013b). In addition, most works have focused on sentence level tasks, rather than longer documents. (Kim, 2014) has demonstrated in a series of experiments that a single layer convolutional net can achieve state-of-the-art performance on a multitude sentence level prediction tasks. On the other end of the spectrum, the neural network by (Krizhevsky et al., 2012) has several layers of depth. In addition, (Socher et al., 2013) proposed an approach where sentences are parsed using multiple levels of recursive neural networks. Therefore, it has been unclear whether deep architectures are always warranted for text based systems. For example, it is not evident how to aggregate sentences into a document representation and the computational processing of long documents with many sentences may be also prohibitively expensive.

One issue with neural networks is that text often varies in length whereas a neural network requires a fix length input. (Collobert and Weston, 2008) proposed to solve this by using a convolutional neural network, where the same set of weights are applied in a sequential window over the input data, in order to learn a feature map, and then a max pool operator ensures that important features propagate to the final feature representation of the input, while unimportant ones are forgotten. Even though the max pool operator neglects the order of the input, the order is still locally taken into account with the convolution operator to which the input can be uni, bi or higher order n-grams.

Another issue with neural networks is that they are traditionally considered black box models, and it has remained a challenge how to interpret the processes and features represented by various hidden layers of the network. In image processing, this problem has been more successfully addressed, by simply plotting the activations of neurons. Due to the pictorial nature of images, the outline of motifs and higher level objects can be seen in these plots, which gives an intuitive understanding of what the neuron detects. For natural language processing however, text is not pictorial in nature, therefore it has been very challenging to capture the detection mechanism of neurons in this application area. A neuron may encode complex semantic meaning of text, which is not evident how to plot or intuitively visualize. Although dimensionality reduction techniques, such as T-SNE (Van der Maaten and Hinton, 2008) have been relatively successful at plotting cluster of word embeddings, in this thesis we extend this approach in two ways. First we apply T-SNE to plot higher level layers of the neural network, particularly the last hidden layer, which gives us an understanding what the network aims to predict and how it understands the text input it received. Second, we elaborate on a novel algorithm to selectively blank out phrases in the text input and observe the prediction changes. This gives an intuitive understanding of how various phrases impact the output, and how these phrases are interpreted by complex non-linear interactions in the depth of the layers. This way, we can see that many phrases have both positive and negative impact on the output depending the context they appear in. This algorithm significantly aids in dispelling the blackbox nature of neural networks for text processing, and it is detailed in section 6.1.

3.3.9 Text Representation

A bag of words vector space representation of text data often involves a high dimensional vocabulary and learning the weights for each word per dimension. Many powerful learning algorithms can easily go awry due to the curse of dimensionality. Hence, the most successful models for bag of words data have often been linear models with sparsity inducing characteristics, i.e. via L1 or elastic net regularization, in order to avoid overfitting. Most efforts have been put into feature engineering in order to allow the algorithm to learn a sufficiently high dimensional representation for predictions. However, feature engineering is a time consuming task and requires human intervention. Moreover, effective feature engineering is often coupled with data preprocessing, which may include stemming, removing stop words, rare words, frequent words, replacing numbers with a designated token, downcasing, tokenization, and finally creating a vector space model of text. Then engineered features may include n-grams, POS tags, dependency relations, meta data, and if we want to include domain information, then all features need to replicated per domain, which results in feature explosion, and thus the need for sparse regularization techniques. Furthermore, all feature engineering and data pre preprocessing requires extensive domain knowledge, and thus for each domain this task needs to be repeated with manual examination of the domain by an expert. A general learning algorithm that works out-of-the-box would reduce the cost of development by a significant degree but has remained elusive.

Therefore, it has been a priority to minimize the time spent on the engineering tasks mentioned above as well to find ways of effectively avoiding the curse of dimensionality. In this work we propose a convolutional neural network to automatically learn good features for textual and meta data via hierarchical representations. This way, we can significantly cut down on the preprocessing and feature engineering work required, as well as we show that the model is unlikely to overfit even on small number of training instances. In addition, we can make use of pretraining on a much larger text corpus in order to imbue the learnt distributed representations of words with semantic meaning. This allows for exploiting similarity of words in the learnt feature space and thus results in more efficient learning.

Another issue with linear vector space models of text is how to incorporate non-linear relationship in the data. Different words within different contexts will have different meanings, which is something a linear model cannot capture well. Previous work focussed on additional feature engineering to encode more intricate relationships in the data. However, not only is feature engineering labour intensive and requires extensive domain knowledge, but also it is not clear what the ideal feature for the learning problem is that produces optimal predictions. A representation learning framework can learn the best feature representation of the data for optimal prediction without explicitly and manually designing these features. By doing several hierarchical non-linear projections of the data, the composition of higher level layers of lower level layers yields exponential benefits in learning via the sharing of lower level structures. Last, this framework is very flexible due to the modular and compositional nature of the network, and therefore it can incorporate various types of multi-modal data, such as unstructured or sequential data, i.e. text, and structured data, i.e. meta data, or even time. For advanced temporal modelling of text and structured data with representation learning, please refer to section 6.2 for more experiments.

3.3.10 Revenue Prediction

Gross movie revenue prediction has been extensively studied in economics and marketing. Previous work made use of meta data (Simonoff and Sparrow, 2000), i.e. running time, budget, genre; polarity scores of reviews (Terry et al., 2005), i.e. 5 star ratings; and aggregate statistics such as mentions of movie title (Zhang and Skiena, 2009). (Joshi et al., 2010) aimed to predict the future gross revenue of movies between 2005-2009 using preliminary reviews from several online newspapers as well as meta data. The authors note that their work is related to sentiment analysis, but instead it focuses on directly determining the value of each movie in the future. Therefore, they framed a problem directly as revenue prediction rather than as a surrogate sentiment analysis. With multi task learning, they aimed to identify media outlet and author specific characteristics. They aimed to identify key phrases and rich language constructs that affect revenue, and how these vary across review sites and individual critics. They performed this task via linear regression where they used uni-, bi-, and trigram term frequency counts extracted from reviews and concatenated that with meta data to predict the movie revenue. To identify site specific characteristics they replicated the individual text features for each domain, as well as kept a shared feature set, which is a form of so called embarrassingly parallel multi task learning. They also added dependency relations and part of speech tags to the feature set. Their results indicated that the review texts were indicative of the opening weekend box office revenue and could partly replace structured data features. Their combined model achieved the best results. In most cases, language syntactic information did not improve the model significantly, but the domain features resulted in moderate error reduction.

Neural networks allow for the modelling of movie revenue directly, rather than casting the issue as a surrogate problem of sentiment analysis often encountered in previous work seen in section 3.1.4. Note that it is undesirable to treat the problem in the model pipelining framework, where we first attempt to solve the issue of sentiment analysis, and use that result to forecast revenue. Not only does this approach rely on human heuristics, i.e. that sentiment correlates with revenue which may not be true, but also any errors made in the first part of the sentiment analysis system will propagate to the revenue prediction part. With a neural network approach, we can do end-to-end learning where we feed raw text and meta data into the model, and predict the metric of interest directly as the output of the model. We can also optimize for a custom objective directly related to our problem. Therefore, the neural network will learn the optimal representation of the data without relying on any human heuristics. As we will see, it turns out that for movie revenue prediction, most sentiment words are neutral, which shows that revenue prediction is a very different problem from sentiment analysis. In fact, the phrases with the highest predictive power often correspond to structured data information, which state facts related to the movie, i.e. that the movie is age restricted and thus not available to a segment of the population. This highlights the benefits of end-to-end learning by reducing partially incorrect design assumptions in the modelling process.

Finally, people also attempted to do multi-task learning or domain adaptation in order to account for domain specific characteristics of the data. With linear models, this approach can significantly increase the predictive power for the model, as seen in (Joshi et al., 2010), due to the fact that linear models are quite inflexible. Therefore, domain adaptation, or embarrassingly parallel multi-task learning, where a feature is replicated for each domain provides an effective way of increasing the complexity of the model in a controlled manner. However, in practice this results in feature explosion, in addition to the manual feature engineering steps detailed in the previous section, which further exacerbates the problem of curse of dimensionality and overfitting. With a non-linear model that can implicitly model complex interactions in the dataset, multi-task learning may yield less benefits. That is because even with a simple categorical variable as an additional input, the network can learn to compose complex functions incorporating the domain category into the model. In addition, if there are some text tokens that naturally correlate with the domains, then this can be exploited for indirectly learning about the domain from the text input. This again highlights the benefits of using non-linear neural networks that can implicitly uncover domain information from the data, without injecting explicit design assumptions about the learning problem into the modelling framework. It is vital to strive for the simplest model possible in view of Occam's razor. As we will show in sections 6.1.3 and 6.1.3, it turns out our proposed approach can in fact indirectly uncover domain information from the reviews. In addition, the proposed neural network is relatively simple in contrast with complex variants and extensions encountered elsewhere in the literature.

3.3.11 Comparison to Generative Models

In this final section, we give a brief comparison to GPs that complements the comparison we did in the end of section 3.2.

Neural networks are instances of discriminative models, which are in contrast to the generative models of Gaussian Processes. As we saw in section 3.2, although the Bayesian way of modelling the joint distribution of the problem $p(y, X)$ can be attractive to avoid overconfident predictions for incomprehensible data (see adversarial examples in (Nguyen et al., 2015)), making the simplification to only model $p(y|X)$ has its own advantages. For example, it avoids the sometimes intractable calculation of $p(X)$, which makes the training of discriminative models very efficient. In particular, for deep learning, this results in exponential efficiency over depth, which means these types of models can be trained on a lot more data, than their Bayesian counterparts. Combined with stochastic learning methods, the scalability of neural networks to large datasets makes them attractive for avoiding overfitting, and achieving good generalization. In addition, GPs assume a Gaussian distribution for $p(X)$ where analytical solution exists, but this assumption may not hold in real life. NNs make no such assumptions.

NNs also only attempt the estimate the single most likely parameters θ that describe the labels well, while avoiding the modelling of the entire parameter distribution. Last as an additional advantage, the architecture of neural networks enables the exploitation of custom made hardware, such as GPUs, that allows for fast matrix multiplications, while for Bayesian models scalability has remained a problem even though advances have been made recently

(Hensman et al., 2013a). As we saw with GPs, being a nearest neighbour type model requires a lot of memory where training is slow in order to tune the kernel and other hyperparameters, while predictions are fast as it only requires a look-up. However, NNs do not explicitly store data which allows for a more compact representation of the learning problem. On the other hand this also results in more computationally intensive predictions. Therefore, to speed up NN training, parallelised and/or distributed training procedures have been explored (Dean et al., 2012). It is possible to approximate a GP with a specific kernel with a large number of specific type of hidden units in a NNs (Neal, 2012). In addition, there have been attempts at exploiting the efficiency of consecutive deep projections of the data in deep neural networks with deep Gaussian Processes (Damianou and Lawrence, 2012).

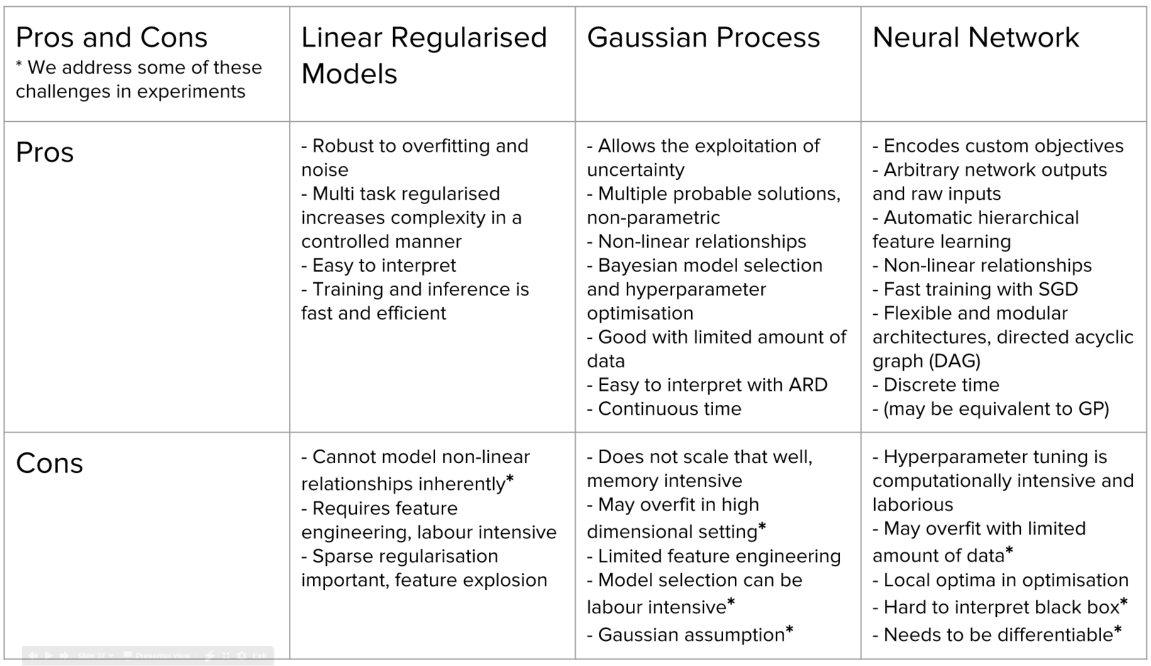
3.4 Pros and Cons of Modelling Frameworks

In this last section, we compare Linear Regularised Models, Gaussian Processes and Artificial Neural Networks. In Figure 3.12, we can see that Linear Models are robust to overfitting and to noise in case the number of features is limited. With multi task learning regularisation, the complexity of the model can be increased in a controlled manner. In addition, due to the linear relationship between the input and output, the importance of each input feature of the model is easy to interpret. Training and inference in a linear model are also fast and efficient. However, linear models have drawbacks too. They cannot inherently model non-linear relationships in the data, and thus they require labour intensive feature engineering. This in practice can result in feature explosion where sparse regularisation becomes very important. We can circumvent the limitation of linearity by decomposing a global non-linear signal into multiple piecewise linear signals with multi task learning (section 4.1). In addition, by applying human constructed basis functions to the input, we can encode non-linear relationships (section 4.1).

Gaussian Process is a more powerful framework than linear models, in that it can inherently model non-linear relationships with a kernel. Being an instance of non-parametric models, it optimises and integrates over multiple probable hypotheses which makes it more robust to overfitting. In addition, it provides an uncertainty estimate around the predictions, which can be further exploited. The GP framework also makes use of Bayesian model and hyperparameter selection by optimising for the marginal likelihood of the data. When the number of examples are limited, it can be a very good choice. Although it is non-linear, the interpretability of the model is not lost in case the Automatic Relevance Determination (Neal, 1995) kernel is used, which retains the relative importance of each feature in the data. GPs can also seamlessly incorporate continuous time with a smooth kernel. On the other hand, GPs do not scale that well and remain memory intensive, though there have been recent advancements in introducing lower bound approximation into the likelihood function and make the inference more efficient (Hensman et al., 2013a). In addition, it may overfit in a high dimensional setting, though in our experiments we provide evidence against this claim, where we train on large dimensional text input (section 5.2). Although more flexible than linear models, GPs still require limited amount of feature engineering, before they project the data to a high (infinite) dimensional hyperplane. In addition, choosing between various kernels for each type of input is still labour intensive, though we provide evidence against this claim too in our experiments when we introduce an automated way of combining and weighing kernels for each type of feature group in the model (section 5.2). Last, GPs make a Gaussian assumption for the likelihood of the data, which may not reflect the environment in which the data was produced. In this case, approximation to a Gaussian likelihood can be used (section 5.1).

Finally, Neural Networks are the most flexible type of framework in this review, though under some settings, they can be equivalent to a Gaussian Process (Neal, 1995). They allow for a very modular architecture where the only requirement is that the network is a directed acyclic graph or DAG. This means that neural network can encode custom objectives and optimise its parameters for that. It can also produce arbitrary outputs and can work with a wide range of raw input types. It requires much less feature engineering than previous models as it can learn optimal representation of the data in a hierarchical and progressive manner. It can also model non-linear relationship very efficiently, i.e. it's a universal approximator for arbitrarily complex functions. Learning in a neural network is quite fast with stochastic gradient descent where fast convergence can be achieved towards to optimum after observing relatively few training examples. As opposed to GPs, a recurrent neural network models discrete time. The downside of ANNs is that the hyperparameter tuning is still very labour intensive and computationally expensive process, though work has been done on Bayesian optimisation in this regard (Snoek et al., 2012). In addition, due to their flexibility, an ANN may easily overfit a problem with limited amount of data. However, here we provide counter evidence against this claim later, when we train a large neural network only on around a thousand datapoints and achieve good generalisation performance (section 6.1). In addition, the optimisation procedure is highly non-convex so it may get stuck in local minima. However, evidence exists that these local optima may be in fact saddle points which are equally good quality (Dauphin et al., 2014). ANNs are also considered black box models that are very hard to interpret. In one of our experiments therefore, we propose a technique to easily interpret the impact of inputs on the output of the network, which gives a high level of insight into how various text phrases interact inside of the network in a non-linear fashion (section 6.1). Finally, a requirement for backpropagation to work is that the network must be differentiable and thus continuous. In this case, we propose a network with slightly extended objective function to show how we can work around this limitation of ANNs with good results (section 6.2).

This concludes our review of machine learning frameworks, and next we discuss some implementation details of our experiments and models.



Pros and Cons * We address some of these challenges in experiments	Linear Regularised Models	Gaussian Process	Neural Network
Pros	- Robust to overfitting and noise - Multi task regularised increases complexity in a controlled manner - Easy to interpret - Training and inference is fast and efficient	- Allows the exploitation of uncertainty - Multiple probable solutions, non-parametric - Non-linear relationships - Bayesian model selection and hyperparameter optimisation - Good with limited amount of data - Easy to interpret with ARD - Continuous time	- Encodes custom objectives - Arbitrary network outputs and raw inputs - Automatic hierarchical feature learning - Non-linear relationships - Fast training with SGD - Flexible and modular architectures, directed acyclic graph (DAG) - Discrete time - (may be equivalent to GP)
Cons	- Cannot model non-linear relationships inherently* - Requires feature engineering, labour intensive - Sparse regularisation important, feature explosion	- Does not scale that well, memory intensive - May overfit in high dimensional setting* - Limited feature engineering - Model selection can be labour intensive* - Gaussian assumption*	- Hyperparameter tuning is computationally intensive and laborious - May overfit with limited amount of data* - Local optima in optimisation - Hard to interpret black box* - Needs to be differentiable*

Figure 3.12: Highlighted pros and cons of modelling frameworks reviewed in this chapter.

3.5 Implementation

In the previous sections we have reviewed major machine learning frameworks. In this section, we give a brief outline regarding the implementation of the experiments in this thesis. Designing and implementing experiments in a suitable way has a large effect on the results, as much of the work is related to architecting the surrounding code that processes the data. Setting up the machine learning code and evaluating results may in fact only account for a small percentage of the overall work (Sculley et al., 2015). Therefore, having solid software engineering foundations and programming ability is a critical part of successful research.

Testing and debugging machine learning application is traditionally considered a very difficult problem because the algorithm learns around the bugs introduced by the developer, while giving slighltly less optimal results. To check the gradients of the implement algorithm, numerical gradient calculations can be done where the gradient is calculated by the finite differences method. This should match the analytically derived gradients. In addition, components of the algorithm can be isolated into their own functionality. For example, when assembling a deep network, separate test cases can be written for hidden layers, recurrent layers or for the cost function. This aids the programmer in isolating the source of suboptimal performance. Although print statements and debuggers are useful for inspecting the execution of a program, with symbolically constructed computation graphs, such as the one provided by Theano, this is not feasible to do. Therefore, Theano provides debugging capabilities with additional computation nodes such the Print node, which prints to the console all information and numerical values that flows through it. However, this can only be done during run time with little choice to halt the program as in conventional programming paradigms, where this sort of debugging can be more compared to the concept of reflection. In some cases, parts of the algorithm has been developed by other developers, in which case reasonable trust can be put into the correctness of that part of the system. However, with new research oriented software packages that are experimental in nature, unfortunately the occurrence of bugs is still very common, therefore the programmer must not ignore that possibility of algorithm malfunction. In this case, bugs can be reported to the original developer with a use case demonstrating the source of error in isolation. The disadvantage of this approach is the programmer has to familiarise himself with the inner software architecture of the library he is using, which may considerably slow down progression with the primary purpose of his experiments. Last, to further check the feasibility and correctness of the application designed, pair programming can be employed where two researchers or one engineer and one researcher sit together at a computer station and together develop the algorithm. In this case, one person would be responsible for the actual coding while the other one would closely follow and catch any subtle bugs early. They can also discuss design decisions and experiment setups.

Theano is a GPU accelerated and optimised python machine learning library (Bergstra et al., 2010). Automatic symbolic differentiation can be used to calculate the gradients of the loss functions. Theano does all its computation on the GPU and returns the results to the CPU. It is built on Numpy and Scipy, which are python numeric and scientific analysis and computation libraries (Oliphant, 2007). Tesla is a high performance general graphical processing unit (GPGPU) for cluster computers. It is not very efficient for small scale computation or for managing the memory between CPU and GPU well, and therefore bandwidth costs limit the number of cores that can be used. Due to the hardware circuits of GPUs, dot product linear algebra computations are very fast, but communication between the GPU and CPU can be slow through the hardware BUS at the same time. To circumvent this, all data is transferred to the GPU at the beginning of the computation, and transferred back once at the end. However, due to the fact that small tasks contain only a limited number of observations, the GPU architecture might not be utilized to its fullest extent. Theano is parallelisable with the use of symbolic operators. This generates a computation graph that is subsequently optimised and streamlined. This can be combined with the principles of the map reduce algorithm, which was invented at Google to process Big Data computations more effectively (Chu et al., 2006). The mapper applies a parallelisable function to all data items, while the reducer collects the results in one variable sequentially. However, this approach does not support random access well. These software development techniques can be used to speed up the experiments and make running of more complex models possible with contemporary hardware.

Agile development is a well known software development methodology that originates from managing research and development projects in manufacturing, and has recent started to make its way managing implementations of machine learning systems. Extensive unit test coverage checks the validity of the experiments and partial results. Agile development advocates writing tests first and constructing short easy to understand objects and methods. This helps keep the experiments and the object oriented program manageable and avoid subtle bugs (Cockburn, 2002). This can make sure there are no hidden bugs or mistakes in the output and experiment results. Such bugs may give a wrong impression of the accuracy of the experiments and in rare cases false confidence in the results, so it is imperative to minimise the probability of such events happening. Test driven development can help with this as it forces the programmer to implement their software in small continuous iterations of integration, focusing their mental power on a small chunk of the problem at one time, which minimises mental overload, and verifies each implementation step by a test case that tests one part of the software in isolation. External service methods can be stubbed out such as making a request to Google Finance service, and for testing purposes small synthetic datasets can be constructed to see if expected outputs match with actual outputs, without the need for computationally intensive training and evaluation on a large dataset. This way, the programmer or experimenter can be reasonably sure of the robustness of their application and experiment results, so that they can be reproduced at a later day with high confidence of accuracy.

In the following sections, we are going to detail the experiments we have run and present a discussion of the results.

Chapter 4

Linear Profit Maximisation with Knowledge Transfer

In the previous chapter we reviewed the theory of financial markets and made the argument for applying custom machine learning models in order to forecast the future evolution of these markets. We reviewed several machine learning techniques for this, including linear and non-linear ones. In this chapter, we focus on linear models. Linear models can be an attractive choice because they have limited capacity and therefore can force better generalization. As a result they are also more robust to fitting accidental patterns in the data. Hence, they can be more resilient to noise, and be able to fit weak signals present in financial data. A linear signal can also be transformed in the output space to model investment decisions, i.e. buying or selling. In addition, by manually constructing basis functions which are applied to the input, such as technical analysis indicators applied to market data, we can model non-linear relationships via the transformed input space with linear models. However, unfortunately linear models are unable to learn inherently non-linear relationships and therefore they cannot learn an optimal way to transform the input and derive better feature representations of the data. Nonetheless, the capacity of linear models can be increased with additional feature engineering, or by applying multi-task regularisation to different parts of the data set, which allows us to learn a piecewise linear function that may behave in a non-linear fashion once aggregated across all tasks. In the next chapters of this thesis, we will see other non-linear models, such as Gaussian Processes and Neural Networks that enable us to model non-linear relationships in an optimal way.

In this chapter, we investigate how we can extend linear regression by optimising for a custom objective that is trading profit. We show that by doing this, our forecast accuracy significantly improves in terms of the amount of profit the system produces. We then examine ways in which we can make linear models progressively more complex via multi task learning. We divide our dataset along companies and also in the temporal dimension along months. We show that by fitting a linear model to each of these tasks, but at the same allowing for data statistics to transfer between tasks, our predictive accuracy improves. This way, we are able to mine knowledge related to each company as well as the overall market, while accounting for temporal dynamics in the dataset. We show that our system is capable of predicting the market, and produces significant profit over the extensive test period, beating the competitive buy and hold baseline. We also demonstrate the advantages of technical analysis, where we identify key indicators with the highest predictive power in the technical analysis literature among around 150 industry standard indicators. In particular, indicators from the pattern matching family prove to be highly useful. Last, we show that improvements can be gained from modulating the learning process on a company and monthly temporal basis. Finally, a backtesting procedures measures the trading performance of the algorithm, and puts the trader in the 90% percentile of fund performances available in a historical study. Our data includes the largest companies trading on the London Stock Exchange, that are part of the FTSE 100 index for the periods of 2000-2013. This includes around 100 companies for the span of more than 12 years of data, and we have daily market data for each of these companies. The test set includes more than a decade of predictions, which we evaluate in a sliding window manner. In summary, this chapter gives supportive evidence that inefficiencies exits in developed markets such as the London stock Exchange, and these inefficiencies can be reliably exploited to obtain risk free profits over long term on a consistent basis.

4.1 Overview

Stock price movements are claimed to be chaotic and unpredictable, and mainstream theories of finance refute the possibility of realizing risk free profit through predictive modelling. Despite this, a large body of technical analysis work maintains that price movements can be predicted solely from historical market data, i.e., markets are not completely efficient. In this chapter we seek to test this claim empirically by developing a novel stochastic trading algorithm in the form of a linear model with a profit maximization objective. Using this method we show improvements over the competitive buy-and-hold baseline over a decade of stock market data for several companies. We further extend the approach to allow for non-stationarity in time, and using multi-task learning to modulate between individual companies and the overall market. Both approaches further improve the predictive profit. Overall this work shows that market movements do exhibit predictable patterns as captured through technical analysis.

A central tenet of financial theory is the Efficient Markets Hypothesis, which states that the market price reflects all available knowledge and accordingly no risk-free returns can be realized without access to non-public information. Despite its prevalence, this theory is not universally accepted, as it cannot explain several small but significant examples of inefficiencies commonly exhibited in markets. A large body of technical analysis techniques claim that recurring patterns can be identified from historical market data, which can be used to realize risk-free profits (Lo et al., 2000). Such techniques are widely used as part of active portfolio management, which aims to outperform passive 'buy-and-hold' management strategies. It is an open question as to the scientific validity of these claims, namely whether active trading and technical analysis can reliably realize above market returns. In this chapter we seek to test the question of whether markets are completely efficient by empirical validation using predictive modelling. We demonstrate that historical market movements and technical analysis contain predictive power for forecasting daily returns in an active trading scenario.

In this chapter, we propose a novel learning algorithm for profit maximization in an active trading scenario, where stocks are bought and sold on a daily basis. We show how trading can be modelled using a similar formulation to logistic regression, permitting a simple gradient based training algorithm and a straight-forward means of prediction on unseen data. This technique allows for recent market data, represented using technical analysis basis functions, to drive investment decisions. Compared to the manual use of technical analysis, where a trader interprets the identified patterns, here our training algorithm assists the trader in deciding which technical analysis indicators are important, and how these should be used in their trading decisions.

To reflect the idiosyncrasies of individual companies, we present a multi-task learning approach that is suitable for jointly modelling several companies on the stock market. This trains a series of per-company models, subject to a mean regularization term which encourages global parameter sharing. We show that this improves over independent modelling of each company, or joint modelling of all companies with tied parameters. A second question is how to handle changing market conditions over time, which is of particular importance in our setting as speculative opportunities are likely to change over time as they have been identified and removed by market participants. For this purpose we use a simple time based regularizer which permits model parameters to vary smoothly with time, which is shown to result in further improvements in predictive profit.

This chapter seeks to answer several research questions. The first is regarding market efficiency, namely whether there are systematic inefficiencies that can be exploited using statistical models. To answer this, we show that excess profits are achieved using our active trading models when compared to simple baseline methods, such as buy-and-hold. A second aspect of this research question is whether technical analysis can improve active trading models compared to using only recent price values, which we also show to be the case, although this difference is less dramatic. Together these results provide an empirical justification of active trading and technical analysis, refuting the efficiency arguments of financial theory.

The second set of research objectives concerns modelling. Our approach develops a model of financial trading, and a training method for optimizing trading profits. To test the validity of explicit profit maximization, we compare against squared error loss, the most common regression objective, and show significant outperformance. Our modelling includes multi-task learning over several different companies and over time, which we show leads to substantial benefits in profit over baseline models trained on individual companies, pooling together the dataset, or ignoring the effect of time. Overall these results suggest that fine grained modelling is useful, including modelling non-stationarities, but there is information from the global data. Multi-task learning is an effective means of balancing these two criteria.

The remainder of the chapter is structured as follows. In §4.2 we present our novel profit maximization model, and extensions to enable joint multi-task learning over several companies as well as temporal adaptation to handle non-stationarities. Next we turn to the evaluation, starting in §4.3 with a brief overview of technical analysis before presenting the validation methodology and comparative baselines in §4.4. Experimental results are presented in §4.5, evaluating the algorithm in several realistic trading scenarios.

4.2 Model

In this section, we describe a linear model for maximising profits in a daily trading scenario. Then we propose extensions of this model to allow for the idiosyncrasies of companies and trading time periods.

An active trading scenario is considered, where a trader makes an investment every day which is then reversed the following day. For small budgets and large markets, the trades are unlikely to impact the prices. Trades are based on the outputs of a linear model over technical analysis features encoding recent market history, as described in §4.3. Then the problem is how to trade based on the real valued prediction, considering the type of trade (buy or sell) and the magnitude of the trade. A linear model is chosen due to its simplicity, although the approach would accommodate a more complex non-linear modelling approach, such as a multi-layer neural network.

The invested money on each day is proportional to the confidence level of the prediction, which is denoted by the absolute sigmoid value. The sigmoid is a useful function because it is easily differentiable, thus resulting in an error term which can be easily minimized. The hyperbolic tangent sigmoid function is applied to the simple linear model of the input,

$$p(\mathbf{x}, \mathbf{w}) = \tanh(\mathbf{w}^\intercal \mathbf{x}) \,, \tag{4.1}$$

resulting in a prediction value $p \in [-1, 1]$ which determines the trade type (buy or sell, depending on $\text{sign}(p)$) and magnitude of investment ($|p|$). Figure 4.1 illustrates the trading actions resulting from several different inputs. Here a negative prediction denotes selling the stock whereas a positive prediction denotes buying the stock, with a maximum investment of £±1.

In order to train such a model, we use as the response variable the relative price movement from the stock. If the stock price falls by 50%, then the three scenarios illustrated in Figure 4.1 result in profits of roughly £-0.5, £0 and £0.5. More formally, the target of the prediction $y(d) \in [-1, \infty)$ is the return on investment,

$$y(d) = \frac{s_{\text{close}}(d+1) - s_{\text{close}}(d)}{s_{\text{close}}(d)} \,,$$

where d is the trading day and $s_{\text{close}}(d)$ is the closing price of the stock on day d. Consider

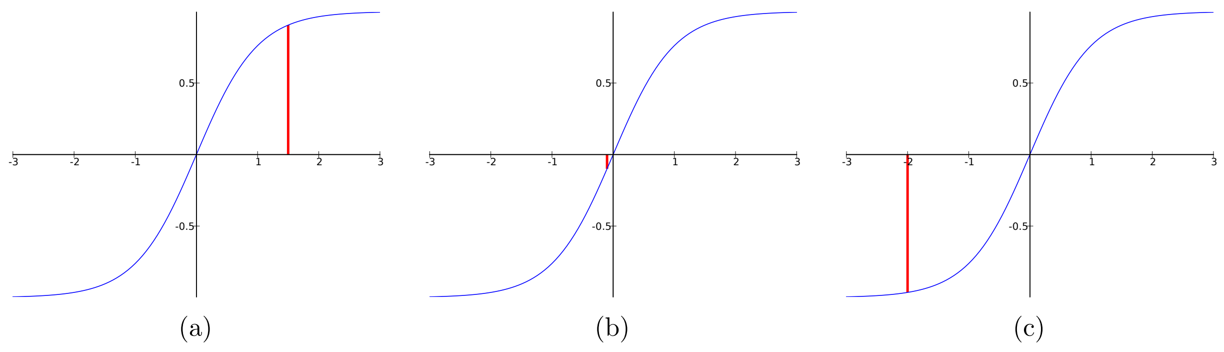


Figure 4.1: Prediction signal: Confident buy, Unconfident (short) sell, Confident (short) sell

now a linear regression baseline algorithm where the optimum weights are found that minimize the error of the predictions as measured by the squared difference between the targets and the predictions. This has the benefit of a closed form solution for the optimal weights (Rogers and Girolami, 2011), however it has two short-comings. First, it is not clear how to trade based on the signal, which can vary over a wide range of values, and secondly the training loss is highly dissimilar to trading profit. To account for the first problem, we can adjust the model predictions by first normalizing by the standard deviation of the prediction, after which we apply the hyperbolic sigmoid in Equation 4.1 to obtain a trading action. This is a rough heuristic to ensure the scale of predictions are comparable to that of other models.

The second problem of the mismatch between the training and the testing loss is more insidious, as the sum of squared errors does not resemble profit. To illustrate the difference between the two objectives, consider the case when the stock price rises. The squared error loss penalizes predictions which do not exactly match the price rise, irrespective of whether they are above and below the true price movement. This makes little sense, as all buy predictions should be rewarded, including extremely high values. A similar effect occurs for negative predictions, which attract heavy penalties out of keeping with the loss from trading.

A better approach is to integrate the sigmoid into the loss function such that optimization can be performed directly for profit instead of prediction accuracy. For this reason instead of minimizing the squared error of the prediction, as for the linear regression baseline, we seek to maximize profit directly. This gives rise to the profit (utility) objective,

$$u(\mathbf{x}, \mathbf{w}, y) = p(\mathbf{x}, \mathbf{w})y = \tanh(\mathbf{w}^\intercal \mathbf{x})y$$

Notation	Explanation
$\mathbf{X}$	training data of technical analysis indicators, $\mathbf{X} \in \mathbb{R}^{C \times M \times D_{c,m} \times T}$
$\mathbf{Y}$	targets, relative price movement of stocks, $\mathbf{Y} \in \mathbb{R}^{C \times M \times D_{c,m}}$
$\mathbf{W}$	model parameters, $\mathbf{W} \in \mathbb{R}^{C \times M \times T}$
λ_c	regularization coefficient between companies
λ_m	regularization coefficient between trading months
λ_l	regularization coefficient for weights
λ_j	proportional transaction cost coefficient
λ_k	fixed transaction cost coefficient
C	number of companies
M	number of trading months
$D_{c,m}$	number of trading days in company c and month m
T	number of technical indicators
R_c	company regularizer term
R_m	time regularizer term
R_l	L-2 regularizer term
U	utility or profit in entire dataset
$\mathbf{S}$	raw market data
J	transaction cost

Table 4.1: Notation for symbols used in this chapter.

which multiplies the prediction $p(\mathbf{x}, \mathbf{w})$ by the relative price movement of the stock, y. This corresponds to the profit realized over a single trade.

It is assumed that fractions of stocks can be traded and that stocks that are not currently possessed can be short sold. Further, trades can be executed at the closing price of the stocks of the companies each day for the invested amount. Last, transaction costs are not factored in the prediction signal, although this could be implemented using a form of hinge loss, where for each transaction a proportional penalty depending on the volume (magnitude) of the trade is subtracted, as shown in Equation 4.2. Likewise, for fixed costs a constant can be subtracted from each daily profit. Optimisation under this setting is still possible via sub-gradient methods.

$$J(\mathbf{W}) = \sum_{c=1}^{C} \sum_{d=1}^{D_c} \left(\lambda_j |\tanh(\mathbf{w}_c^{\intercal} \mathbf{x}_{c,d})| + \lambda_k \right) \tag{4.2}$$

The above assumptions are designed to be reasonable while still yielding a simple and differentiable objective for training the model. Table 4.2 gives an explanation of the notation used throughout the paper. The overall utility or profit U is obtained by aggregating the profit over all time periods, for which we compare two alternative methods:

$$U(\mathbf{W}) = \sum_{c=1}^{C} \sum_{d=1}^{D_c} \tanh(\mathbf{w}_c^\intercal \mathbf{x}_{c,d}) y_{c,d} \tag{4.3}$$

$$U'(\mathbf{W}) = \ln \prod_{c=1}^{C} \prod_{d=1}^{D_c} \big(\tanh(\mathbf{w}_c^\intercal \mathbf{x}_{c,d}) y_{c,d} + 1 \big) . \tag{4.4}$$

The first objective in (4.3) simply considers the total aggregate profit, which is appropriate if each trade is performed independently based on the same investment budget. The second objective (4.4) allows for compounding of profits whereby the proceeds from day d are reinvested and subsequent trades are based on this revised budget. In Equation 4.4 the constant 1 is added to the daily utility to reflect the multiplicative change in bank balance, and the logarithm is applied to simplify the formulation.

The training objective is to maximize the utility, as defined in (4.3) or (4.4), including an additive L_2 regularizer term to bias weights towards zero,

$$R_l = \lambda_l \sum_{t=1}^{T} \|\mathbf{W}_{\cdot,\cdot,t}\|_F^2 , \tag{4.5}$$

where $\|\cdot\|_F$ is the Frobenius norm of the weights and the coefficient λ_l modulates the effect of the regularization term relative to the profit. In the optimization, the loss function which measures the negative utility is minimized with respect to the weights $\mathbf{W}$ in order to learn the optimal model weights,

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} -U(\mathbf{W}) + R_l(\mathbf{W}) . \tag{4.6}$$

The optimization in (4.6) is solved using the L-BFGS algorithm (Byrd et al., 1995), a quasi-Newton method for convex optimization. Note that the objective in (4.6) is non-convex, and therefore gradient based optimization may not find the global optimum. However, our experiments showed that L-BFGS was effective: it consistently converged and was robust to different starting conditions. The L-BFGS optimizer requires first order partial derivatives of the objective function. The gradient of each component of the objective are as follows:

$$\frac{\partial}{\partial w_{c,t}} U = \sum_{d=1}^{D} x_{c,d,t} y_{c,d} \cosh^{-2}(\mathbf{w}_c^\intercal \mathbf{x}_{c,d})$$
$$\frac{\partial}{\partial w_{c,t}} U' = \frac{\sum_{d=1}^{D} y_{c,d} x_{c,d,t} \cosh^{-2}(\mathbf{w}_c^\intercal \mathbf{x}_{c,d}) \prod_{d'=1,d'\neq d}^{D} y_{c,d'} \tanh(\mathbf{w}_c^\intercal \mathbf{x}_{c,d'}) + 1}{\prod_{d=1}^{D} y_{c,d} \tanh(\mathbf{w}_c^\intercal \mathbf{x}_{c,d}) + 1}$$
$$\frac{\partial}{\partial w_{c,t}} R_l = 2\lambda_l w_{c,t} \, ,$$

where the first two equations are the partial derivatives of the two alternative loss functions – for non-compound (4.3) and compound profits (4.4), respectively – and the last equation is the gradient of the regularizer term.

4.2.1 Multi Task Learning

The algorithm makes a prediction for each company c each trading day d based on the data vector $\mathbf{x}_{c,d}$. Therefore, it produces multiple simultaneous outputs, one for each company. As new data points are acquired for each daily time step, a prediction is made for each company at the same time. We assume that different companies operate under different conditions, which in turn affect their trading on the stock market. Therefore each company is best modelled using different parameters. However as our dataset consists of companies which operate in the same market and are all 'blue-chip' stocks with high market capitalization, we would also expect that their price behaviors are linked.

Balancing these two effects is achieved by mean regularized multi-task learning (Evgeniou and Pontil, 2004). This method learns multiple related tasks jointly, which can improve accuracy for the primary task learned. Despite its simplicity among other multi-task learning methods, it can be highly effective. This is implemented by including a new penalty term, the 'company regularizer',

$$R_c = \lambda_c \sum_{c=1}^{C} \|\mathbf{W}_{c,\cdot,\cdot} - \frac{1}{C} \sum_{c'=1}^{C} \mathbf{W}_{c',\cdot,\cdot}\|_F^2 \, , \tag{4.7}$$

which penalizes differences between company specific weights and the mean weights across all companies. When the company regularizer coefficient, λ_c, is set to zero there is no regularization between companies, such that each company is modelled independently. Conversely, setting $\lambda_c = \infty$ means all parameters are tied, i.e., all company data is pooled together.

4.2.2 Non-Stationarity

Markets are dynamic systems since inefficiencies will be eventually discovered and exploited by traders, and thus exploitable signals in the market data may fade over time. Besides evolving speculative behaviors, it is likely that the underlying dynamics of financial systems and changes to external economic conditions (unknown to our model) result in a non-stationary process. Although the model may need to change with time, it is unlikely to change rapidly over a short period, but rather evolve smoothly with time.[1]

To encode the assumption of smooth variation with time, we elect to use an additional time regularization term. This is related to other temporal modelling methods such as the Fused Lasso (Tibshirani et al., 2005), which penalizes absolute changes in weights between adjacent time intervals using the L_1 norm. Here instead we use a L_2 term,

$$R_m = \lambda_m \sum_{m=2}^{M} \|\mathbf{W}_{\cdot,m,\cdot} - \mathbf{W}_{\cdot,m-1,\cdot}\|_F^2 \,, \tag{4.8}$$

where m is the trading month, used as the granularity of our temporal modelling. The regularizer penalizes weight differences for adjacent trading months, m and $m-1$, such that weights smoothly vary over time. The extreme behavior of the time regularizer represents either pooling ($\lambda_m = \infty$), allowing for no temporal variations, or independent modelling of each month ($\lambda_m = 0$). A visualisation of the company and temporal multi task regularisers can be seen in Figure 4.2.

The final optimization objective, including both the company and time regularization terms, is now

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} -U(\mathbf{W}) + R_l(\mathbf{W}) + R_c(\mathbf{W}) + R_m(\mathbf{W}) \,.$$

This objective discourages large differences between parameters for adjacent trading months (Equation 4.8), and also discourages large differences between the weights for individual companies versus the average over all companies (Equation 4.7). These biases are balanced against data fit, and consequently we expect the model to learn different parameter values for each task as needed to maximize training profit. The partial derivatives of the regularization terms

[1]Note that market crashes and other sudden events will not be handled well by this approach, a more sophisticated method would be needed to handle such cases.

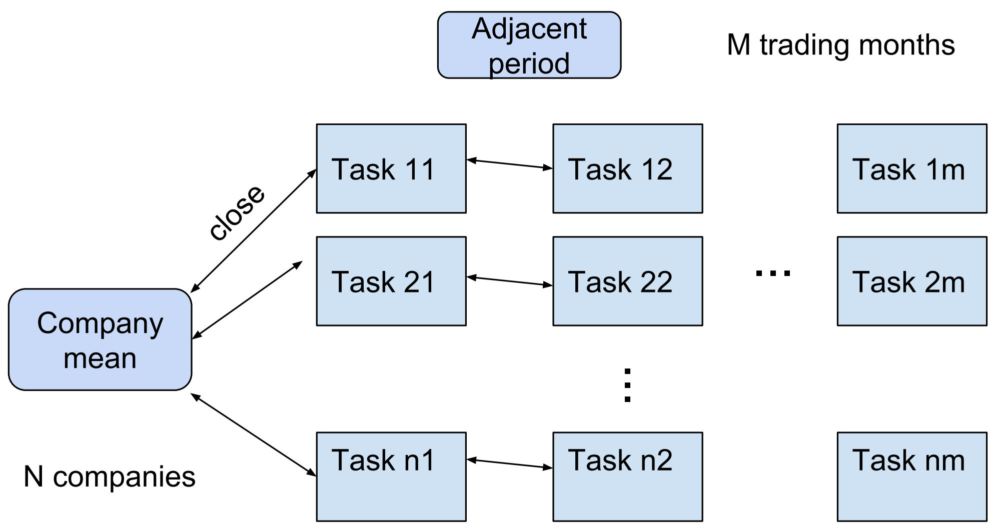


Figure 4.2: Multi task regularisation across the mean of company weights and adjacent monthly trading period weights.

are as follows:

$$\frac{\partial}{\partial w_{c,m,t}} R_c = 2\lambda_c \left(w_{c,m,t} - \frac{1}{C} \sum_{c'=1}^{C} w_{c',m,t} \right)$$

$$\frac{\partial}{\partial w_{c,m,t}} R_m = \begin{cases} 2\lambda_m (w_{c,m,t} - w_{c,m-1,t}) & \text{if } m = M \\ 2\lambda_m (w_{c,m,t} - w_{c,m+1,t}) & \text{if } m = 1 \\ 2\lambda_m (2w_{c,m,t} - w_{c,m-1,t} - w_{c,m+1,t}) & \text{otherwise} \end{cases} .$$

The multi-task regularizers facilitate the sharing of data statistics across task models, while also limiting overfitting the weak signal characteristic of noisy financial datasets.

4.3 Technical Analysis

Now we consider experimental validation of our trading model. Our dataset was sourced from Google Finance, taking market data over the period 2000–2013 for all companies constituting the FTSE index of the top 100 stocks by market value in the United Kingdom. The FTSE index composition changed during this period, due to companies being excluded or included, and we retain the 64 companies which remained in the index for the full period (see Appendix A.1 for the company ticker symbols). This confers a survivor bias to our results, as any companies suffering extremely poor performance or bankruptcy during the period have been excluded.

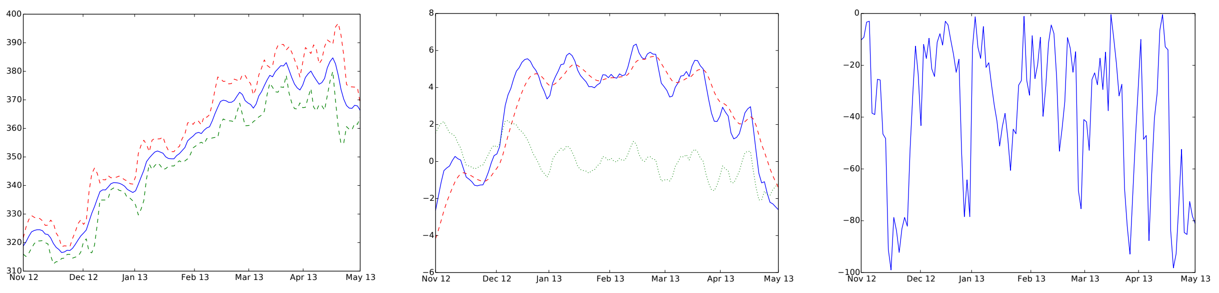
However note that this bias also affects our baselines, in particular inflating the performance of the passive 'buy-and-hold' strategy. The market data was cleaned to account for incorrect adjustment for dividends, stock splits or merges by manually verifying all daily price movements above 10%, and excluding any improperly adjusted prices.

The dataset is divided into individual 'tasks' by company and by trading month. The daily market data is transformed by applying a suite of technical analysis features, which are then each normalized and standardized before being used in the learning algorithm.

Technical analysis can reveal behavioral patterns in trading activity on stock markets. It has the potential to uncover behavioral cues of market participants and capture psychological biases such as loss aversion or risk avoidance. In its arsenal, sound statistical and quantitative analysis methods can be found in addition to heuristic pattern analysis such as candlestick pattern matching functions. The three pillars of technical analysis are history tends to repeat itself, prices move in trends, and market action discounts everything (Lo et al., 2000; Neely et al., 1997). This means that all past, current and even future information is discounted into the markets, such as emotions of investors to inflation or pending earnings announcements by companies. Therefore, technical analysis treats fundamental analysis, which analyzes external factors such as company performance reports and economic indicators, redundant for making predictions.

We apply technical analysis to the market data for each company using the TA-lib technical analysis library.[2] This transforms raw market data series, $\mathbf{S}$, of open, high, low, close and volume values with a family of technical analysis functions, ϕ, to obtain the technical indicator series representation of the data, $\mathbf{X} = \phi(\mathbf{S})$. Functions are applied from the family of overlap studies, momentum, volume, cycle, price transform, volatility and pattern recognition indicators. For the list of technical indicators, please see Appendix A.2. Statistical functions, mathematical transforms and operators are not used because these are helper functions of other more complex functions. In addition, MAVP, MACDFIX and ROCR100 are omitted due to the fact that equivalent functions, such as scaled versions, are already present in the library. Figure 4.3 depicts an example of a few technical indicator values calculated on a single stock. Bollinger Bands return two time series that are two standard deviations away from the moving average (MVA), which is a measure of volatility. Moving Average Convergence Divergence

[2]Technical Analysis library - `http://ta-lib.org/`



(a) (b) (c)

Figure 4.3: Bollinger bands (a), MVA (blue solid), +2STD (red dashed), -2STD (green dashed); MACD (b, blue solid), EMA (red dashed), DIV (green dotted); Williams' %R (c); for Tesco, Nov 2012 - May 2013

(MACD) subtracts the 26-day exponential moving average (EMA) from the 12-day EMA. This is used as a rough heuristic by traders as follows: when the MACD is above the signal line, it recommends buying while when it is below, it recommends selling. A dramatic rise suggests a small speculative bubble and end of the current trend. Williams' %R compares the closing price to the high-low range over 14 days. A high value indicates that a stock is oversold, while a low value shows that it is overbought.

The first six months of the dataset is reserved to bootstrap the indicator calculation. Most indicators rely on a history of market data for analysis so as to give forecasts. As the TAlib suite provides a large range of technical analysis indicators, we process them in a simple and agnostic manner to derive the feature representation of our data. The parameters of most indicators were left at their default values. In cases where an indicator returns multiple value series, one feature was created for each series, and indicators returning invalid or constant values were discarded. Overall, this resulted in a dataset with $T = 133$ features, representing the market conditions for a given company on a given trading day.

4.4 Validation

In order to evaluate the performance of the algorithm, a sliding window experimental setup is used, as illustrated in Figure 4.4. For each evaluation round, one and a half years of data across all companies is used for training, the following month is used for validation and the subsequent month is used for testing. In the case of time-varying models, the weights from the most recent month are used for validation and testing. Validation is only performed on the first window

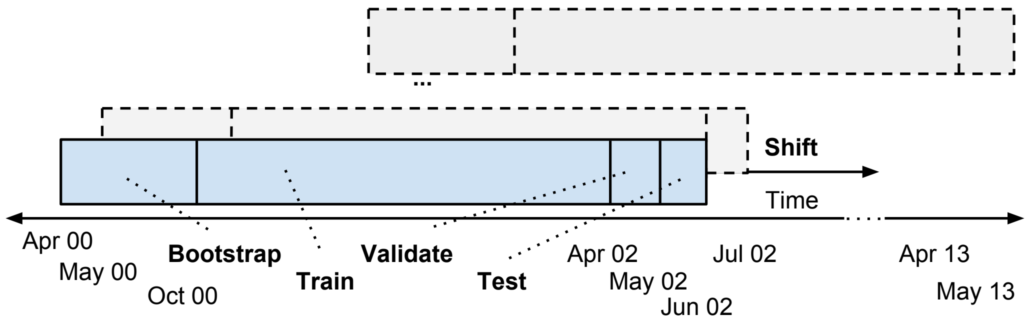


Figure 4.4: Sliding window evaluation

of data for efficiency reasons, and is used to select the three regularization coefficients which control the importance of the penalty terms of the loss function. After the first validation and evaluation, the training and testing window is shifted by one month and the process is repeated until the end of the dataset is reached in 2013. This evaluation setup is designed to match a trading scenario, where short term extrapolation predictions are needed to guide investment decisions.

The model has several parameters: the weight, λ_l, company, λ_c, and time, λ_m, regularizer coefficients (see equations 4.5, 4.7 and 4.8). These need to be tuned to control the relative effect of the data fit versus the regularization for weight magnitude, deviation from the market mean, and weight change with time, respectively. These parameters are automatically tuned by grid search using a logarithmic scale between 10^{-10} and 10^{10}. In addition, 0 and infinity are added to the possible values to simulate independent task learning and pooling. The performance of the model is measured on the validation set, and the values that give the highest validation profit are selected.

We evaluate against several baseline measures:

Random: In random trading, the predictions are produced by a uniform distribution between -1 and +1. In this case the expected predictions are 0 and consequently the expected trading profit is also 0.

Buy-and-Hold: Setting all predictions to +1 is similar to a long term buy-and-hold position, but without reinvestment of daily profits or losses. A buy and hold strategy can be seen as spreading the risk between all possessed assets, in this case the FTSE index, which confers diversification benefits as advocated by Modern Portfolio Theory.

Short-and-Hold: There is an always sell strategy which is the inverse of buy-and-hold, above. This corresponds to a long term short position by short selling the stock.

Ridge Regression: To test the importance of profit maximization, we consider also ridge regression which instead minimizes squared error subject to an L_2 regularization term. In this case we do not perform multi-task learning over companies or time, but instead fit a single regression model per company and ignore temporal variation. For fairness of evaluation, we use the sliding window evaluation and validation method for fitting the regularization hyper-parameter, as described in §4.4.

4.5 Evaluation

Our experimental validation seeks to provide empirical answers to several research questions: whether our approach outperforms simple baselines, the importance of using a profit objective, the importance of technical analysis features, and how multi-task learning affects performance, both over individual companies and over time. We now address each of these questions in turn.

The overall profit results are shown in Table 4.2, where the models in the top portion of the table using our techniques for profit maximization, and in the bottom portion, the baseline techniques. We present other evaluation metrics, including annual return in §4.5.1. It is clear that the profit-trained models (profits of £45.3–£64.9) outperform the baselines (-£26.2–£26.2) by a large margin. During the testing period the market went through peaks and troughs, including two market crashes, with an overall gradual rise, leading to a positive profit from the buy-and-hold strategy. Despite this our method was able to achieve excess profits of up to £38.7. The best model predictions significantly outperform random trading. Significance hereinafter is accessed using the Wilcoxon ranked-sign test, $p < 0.01$. Our method has identified important patterns in the data to achieve excess profits, demonstrating that market inefficiencies do exist in historical FTSE market data and can be exploited. Comparing the ridge regression baseline against the equivalent model `Single task, tech anal` trained with a different loss function, shows that optimizing for profit outperforms squared error loss by £40.3. In fact, ridge regression performed poorly, significantly outperforming only random trading and Short-and-hold. Using the appropriate loss function was clearly the single most important modelling decision in terms of net profit. Note however that using the compounding or non-compounding formulation of profit made little difference, with compounding under-performing by £2.8, although this did speed up training.

A related question regards the utility of technical analysis features (as described in §4.3). Our intuition was that many of these features could be useful, and using many together would provide a rich and expressive basis for (non-linear) modelling and thus outperform a linear autoregressive model. The rows in Table 4.2 labelled `Single task, tech anal` and `Single task, window of returns` differ in their feature representation: the former uses our 133 technical analysis features, while the window of returns uses as features the market history for the past 90 days, i.e., an autoregressive model with a 90 day time lag. The difference in profit between the two systems is modest but statistically significant, which shows evidence of the utility of technical analysis. An advantage of using technical indicators is that they can perform non-linear transformations on the market data. Therefore, using this basis allows our linear model to learn non-linear relationships over this time-series data. An interesting extension would be to allow for non-linear functions, which could remove the need for hand-engineered technical analysis features.

Next, we consider the importance of multi-task learning. Starting with the `Single task, tech anal` we can see that multi-task learning over companies (`Multi task, tech anal, comp reg`) provides a statistically significant improvement of £10.8 over independent models per company. Moreover, the temporal regularizer (`Multi task, tech anal, time reg`) also results in a significant gain of £6.6, showing that our method for modelling non-stationarity is effective. Together the two multi-task regularizers (`Multi task, tech anal, comp-time reg`) provide a significant profit increase of £15.6 over the single task method. The values learned for the regularizer coefficients were $\lambda_c = 10$ (company), $\lambda_m = 10^4$ (time), and $\lambda_l = 10^{-4}$ (weight magnitude). Note that none of these values are extreme, illustrating that multi-task learning is favored over independent learning or pooling. The magnitude of this improvement suggests that the two regularization methods work in complementary ways, and both identify important aspects in our data.

By examining the feature weights during testing, it is possible to determine which weights were important and how they contributed to predictions. For the top positive and negative indicators by weights sampled at random intervals for randomly selected companies CPI and BG, see Table 4.3. The fact that some of the top indicators are different in various tasks

Method	Absolute profit
Multi task, tech anal, comp-time reg	**£64.9**
Multi task, tech anal, comp-time reg, compound prof	£62.1
Multi task, tech anal, comp reg	£61.1
Multi task, tech anal, time reg	£55.9
Single task, tech anal	£49.3
Single task, window of returns	£45.3
Buy-and-hold	£26.2
Ridge regression	£9.0
Random	£0.0±4.6
Short-and-hold	£-26.2

Table 4.2: Trading results based on a £1 budget, evaluated on 2002-2013 FTSE data over 62 companies

Company	Date	Top positive	Top negative
CPI	2004-03-31	MACDEXT	CDLCLOSINGMARUBOZU
CPI	2005-08-31	CDLEVENINGSTAR	CDLCLOSINGMARUBOZU
CPI	2008-08-31	CDLHIKKAKE	CDLHOMINGPIGEON
CPI	2009-07-31	CDLPIERCING	CDLCLOSINGMARUBOZU
CPI	2009-11-30	ADXR	CDLCLOSINGMARUBOZU
CPI	2010-03-31	CCI	CDLCLOSINGMARUBOZU
BG	2004-07-31	CDLHANGINGMAN	CDLCLOSINGMARUBOZU
BG	2006-06-30	CDLHANGINGMAN	CDLCLOSINGMARUBOZU
BG	2006-08-31	CDLHANGINGMAN	CDLCLOSINGMARUBOZU
BG	2008-09-30	HT DCPERIOD	HT PHASOR
BG	2011-02-28	HT SINE	STOCHRSI
BG	2012-10-31	HT DCPERIOD	STOCH

Table 4.3: Top indicators for Capita plc and BG Group plc with multitask learning

justifies having an individual model for each company separately and relating them to the market average model via the company regularizer. It also shows that the time regularization can provide additional flexibility in the model. However, note that the top negative feature persisted for a long time for many companies. The tasks were not forced to share the same weights as in single task learning, but rather they could learn their own weights which provided better predictions during testing.

The Hilbert Transform is a technique used to generate in-phase and quadrature components of a de-trended real-valued signal, such as price series, in order to analyze variations of the instantaneous phase and amplitude. An interesting note is that HT PHASOR, Hilbert Transform - Phase components, frequently appeared in the top four positive indicators for companies during 2002-03 while it also appeared in the top four negative indicators for some of the same companies during 2005-06. The HT DCPERIOD, or Hilbert Transform - Dominant Cycle Period, is an adaptive stochastic indicator. Given a cyclic price signal, it attempts to identify the beginning and end of the cycle. CCI, or Commodity Channel Index, measures the current price level relative to an average price level over a period of time. It can be used to track the beginning of a new trend or warn of extreme conditions. Both HT DCPERIOD and CCI had positive weights which suggests that they can provide information with regard to the beginning of new price trends.

Another interesting observation is the inclusion of behavioral pattern matching indicators starting with the CDL prefix. CDLCLOSINGMARUBOZU frequently appeared in the top negative indicators for many companies during various periods. Marubozu is positive when the closing price was at a high during the period, indicating a bull market, and negative when it was at a low, indicating a bear market. Hence, a negative weight would contribute to a SELL signal when positive and to a BUY signal when negative. This could indicate trend reversal in stock prices or a reversion to longer term average returns. CDLHANGINGMAN was another top positive feature. It indicates that despite a large sell-off, the buyers remain in control and manage to push the prices further up in the short term before an eventual drop. With a positive weight it has the effect of buying stock shortly before the peak of the price trend, which is still a good time to do so.

Next, the change of the feature weights is examined over time. Figure 4.5 demonstrates that the predictive weight of feature CDLLONGLINE was fairly negative during the initial testing period, but later it became positive. On the other hand, HT PHASOR behaved the opposite way. Similarly, the weight of CDLSHOOTINGSTAR started with negative value, then during the middle of the testing period it became positive, and at the end it went back to negative. On the other hand, the weights of CDLINNECK seemed to vary periodically across the time scale. These examples justify the use of time regularization as they show that there are temporal changes in market conditions to which the feature weights can adapt.

Last, the importance of each feature group is determined, which could result in the omission of certain features and a simpler model consequently. Table 4.4 contains ablation analysis of each technical analysis feature group in terms of predictive power. This shows that the Pattern Matching indicators had the most predictive power during the experiments, with Momentum

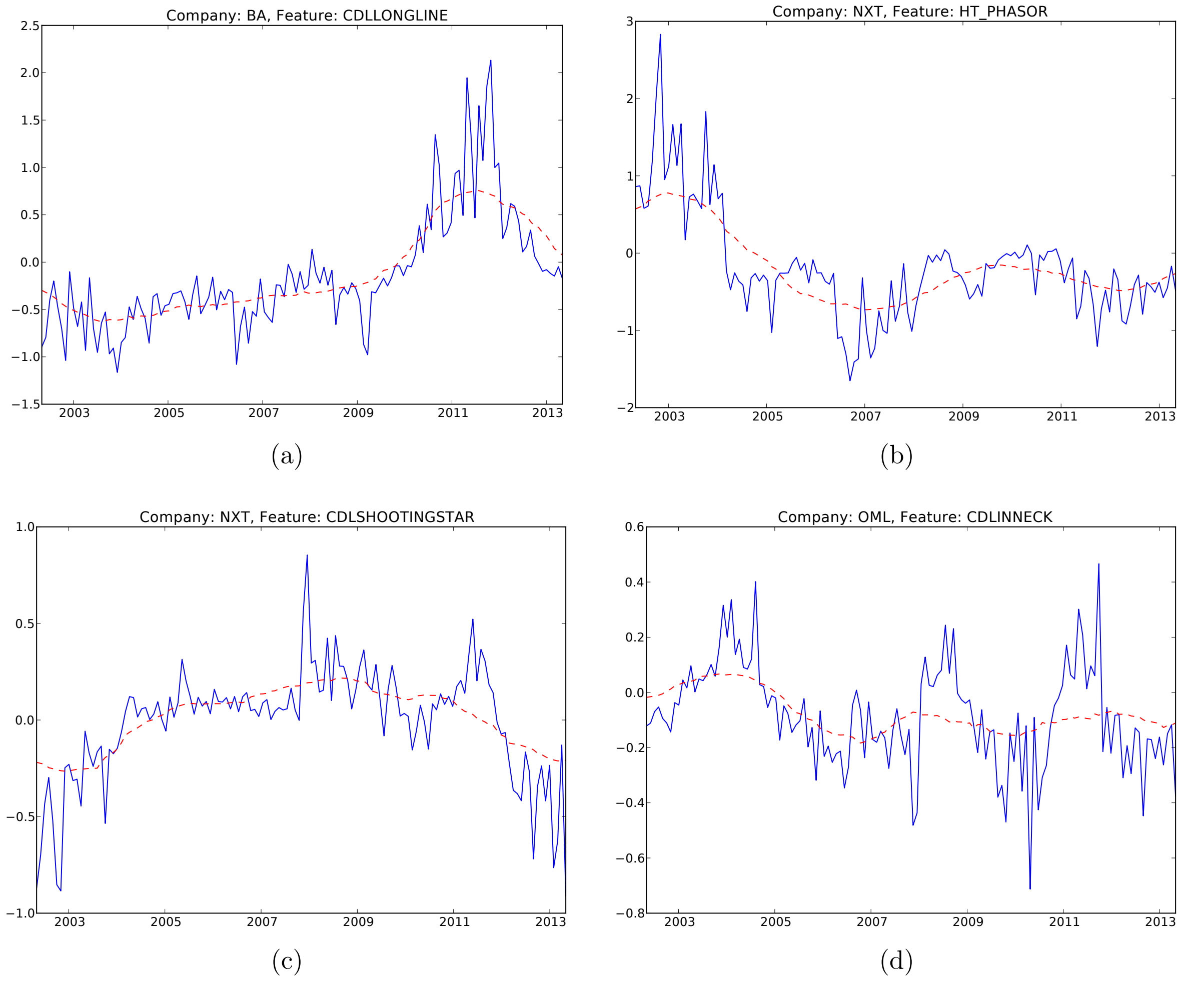


Figure 4.5: Time variations in feature weights (blue line) with 30-month moving average (red dotted line) with multi task learning

Feature group	Absolute profit
All	**£64.9**
Pattern Recognition	£54.8
Momentum Indicators	£47.4
Overlap Studies	£37.8
Volume Indicators	£27.1
Cycle Indicators	£21.2
Price Transform	£9.0
Volatility Indicators	£-5.7

Table 4.4: Trading with selected feature groups

indicators coming close second. On the other hand, Volatility indicators were not very useful in predicting profitability, but all indicators together still gave a solid boost in performance.

4.5.1 Trading Simulation

The experimental results reported above show the efficacy of our proposed modelling technique. However the above evaluation used a simplistic trading setting which does not correspond to the conditions an investor would face on the market. Now we seek to augment the evaluation to cope with real market conditions, by 1) maintaining a running budget to determine the amount invested each day, 2) disallowing short-selling and 3) including transaction costs. Note that we allow for fractional stocks to be traded, which is also unrealistic, but would have only a negligible effect when trading with a sufficiently large budget. Together these changes provide a more realistic evaluation of the trading profits and losses.

First, consider the trading amount, which previously was fixed to range between £-1 and £1. Here instead we allow the budget accumulate throughout the testing period, and scale all trades by the funds available each day. This limits the downside during a run of poor performance, as investments become proportionally smaller, while also increasing the profits (and risk) after sustained successful trading. This evaluation method corresponds to the compounding training objective U' in (4.4). However for simplicity in this section we evaluate only the model trained without compounding, denoted `Multi task, tech anal, comp-time reg` in §4.5. The results of trading with a cumulative balance are shown in Figure 4.6. Even with two significant market crashes, the algorithm made a loss only temporarily. On average the algorithm made a profit of 101% which is equivalent to a 6.53% annual return, exceeding the baseline buy and hold strategy which returned 22% profit overall and 1.8% annually.

Secondly, although short selling is permitted on many financial markets, it is a controversial practice and is subject to several restrictions and costs. In brief, short selling involves borrowing a stock from a broker, which is immediately sold on the market. The transaction is closed later by repurchasing the stock to repay the debt to the broker. This strategy can be used to profit in a falling market, however, as it involves borrowing with an unlimited potential for loss, it is not a widely available service and typically incurs significant costs and collateral requirements. For these reasons we now consider evaluation where short-selling is disallowed. This allows for the algorithm to be applied to a much wider range of markets and stocks. To support this change, we also allow stocks to be held in long positions, and maintain daily cash and stock balances for each security. This contrasts with the evaluation in §4.5 where each trade was reversed on the following day, such that no ongoing positions were held. The revised evaluation process starts with a budget of £1 for each security, split evenly between stock and cash. Buy predictions $p(\mathbf{x}, \mathbf{w}) > 0$ are applied to the cash balance, proportionally to the prediction magnitude, which is then used to increase the investment in the stock. Sell predictions $p(\mathbf{x}, \mathbf{w}) < 0$ are applied proportionally to the stock balance, thus depleting the stock and increasing the cash balance. Note that as both balances are tracked separately and will often differ in value, and consequently the same magnitude prediction for buy versus sell may result in a different value trade. This trading strategy is denoted **trading position**. For comparison we also present two modifications to this trading strategy: first **fixed lot** trading which scales each trade, by a fixed constant, e.g., 1%. This helps to limit extreme trading behavior where all or none of the budget is invested. The other modification is **rebalancing** which equalizes the value of stock and cash in each account before applying the trade. This restores the symmetry of buying and selling, and limits the exposure to large losses or gains. Note that **fixed lot** is orthogonal to **rebalancing**, and we evaluate using both techniques together.

The final change to our evaluation method is to include **transaction costs** at 0.6% of the transaction value. The results of the trading strategy simulations are shown in Table 4.5, where each trade was executed using the closing price of each day. As expected transaction costs tend to erode the profits, however this was not the case with some trading strategies. In particular, with a fixed 1% lot size, the algorithm still made a substantial profit. When combined with the

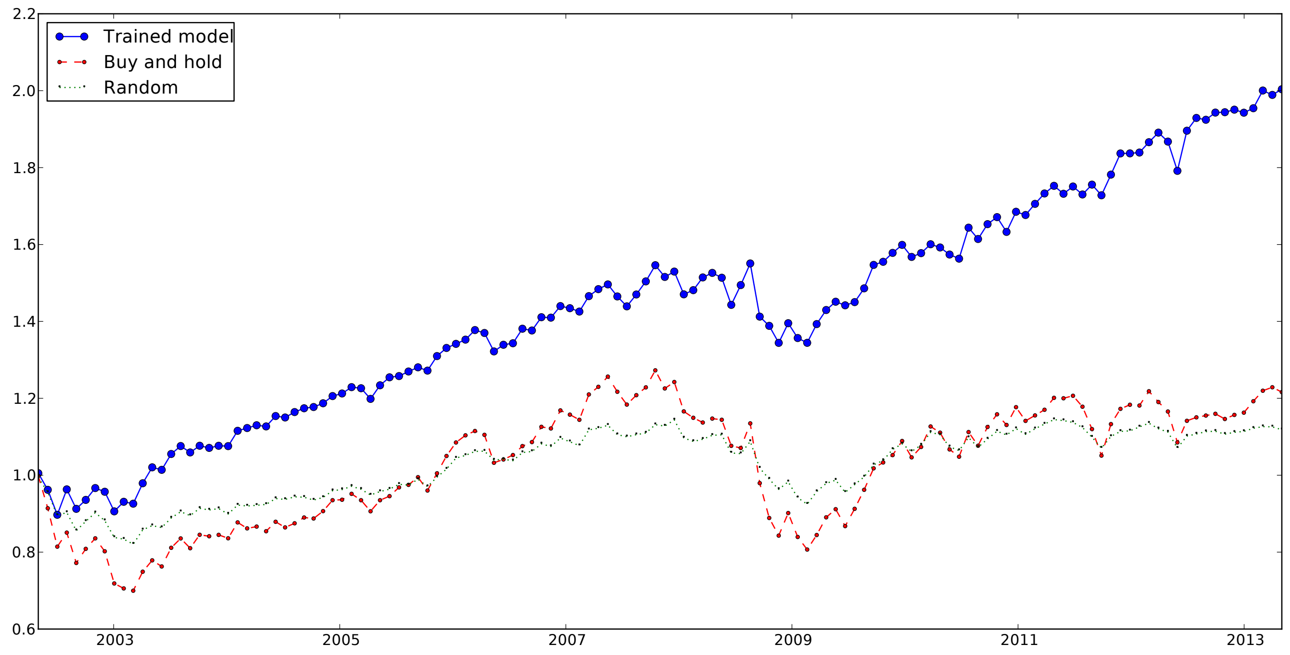


Figure 4.6: Balance evolution, or cumulative profits, of various trading strategies

rebalancing strategy the profits were even greater than positional, which is even more surprising considering the costs levied on the rebalancing transactions. What this suggests is that it is important to realize gains and losses quickly. With a positional trading strategy with 100% lot size, early trades can have a large affect on the leverage of later trades. While rebalancing is also similarly affected by compounding balances, it is overall more conservative and maintains a more diversified portfolio. The algorithm may have predicted small changes in the relative price movement accurately, but these changes were not enough to offset the transaction costs, which were not included in the model's training objective. However, when the transaction costs were zero, then there was no drawback of predicting small price changes and thus a 100% lot strategy produced more profit than 1% lot. Note that the algorithm came very close to the buy and hold strategy in the transaction cost case even though it suffered significant transaction costs compared to the practically no transaction costs of the former. In future work, we plan to model transaction costs directly in the objective of the algorithm.

Performance metrics

Modern Portfolio Theory gives various measures to evaluate the performance of a trading strategy. According to the Capital Asset Pricing Model (Sharpe, 1964) in order to evaluate

Transaction Cost	Trading Position	Lot Size	Prediction Model	End Balance
0%	Positional	100%	Trained Model	**£2.01**
0%	Positional	100%	Buy-and-hold	£1.22
0%	Positional	100%	Random	£1.11
0%	Positional	1%	Trained Model	£1.18
0%	Rebalance	1%	Trained Model	£1.33
0.6%	Positional	100%	Trained Model	£0.00
0.6%	Positional	100%	Buy-and-hold	£1.21
0.6%	Positional	100%	Random	£0.00
0.6%	Positional	1%	Trained Model	£1.12
0.6%	Rebalance	1%	Trained Model	£1.20

Table 4.5: Trading simulations on test data with £1 starting balance, 2002-2013, 62 companies

Metric	Value
Alpha	4.3%
Beta	0.54
Sharpe	1.52

Table 4.6: Annualized performance metrics of own strategy compared to baseline buy and hold, risk adjusted by 3 month sterling UK treasury bills discount rate, 2002-2013

whether an investment is worth the capital, the investor must be compensated for the time value of his capital and for the risk of losing the investment, also known as risk premium.

Jensen's alpha (Jensen, 1968), α_a, can be interpreted as how much the fund strategy outperforms the baseline investment strategy risk adjusted, $\alpha_a = r_a - (r_f + \beta_a(r_m - r_f))$, where r_a is the returns of portfolio or own strategy, r_f is the risk free returns, 3 month UK treasury bond yields in sterling, where ee used the monthly average 3-month sterling treasury bill discount rate data for 2002-2013 as reported by the Bank of England [3], r_m is the market or baseline returns, always buy or buy and hold. A value above 0 means that the algorithm was able to beat the market in the long term without taking excess risk. Beta, β_a, measures how volatile or risky the strategy is compared to the baseline, $\beta_a = \frac{\sigma(r_a, r_m)}{\sigma^2(r_m)}$, where σ is the covariance function. The Sharpe ratio measures excess return adjusted by risk (Sharpe, 1998), $S = \frac{r_a - r_f}{\sigma(r_a)}$
.

The results are summarized in Table 4.6. An annual rate of 4.3% was calculated for alpha, meaning that the algorithm generated risk free excess profits in the long term, putting it in the 95% percentile of funds as measured by a *t*-distribution of 115 fund performances over

[3]`http://www.bankofengland.co.uk/`

20 years (Jensen, 1968). Beta was measured on the monthly time series of the buy and hold and the own portfolio returns, where 1 denotes the market risk. An annualized value of 0.54 means the algorithm returns were considerably less risky than the market returns. The Sharpe ratio of 1.52 was computed based on the monthly standard deviation of returns, noting that 1 is considered good, 2 very good and 3 excellent (Lo, 2002). In summary, all three measures confirm the presence of risk adjusted profits from our algorithm's trading predictions.

4.6 Conclusions

This chapter has demonstrated that stock market price movements are predictable, and patterns of market movements can be exploited to realize excess profits over passive trading strategies. We developed a model of daily stock trading of several stocks, and a means of training to directly maximize trading profit. This results in consistent risk-free profit when evaluated on more than a decade of market data for several UK companies, beating strong baselines including buy-and-hold and linear regression. Beyond individual stock modelling, we presented a multi-task learning approach to account for temporal variations in market conditions as well as relationships between companies, both demonstrating further improvements in trading profit. Technical analysis indicators, in particular from the pattern matching and momentum family, were found to have better predicting power than plain historical returns calculated on a window of adjacent trading days. Finally, we demonstrated in realistic trading scenarios that the algorithm was capable of producing a profit when including transaction costs. For a list of company symbols and technical indicators used in this experiment, please refer to Appendix A.1 and A.2.

In this chapter we have introduced a linear model with profit optimisation objective. Linking back to the research questions in chapter 1, we showed via these experiments that technical analysis contains predictive power for forecasting markets. We also found evidence for temporal dynamics by constructing temporal multi task regularizers and achieving superior results. Similar improvements could be obtained via domain adoption in the form of company regularizers across the FTSE constituents. We were able to obtain improved performance by using TA indicators instead of plain historical returns as input, and we were also able to uncover non-linearities in the data via a piecewise linear approximation of each task, which produced a global non-linear signal across the whole market. Last, we found evidence that the optimization objective is very important as our profit objective considerably beat the linear regression baseline.

In the next chapter 5, we will extend this linear model with a Bayesian treatment of the optimal weights distribution. In addition, we will examine how to incorporate the model in the powerful Gaussian Process framework, which allows for the automatic identification of relevant technical analysis indicators.

Chapter 5

Kernelised Bayesian Market Prediction

In the previous chapter, we demonstrated a linear model to trade around 100 established blue-chip stocks on the London Stock Exchange for the test period of more than a decade. We showed that consistent risk free profits can be obtained.

In this chapter we take the linear model further. One limitation of the previous model was that it takes the point estimate solution (or maximum likelihood estimate) for the optimum parameters. However, financial markets are very noisy systems, and likely when we have limited data available, integrating over a distribution of possible hypotheses that explain the data yields better prediction accuracy. Therefore, in our first model we outline a Bayesian treatment of the profit maximisation objective in the linear model, and show improvement over a MAP estimate solution. Next, we show how we can integrate the profit as a pseudo likelihood in the Gaussian Process framework, and use Gaussian approximation methods to do trading with GPs. A Gaussian Process is a kernelised framework that is more powerful than linear models. We show that with a GP, the predictive performance further improves. In addition, by applying Automatic Relevance Determinates to the kernel, we are able to isolate the most effective technical analysis indicators.

In the second part of the chapter, we investigate a different dataset related to credit risk prediction of small and medium sized business. This dataset is multi modal, containing time information for loan applications, as well as fundamentals of the companies that applied for borrowing, and user generated text in the form of comments filled out by potential lenders evaluating the business. We construct a Gaussian Process with a Rational Quadratic kernel that allows us to identify multiple overlapping patterns in the data related to the distribution of interest rates on the market. In addition, we propose a new kernel combination technique in order to effectively combine and weigh all the different types of data sources in our dataset for optimal predictions. We show that our GP achieves good accuracy in predicting the trading interest rates of these loans, which are a sign of credit risk, and beats other models based on Support Vector Machines. Last, we demonstrate a way to exploit the uncertainty of the predictions via optimising for trading profits by spotting arbitrage opportunities. This is done by a separate optimisation procedure fitted on top of the GP and achieves significant profits in our simulations.

In conclusion, this chapter demonstrates several ways we can apply Bayesian machine learning techniques to forecast markets in various domains and discover inefficiencies that may exist and produce excess risk adjusted profits.

5.1 Bayesian Profit Maximisation

The potential for profit from modelling stock movements on financial markets based purely on publicly available information is a hotly contested debate. Here we show how the investment actions of active traders can be framed as a simple learning objective, which allows for trading models to be learned efficiently from historical data. We develop a linear model, relating technical analysis features over the recent price fluctuations to investment decisions, before considering Bayesian extensions using the Laplace approximation, and a kernel treatment in a Gaussian Process model. The linear model builds on the work presented in chapter 4 but here we also provide a novel probabilistic interpretation of it. In summary, this work provides a strong empirical justification for technical analysis, through consistent improvements of the learned models over a buy-and-hold baseline and gains from Bayesian treatments over point estimates.

5.1.1 Introduction

Stock price movements appear to be chaotic and unpredictable, and mainstream theories of finance (Malkiel, 2003) refute the possibility of realizing risk free profit through predictive modelling based solely on public information. Despite this, a large body of technical analysis work (Lo et al., 2000) maintains that price movements can be predicted solely from historical price data. Our research hypothesis is that markets are not completely efficient, and we aim to empirically test this using predictive modelling.

Traders traditionally look at candlestick charting, also known as technical analysis, to predict future price movements of securities using expert judgement (Lo et al., 2000). In this chapter, similar to the previous one, we consider learning a predictive model to uncover hidden patterns from past historical data instead of relying on expert judgement. An important question is the choice of training objective for fitting models to financial data: standard regression objectives such as squared loss would be inappropriate for the reason that profit and prediction accuracy are often not significantly correlated in a financial context. Here we present a novel objective based on the profit derived from an active trading strategy, which is also used in prediction contexts for deciding what trading actions to make on unseen data.

The contributions of the work are as follows:

- A novel learning algorithm with a profit maximisation objective is developed that is suitable for modelling trading decisions on the stock market. In this context, portfolio management is framed using a linear model, resulting in an objective similar in functional form to logistic regression, allowing for standard gradient based optimisation. In addition, the equivalence to a probabilistic interpretation of profit is proved.
- Next, we consider a Bayesian treatment instead of a point estimate, to better incorporate uncertainty and make more robust predictions in the presence of noise. We show how the Laplace approximation can be applied to the profit objective, and used for tractable posterior inference.
- Finally, we consider a kernelised Bayesian approach using Gaussian Processes. In this instance the profit objective takes the place of the data likelihood, allowing inference via the Laplace approximation or Expectation Propagation. This brings the flexibility to incorporate rich kernels, and allows model selection through optimisation of kernel hyper-parameters.

Overall we show that technical analysis contains predictive power for forecasting daily returns, demonstrating substantial empirical improvements over buy-and-hold or random trading baselines. Moreover Bayesian modelling provides further substantial gains over point estimates, particularly Gaussian Processes which result in highly significant profits.

5.1.2 Profit Maximisation

Consider an active trading scenario, where a trader makes an investment every day which is then reversed the following day. We have a amount of capital for each stock and the problem is how much to invest (or sell) based on a real valued signal. Equivalently, we make a probabilistic decision whether to buy ($+a$) or sell ($-a$), before making a trade (see §4.1). We assume short selling is allowed, i.e., stocks that are not currently held can be sold and then later purchased. We impose a limit on the value of short selling of $-a$. The sign of the signal enables the holding of long (positive), and short (negative) positions, whereas the magnitude or probability denotes the confidence level of the prediction. In this setting, we consider a linear model in which we assume the signal and stock for a given day are dependent on the linear combination of real valued technical analysis features and their weights, as detailed in §5.1.4.

In the next section, we review two ways of exploiting the trading signal, the first is proportional, which is similar to the model outlined in chapter 4, but here we decompose the trading signal into two orthogonal components. The second one is probabilistic, which makes use of this decompositions and thus we show it results in the same expected return as the proportional strategy, even though it is more aggressive in investing capital.

Proportional Trading

In the proportional trading setting, each investment decision is made proportional to the signal,

$$h(\mathbf{x}, \mathbf{w}) = \tanh(\mathbf{w}^\intercal \mathbf{x}) \tag{5.1}$$

where the linear signal $\mathbf{w}^\top \mathbf{x}$ is squashed through the hyperbolic sigmoid which maps the values into the range $[-1, 1]$. These extrema correspond to the maximum divestment (sell) and maximum investment (buy), respectively, and points in between correspond to less confident trades where only a fraction of the budget is allocated.

The signal has two components: $i(\mathbf{x}, \mathbf{w}) = \text{sgn}(h(\mathbf{x}, \mathbf{w}))$ which is the buy-sell decision and $a(\mathbf{x}, \mathbf{w}) = |h(\mathbf{x}, \mathbf{w})|$ which is the invested capital. The target is the daily returns, $y(t) = \frac{s_{close}(t+1) - s_{close}(t)}{s_{close}(t)}$, the relative change in closing price from day t to $t+1$. The objective of training is to learn the parameters $\mathbf{w}$ that lead to high profits. Intuitively, this means matching the sign of $h(\mathbf{x}, \mathbf{w})$ and $y(t)$, and also making high magnitude predictions, $|h(\mathbf{x}, \mathbf{w})|$, whenever sure of the direction of price movement – particularly important when $|y(t)|$ is high.

More formally, the daily profit (utility) for a given day can be expressed as

$$u(\mathbf{x}, \mathbf{w}, y) = h(\mathbf{x}, \mathbf{w})y \tag{5.2}$$

The training process for optimising u for $\mathbf{w}$ is described below. For more information on this model, please refer to chapter 4.

Probabilistic Trading

Here we show how the proportional strategy presented above can be interpreted in a probabilistic framework. Probabilistic trading may be more attractive when the trader has a fixed amount of capital to invest, while proportional trading might benefit more from reduced trading volume. In addition, we will see that having a probabilistic interpretation of profit can be beneficial when incorporating the profit objective as a likelihood term in a Bayesian framework.

The proportional trading strategy is equivalent to a probabilistic binary trading scenario in which the trader first randomly decides whether to buy or sell from a Bernoulli distribution, before making a trade using their full amount of capital (a or $-a$). The Bernoulli is governed by our model, with parameter $p = \sigma(\mathbf{x}^\top \mathbf{w})$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the logistic sigmoid. We now show that this strategy results in the same profit objective as for proportional trading shown in Equation 5.2.

Under the probabilistic investment strategy, the expected utility can be expressed as

$$\mathbb{E}[u(\mathbf{x}, \mathbf{w})] = \sigma(\mathbf{x}^\top \mathbf{w})y + \left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right)(-y)$$

where the left and right summands correspond to buying and selling, respectively, weighted by reward $+y$ or $-y$. This expression can be further simplified to

$$\begin{aligned}\mathbb{E}[u(\mathbf{x},\mathbf{w})] &= \big(2\sigma(\mathbf{x}^\top\mathbf{w}) - 1\big)\, y \\ &= \tanh(\mathbf{x}^\top\mathbf{w})y\,,\end{aligned}$$

matching the formulation for proportional trading in Equation 5.2. □

Training Objective

A more common training objective for real valued price movements is the squared error, as optimised in most regression algorithms. While this may be suitable for other datasets, in our setting of financial decision making we can formulate the expected profit directly, resulting in a very different objective. To illustrate the difference between squared error and expected profit loss functions, consider when the price changes significantly but the model predicts 0. In this case the sum of squared errors is high, but there is no trading gain or loss. Penalising this conservative baseline to the same extent as mistaking the sign of the price movement is unwarranted, and consequently squared error loss is inappropriate in this setting. Even if we consider the opportunity cost of the missed trade, the shape of the profit function is likely different from the squared loss. Although the optimum of both functions theoretically are at the same point, in practice when dealing with approximate learning problems, the achieved local minimum will depend on the function shape around the manifold of the data.

To formulate the full training objective, we consider the profit over a time series of $n = 1 \dots N$ price movements. In this case the profits can compound, i.e., each day the budget is invested, and the proceeds of this trade gives rise to the budget for the subsequent day. The compound profit objective is formulated as

$$\mathcal{L} = -\ln \prod_{n=1}^{N} \left(u(\mathbf{x_n}, \mathbf{w}, y_n) + 1\right) + \lambda \|\mathbf{w}\|_2^2 \tag{5.3}$$

where 1 is added to the expected daily utility defined in Equation 5.2 to reflect the proportional change to the investment budget, while the logarithm is used to simplify computation. The second term is an L_2 regulariser which smooths weights towards zero, while λ modulates its effect relative to the profit.

Training consists of minimising Equation 5.3 using gradient based methods, here we employ the L-BFGS algorithm (Byrd et al., 1995). Although the objective is non-convex, our experiments showed that this optimisation method was effective, especially when used with random restarts. The gradient for a single weight is given by

$$\frac{\partial}{\partial w_d}\mathcal{L} = -\frac{\sum_{i=1}^{N} y_i x_{i,d}\,\mathrm{sech}^2(\mathbf{w}^\top \mathbf{x_i}) \prod_{j=1:j\neq i}^{N} y_j \tanh(\mathbf{w}^\top \mathbf{x_j}) + 1}{\prod_{k=1}^{N} y_k \tanh(\mathbf{w}^\top \mathbf{x_k}) + 1} + 2\lambda w_d \tag{5.4}$$

where $\mathrm{sech}(x)$ is the hyperbolic secant function. In the next section, we show we can integrate this profit objective in a Bayesian framework to integrate over the potential parameter space of the model.

5.1.3 Bayesian Inference

In trying to fit a predictive model to small samples of noisy financial data, overfitting is an important concern. Moreover, the posterior of the parameter space is very broad giving rise to many feasible explanations of data. In this case, it is dangerous to take the point estimate, which may overconfidently predict large magnitude investments and for this reason we now consider Bayesian approaches for reasoning under uncertainty.

Laplace Approximation

We now consider Bayesian inference under the linear model described in §5.1.2, in place of maximum a-posteriori (MAP) parameter estimation, above. Here we use the Laplace approximation which approximates the posterior using a Gaussian centered at the MAP solution, and with the negative inverse Hessian as the covariance matrix (Rogers and Girolami, 2011). This is a consequence of taking the 2nd order Taylor series expansion of the function at the MAP solution, resulting in a Gaussian posterior,

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) \approx \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{5.5}$$

$$\begin{aligned} \text{where} \quad \mu &= \hat{\mathbf{w}} \\ \mathbf{\Sigma^{-1}} &= -\left(\frac{\partial^2 \ln \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y})}{\partial \mathbf{w} \partial \mathbf{w}^\top}\right)\Bigg|_{\hat{\mathbf{w}}} \end{aligned}$$

where $\mathcal{L}$ is the pseudo likelihood function for which we use the cost function from Equation 5.3 and $\hat{\mathbf{w}}$ is the MAP weight estimate from solving Equation 5.3. Note that although this is not strictly a likelihood, it is closely linked to a probabilistic formulation as shown in §5.1.2.

A second problem is making predictions on test data. Given the approximate posterior in Equation 5.5, it is still not possible to formulate the exact Bayesian prediction, as no conjugacy exists between the hyperbolic sigmoid and the Normal distribution. For this reason we resort to Monte Carlo sampling,

$$h(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} h(\mathbf{x}, \mathbf{W}_k), \;\; \mathbf{W}_k \sim \mathcal{N}(\mu, \boldsymbol{\Sigma}) \tag{5.6}$$

where a sufficiently large number of $K = 10,000$ weight samples are drawn, the prediction for each sample is computed, and the results averaged. When there is a lot of mass either side of μ, the weighted averaged prediction can significantly differ from the point estimate prediction. That is because the tanh is a non-linear link function where different $\mathbf{W}$ lead to different areas of activations in the non-linearity, as a consequence of which the average prediction will be different from just activating the link function at the MAP. For example, other reasonable parameter values might predict reverse trades and it would be unreasonable to make a confident trade using just one (MAP) parameter setting in this case. Whereas if many of the likely parameter vectors from the posterior predict similar trades, then this is a much more robust indicator for trading.

Gaussian Process

Going from a linear to a kernelised Bayesian model in the previous sections, we now investigate a Gaussian Process approach to profit maximisation. A Gaussian Process is a stochastic process where function realisations consist of random values associated with every point in time and space following a normal distribution. For a more thorough review of GPs, please refer to section 3.2.

In the linear model, $\mathbf{w}^\top\mathbf{x}$, is now replaced by the latent functions f, the prior of which is assumed to be a Gaussian Process. Integrating over the latent functions, the marginal likelihood is given by

$$\begin{aligned} p(\mathbf{y}|\mathbf{f}) &= \mathcal{N}(\mathbf{f}, \sigma^2 I_N) \\ p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}) \mathrm{d}\mathbf{f} \\ &= \mathcal{N}(0, \mathbf{K} + \sigma^2 I_N) \end{aligned}$$

This is a GP with covariance matrix $\mathbf{K}_{N\times N}(\mathbf{X})$ and $m(\mathbf{X}) = \mathbf{0}$ mean. The predictive mean and variance are given by,

$$\begin{aligned} \mu_* &= \mathbf{k}_*^\top (\mathbf{K} + \sigma^2 I)^{-1} \mathbf{y} \\ \Sigma_* &= \mathbf{k}_{**} - \mathbf{k}_*^\top (\mathbf{K} + \sigma^2 I)^{-1} \mathbf{k}_* \end{aligned}$$

We use a linear covariance function (Williams and Rasmussen, 2006), defined as

$$k(x, x') = \sigma + \sum_{d=1}^{D} \lambda_d x_d x'_d$$

where $\sigma \geq 0$ controls underlying covariance and $\lambda_d \geq 0$ the covariance arising from each feature. We consider tied hyper-parameters, $\lambda_d = \lambda$, as well as separate values for each input dimension (so called automatic relevance determination or ARD). Using the tied version with a GP is equivalent to the Bayesian linear profit maximisation model in §5.1.3. The hyper-parameter values are learned by optimising for the type II Maximum Likelihood using L-BFGS.

Profit Pseudo Likelihood

Instead of using a Gaussian likelihood for regression, we implement the profit maximisation objective as the pseudo likelihood of the GP inside the GPy toolkit.[1] This is not a likelihood as such, although it does have a probabilistic interpretation as we have seen, and it can still be used in the GP framework as being an unnormalised log probability density. It is approximated by Laplace's method to yield a Gaussian distribution compatible with the GP prior and posterior distributions so as to enable probabilistic inference. The log likelihood of the profit can be

[1]`http://sheffieldml.github.io/GPy/`

written as

$$\ln \mathcal{L}(\mathbf{y}|\mathbf{f}) = \ln \prod_{n=1}^{N} (\tanh(f_n)y_n + 1) \tag{5.7}$$

The hyperbolic tangent transformation is applied through the hyperbolic co-tangent reversed link function, $\coth(x) = \tanh(x)^{-1}$. For obtaining the highest marginal likelihood, the hyper parameter optimisation procedure requires the gradients up to the third order which are easy to calculate. Due to memory constraints, a randomly sampled subset of the training data was used. In the sparse GP approximation with Expectation Propagation, these were selected as inducing points with k-means clustering (Minka, 2001; Hensman et al., 2013a). Similarly to the Bayesian linear model, predictions are approximated using Monte Carlo averaging. We sample the latent functions $K = 10,000$ times and average their outputs after the link function transformation is applied.

5.1.4 Technical Analysis Features

The calculation of technical analysis features is done similar to the method described in chapter 4. Here, we briefly review this.

The goal of technical analysis is to find informative patterns for stock price prediction from noisy time series financial data in order to predict future price movement (Lo et al., 2000). The dataset includes daily market data of long term constituents of the FTSE 100 index. This includes 62 companies between 2011-2013. We perform technical analysis on each company dataset using the TA-lib technical analysis library.[2] This transforms market price time series of (open, high, low, close, volume) values with a family of technical analysis functions, which form a feature representation of the data. After processing our data for technical analysis features, we normalise each feature to have unit variance and zero mean.

Figure 5.1 depicts an example of a few technical indicator values calculated on a single stock. Bollinger Bands return two time series that are two standard deviations away from the moving average (MVA), which is a measure of volatility. Moving Average Convergence Divergence (MACD) subtracts the 26-day exponential moving average (EMA) from the 12-day EMA. This is used as a rough heuristic by traders as follows: buy when the MACD is above the signal line, and sell when it is below. A dramatic rise suggests a small speculative bubble, while a

[2] `http://ta-lib.org/`

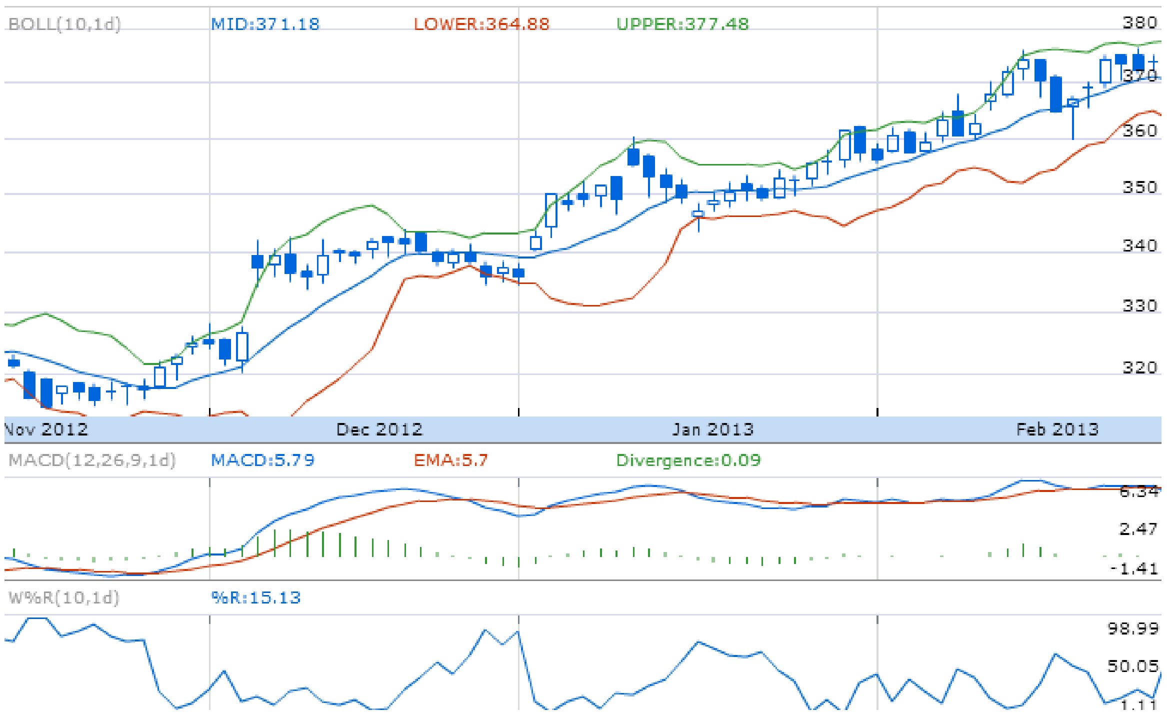


Figure 5.1: Bollinger bands (top) two standard deviations away from price series, plus (green) and minus (red); MACD (mid, blue), EMA (red), divergence (green); Williams %R (bottom, blue); for Tesco, Nov 2012 - Mar 2013. Source: Google Finance.

divergence suggests the end of the current trend. Williams %R is said to compare the closing price to the high-low range over 14 days. A high value indicates that a stock is oversold, while a low value shows that it is overbought.

Indicators are applied from the family of overlap studies, momentum, volume, cycle, price transform, volatility and pattern recognition indicators. As the TA-lib suite provides a large range of technical analysis indicators, we process them in an agnostic manner to derive the feature representation of our data. In case an indicator returns multiple value series, one feature is created for each series. After processing we obtain $D = 133$ dimensional feature vectors for our data.

Validation

The first six months of the dataset is reserved to bootstrap the calculation of the technical analysis features, some of which require long term price data to compute e.g., moving average statistics. For training, we use 18 months of price data for all companies. The following month is used for validation and the last month is used for testing, as shown in Figure 5.2. This setup is repeated 62 times by sliding the data window over our full dataset. Results are reported over the aggregated 62 test months.

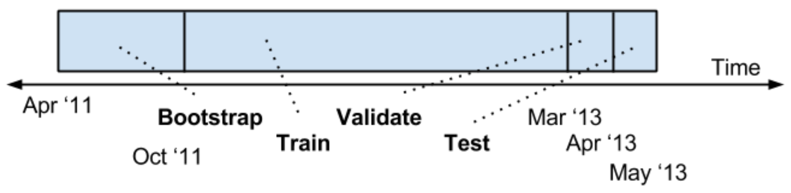


Figure 5.2: Sliding window evaluation

The model needs to tune the L_2 regulariser coefficient, which controls the relative effect of the data fit versus the regularisation term (see Equation 5.3). This is done automatically by random search (Bergstra and Bengio, 2012), which has been shown to outperform standard grid search. The process works as follows: the parameter value is drawn from a logarithmic uniform distribution 100 times randomly between 10^{-10} and 10^{10}, the performance of the model is measured on the validation set, and the value that gives the highest performance is selected.

We implement four baseline measures:

Random trading: where the predictions are produced by a uniform distribution between -1 and +1. Here the trading profit is expected to be zero, because the expected value of the predictions is 0 and is uncorrelated to y.

Buy-and-hold: This strategy is implemented as always buying, i.e., the predictions are +1 for all days. Trades and their subsequent reverse trade balance out, such that this is equivalent to buying initially, then holding the stock and only selling after the last trading day.

Short-and-hold: As with buy-and-hold, but selling such that predictions are always -1. This corresponds to a long term short position.

Ridge regression: This strategy predicts price changes using a simple linear regression model with an L2 regulariser.

5.1.5 Evaluation

The evaluation is performed in the probabilistic trading context. The test set performance is measured by

$$U_* = \sum_{n=1}^{N_*} \operatorname{sgn}\left[h(\mathbf{x_{n*}})\right] y_{n*} \tag{5.8}$$

where $h(\mathbf{x_{n*}})$ is the prediction function, based either on the parametric MAP estimate or a sampled average prediction from the Laplace or GP models. Note that Equation 5.8 differs slightly from the profit objective (Equation 5.2) – here the sgn function acts as a normaliser such that the different models make predictions on similar scales.

Experimental results in Table 5.1 show the overall profit on the test data, which consisted of 1240 data points and assuming that £1 was traded each day. Note that all models outperform all three baselines, and the significance can be observed relative to random trading compared to which all our models' results are at least two standard deviations better. Our profit based models also outperform ridge regression, confirming the need to directly maximise profit in model training. The Bayesian Laplace approximation implemented in the Gaussian Process framework outperforms the MAP point estimate, confirming the need for full posterior Bayesian inference. The automatic relevance determination kernel improves the results further by allowing for a varying noise parameter for each feature.

Table 5.1: Absolute profit on test set, last month across all companies

Model	Absolute profit (£)
Gaussian process	**3.19**
Laplace	1.58
MAP estimate	1.22
Short sell	0.18
Random	0.0±0.51
Ridge regression	0.48
Buy and hold	-0.18

A natural question is the how well these models perform with different quantities of training data. This is illustrated in the learning curve in Figure 5.3. For datasets with 400 points or greater, the Gaussian process model outperforms the MAP point estimate on the test set. The learning curve was measured up to 2500 training data points on a logarithmic scale and the test set performance was averaged over 50 runs with different random training samples and plotted with the standard error. With this averaging, random trading performed with mean 0 and standard deviation 0.16.

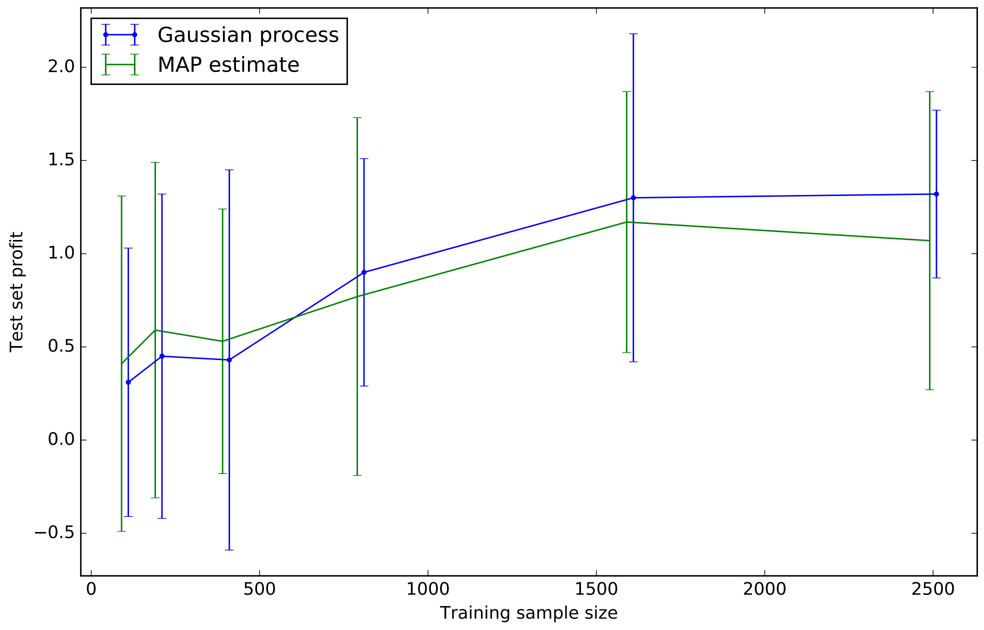


Figure 5.3: Learning curve of Gaussian process and MAP models with 100,200,400,800,1600,2500 randomly sampled training datapoints, averaged over 50 runs, with standard error.

Finally, we inspect the actual predictions and the resulting trades the models made. Figure 5.4 shows an example training run and the resulting test set performance of the Gaussian process, the Bayesian linear and the MAP estimate compared with buy-and-hold. We can see that all models have difficulties with similar instances. As a result, the profit they accumulate is roughly synchronised. However, the Gaussian process makes fewer mistakes on average, resulting in a higher overall profit in the end than that of the MAP model. Passive investment represented by the buy-and-hold strategy actually loses capital during the simulation.

Last, it is worth noting that due to the flexibility of Gaussian processes, they are prone to overfit the training data even though GPs have built in model selection via measuring the marginal likelihood of the data. We have found that the squared exponential kernel (radial basis function) kernel did not perform consistently well, often underperforming the linear model presented above. This finding may be a consequence of the large number of hyperparameters in these models, leading to optimisation difficulties for the non-convex objective, as well as

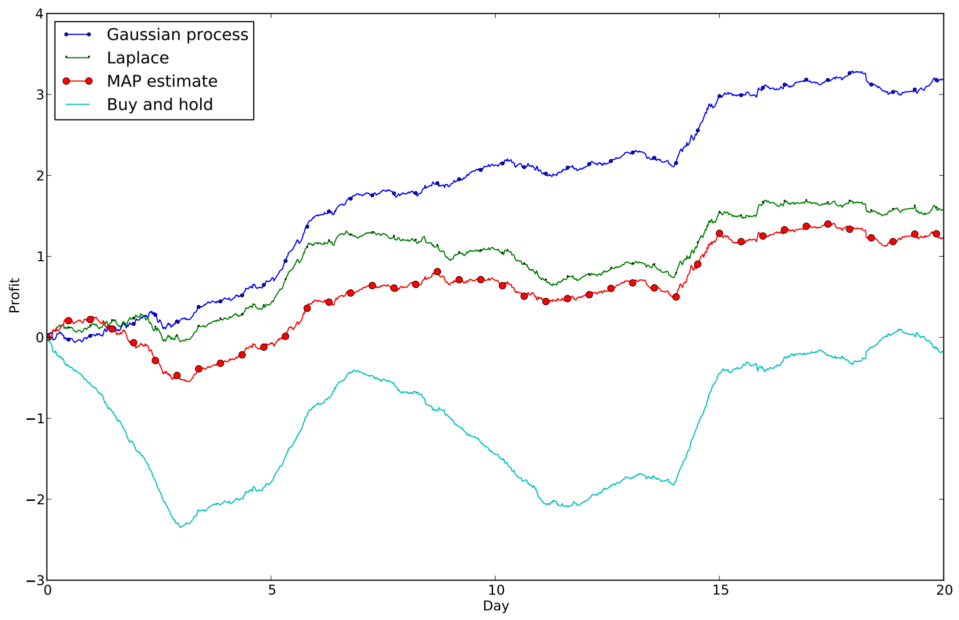


Figure 5.4: Cumulative test profit pooled across all companies, 2000 training data points

overfitting. From a Bayesian model selection approach, we observed that the simpler linear kernel achieved higher marginal likelihood on the very noisy training data in most cases, and also higher test set performance.

5.1.6 Conclusion

The results give support for the validity of technical analysis and show that stock markets are predictable. A profit maximization algorithm has been developed that made a consistent profit by day trading over 62 companies for one month, beating the buy-and-hold baselines, as well as random trading and linear regression. A Bayesian treatment of the objective first with Laplace Approximation of the Maximum A Posteriori solution, then in the framework of Gaussian Process has been shown to improve predictive performance further on the test data. In the future, multi task learning in the form of coregionalization can be investigated. Multi task learning has to potential to uncover relationships between companies, while still allowing variations between them. We also plan to investigate non-stationary models to allow modelling of changing temporal dynamics in financial time series.

For a list of company symbols and technical indicators used in this experiment, please refer to Appendix A.1 and A.2. In the next section, we apply the Gaussian Process framework to perform fundamental analysis of peer-to-peer loans. In this problem, we make use of various datasources, including time, text, financial data, and show a novel kernel technique to combine them under a joint posterior. Then we predict interest rates with the combined non-linear kernel and derive a profit maximisation objective fitted to the GP posterior. Finally, we achieve significant profits in a trading simulation.

5.2 Non-Linear Bayesian Credit Risk Prediction

In the previous section we saw a derivation of trading profit in a Bayesian context, which we then integrated into the Gaussian Process framework. Next, we design a non-linear Bayesian credit risk prediction model for small and medium sized businesses, using multiple datasources, and fit a profit maximisation model on top of the GP posterior. This shows that fundamental analysis contains predictive power in financial market forecasting, where risk free excess profits can be achieved.

Peer-to-peer lending is a new highly liquid market for debt, which is rapidly growing in popularity. Here we consider modelling market rates, developing a non-linear Gaussian Process regression method which incorporates both structured data and unstructured text from the loan application. We show that the peer-to-peer market is predictable, and identify a small set of key factors with high predictive power. Our approach outperforms baseline methods for predicting market rates, and generates substantial profit in a trading simulation.

5.2.1 Introduction

Peer-to-peer (P2P) lending is an emerging market that facilitates lending between individuals and small-medium sized companies. One characteristic of these markets is that they exhibit highly non-linear behavior, with substantial noise. Temporal dynamics can be observed as well due to varying loan terms and changing market conditions. Last, social lending involves significant quantities of user generated text which is an integral part of the loan application process. All this key information can be incorporated in the Gaussian Process framework which can handle non-linear mapping functions. These latent functions are well suited to handle noise, and the posterior of the model supports exploitation of market inefficiencies via automatic trading.

In this chapter we present a Gaussian Process regression model for predicting market loan rates, which infers an underlying latent function linking the attributes of each loan and its rate, while accounting for the considerable noise in the data. This models not just the rates for unseen loans, but also the uncertainty of the predictive posterior, which allows for full Bayesian inference and a form of outlier detection, namely, to detect mispriced loans on the market, and quantify expected profits that could be realized by buying and reselling loan parts at correct market value.

The Gaussian Process is a state of the art machine learning framework which allows for rich modelling of several aspects of the data. We incorporate temporal variation, structured data, and unstructured text using a kernel combination technique that modulates between addition and multiplication. We find that this modulation beats a model with purely additive or multiplicative kernel combination.

Each loan application contains rich information about the borrower, third party credit scoring, textual information, such as a questions and answers section as well as a cover letter from the borrower explaining their background and the purpose of the loan. There are around 1,000 – 1,500 words of English text per document. In order to harness the semantic content from this text, we use LDA topic modelling and TF/IDF scoring of terms. We find that our model outperforms baselines and other well known linear and non-linear learning algorithms such as support vector regression. Note that the dataset used in this work is new and was collected by the authors themselves, which is one of the contributions of this research.

5.2.2 Model

Here we use a Gaussian Process (GP) regression model, a Bayesian kernelized extension of linear regression which models Gaussian covariance between pairs of datapoints (Williams and Rasmussen, 2006). For a detailed overview of GPs, please refer to section 3.2. A GP prior defines a latent set of random values over an input space (here feature vectors), where any finite values of variables follow a multi-variate Gaussian distribution (here noise-corrected loan rates). GP regression is formulated as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$
$$y \sim \mathcal{N}(f(\mathbf{x}), \sigma_n^2)$$

where $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ is the mean function and $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$ is the covariance function, which are both user defined functions. The observations y are assumed to be noise-corrupted versions of the latent process, where σ_n^2 is the variance of the Gaussian noise. We follow standard practice by letting $m = 0$, while we consider a suite of covariance functions designed to work with the various aspects of our input (structured, temporal, text). Given a labelled training sample, $(\mathbf{X}, \mathbf{y})$, consisting of $(\mathbf{x}, y)$ pairs, we can express the Bayesian posterior, and integrate out the latent function values, $\mathbf{f} = f(\mathbf{X})$, resulting in an analytic formulation for the marginal likelihood, $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f}$. Although this involves the inversion of the covariance matrix with complexity $O(n^3)$ where n is the number of training instances, scaling to larger datasets can be accommodated using approximation methods with linear or sub-linear time and space complexity in n (Williams and Rasmussen, 2006; Hensman et al., 2013a). Maximizing the log marginal likelihood using gradient ascent allows for the noise parameter σ_n^2, as well as other model hyperparameters (parameterizing k, described below) to be optimized.

The choice of the covariance function, k, varies depending on the application and the nature of the features. Here we use the rational quadratic covariance function (Williams and Rasmussen, 2006) defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_d^2 \left\{ 1 + \sum_{d=1}^{D} \frac{(x_{id} - x_{jd})^2}{2\alpha l^2} \right\}^{-\alpha} \tag{5.9}$$

where D is the number of features, and σ_d^2, l and α are the kernel hyper-parameters, named amplitude, lengthscale and power respectively. This covariance function is equivalent to an infinite sum over radial basis function (RBF) kernels of different lengthscales; setting $\alpha = \infty$ recovers the RBF. Accordingly, this kernel is appropriate if patterns are present in the data occurring over different lengthscales, e.g., global correlations combined with local smoothness, such as long term trend in loan rates and short term variations in trading patterns. For some features we use Automatic Relevance Determination (Neal, 1995) with the above covariance function, which uses a different scale parameter for each input dimension, i.e., the l^2 term in

(5.9) is replaced with l_d^2. Accordingly, the ARD identifies the relevance of features based on their relative lengthscales, with shorter scales indicating more rapid fluctuations in the output with the feature, and accordingly higher importance. This way, it can be used for effective feature selection.

Since we have different types of features extracted from the data, the question remains as to how best to combine these. Multi modal features can be integrated by adding, multiplying or by convolving other valid covariance functions over the data (Williams and Rasmussen, 2006). For this reason, we define the addmul kernel operator that modulates between kernel addition and multiplication,

$$\text{addmul}(k_1, ..., k_n) = \prod_{i=1}^{n} (k_i + b_i) \tag{5.10}$$

where k is an input kernel (defined over input pairs, omitted for clarity) and b is a bias constant. Expanding this formulation reveals a weighted sum of multiplicative combinations of the input kernels. Collectively the bias, $\mathbf{b}$, and scaling hyperparameters (σ_d^2 for each kernel, see Equation 5.9) determine the weighting of each kernel combination term. The intuition behind the modulated kernel is that we do not want to hardcode either multiplication or addition over the kernels but rather naturally find a balance between these operators, fit by optimizing the Bayesian marginal likelihood. This comes down to the difference between arithmetic and geometric averaging, namely that while the arithmetic (addition) average smooths out component variation, the geometric (multiplication) allows each component to significantly affect the overall result, e.g., a component kernel close to zero will lead to overall ~zero covariance. Having all views agreeing is desirable, however this is unlikely to happen often in practice, hence modulating the geometric mean to consider combinations of fewer components is a more beneficial strategy. This means the modulated kernel can express standard kernel addition and multiplication, as well as products over subsets.

Text is often high dimensional, owing to a large vocabulary size, which complicates the use of ARD with per-word features due to twin issues of optimization convergence and overfitting. Therefore, in order to prevent overfitting the high dimensional input space, we apply the standard rational quadratic kernel with isotropic lengthscales over the textual feature groups (LDA and TF/IDF) and the ARD rational quadratic over the smaller dimensional time and numeric features groups, where it is more likely to be effective at feature selection. These kernels are combined with the addmul operator,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \text{addmul}(k_t, k_n, k_l, k_f) \tag{5.11}$$

where K is the covariance between two $\mathbf{x}$ loan instances, k_t is the kernel over the time features, k_n is the kernel over the numeric feature group, k_l is the kernel over the LDA topic distributions and k_f is the kernel over the TF/IDF scores. Together the bias terms, each kernel's σ_d, l and α values, and the Gaussian noise variance σ_n^2 form the model hyperparameters, which are fit by optimizing the marginal likelihood.

After training, which involves fitting the model hyperparameters to maximize the marginal likelihood of a supervised training sample, we seek to make predictions on unseen data. Under GP regression, the predictive posterior over the loan rate, y_*, for test point $\mathbf{x}_*$, is another Gaussian, $p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ which can be analytically computed (see (Williams and Rasmussen, 2006) for details). We use the predictive mean for evaluating predictive error rates, but as a secondary evaluation consider its use in a trading scenario. In a trading context, assume we buy a loan part with observed rate $\hat{y}$ and attempt to resell it later at rate y. For maximum profit, we want to maximize the difference between these two rates, $U(y|\hat{y}) = \hat{y} - y$, which quantifies the extent to which the loan is underpriced. Assuming that the market rate is distributed according to the GP posterior, we can quantify the expected profit as

$$\begin{aligned} \mathbb{E}[U(y|\hat{y})] &= \Pr(y_* \leq y) U(y|\hat{y}) \\ &= \Phi(y|\hat{\mu}, \hat{\sigma}^2) U(y|\hat{y}) \end{aligned} \tag{5.12}$$

where $\Phi(y|\hat{\mu}, \hat{\sigma}^2)$ is the Gaussian cumulative density function of the posterior. The optimizing y is then found using gradient ascent of (5.12), which we use to price the loan part for sale. Alternatively, the expected profit can be employed to identify the best trading opportunities across several different candidate loans. The optimization can be performed with respect to markup as well, which can be derived from the rate and other loan properties. A visualisation of the expected profit can be seen in Figure 5.5.

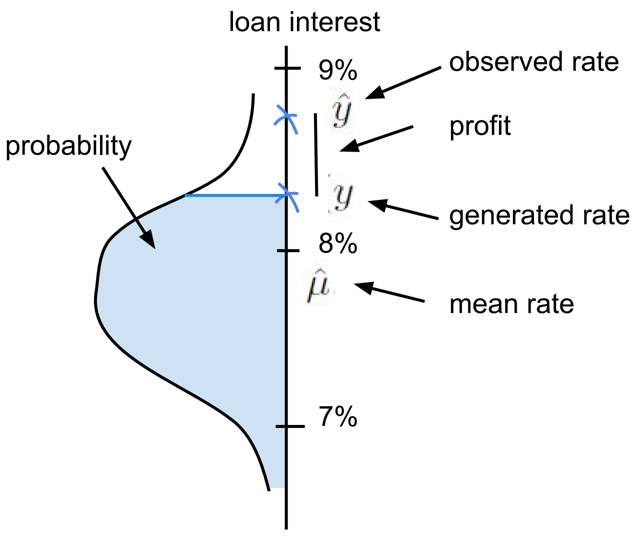


Figure 5.5: Visualisation of expected profit.

5.2.3 Features

The feature groups used in this experiment are shown in Table 5.2, we now elaborate on each of these features.

Numeric
closing rate
min accepted bid
max accepted bid
Altman ratio 1-5
term length
trading years
amount requested
bad debt rate
credit history
relative score

Time
auction date
scrape date
elapsed time
LDA
100 topic freqs
TF-IDF
$\sim$ 3000 word scores

Table 5.2: Input feature groups

Structured Features

It is hypothesized that key loan information such the requested amount, the term length of the loan and the closing interest rate affect the buyer rates of the loans. Historical credit scores are likewise considered an important factor since they influence the probability of default. A number of crucial financial indicators can be extracted from the annual filed management accounts of the companies for the last few years, and based on this information, a trader can infer the health of the business. We extract the Altman financial ratios (Atiya, 2001) as an indicator of bankruptcy probability for small and medium sized companies, similar to those listed on Funding Circle. For more details, please refer to section 2.2.5.

The more money a borrower requests, the less time they have been trading, and the longer they need the money, the more riskier the application is perceived. The estimated annual bad debt is Funding Circle's assessment of the business's health, and therefore included.

The closing rate and the bid spread gives indication as to how other investors have judged the loan application, and will likely influence the buyer rate as it is these investors that will sell the loan parts initially. It is likely that further information exists in the depths of bids and the investor identities, and thus we perform text analysis both with and without these identities.

Other potential modelling improvements could come from using the categorical data to support multi task learning, e.g., using coregionalization or hierarchical kernels, to uncover richer relationships in the data, such as explicit modelling of different sectors or individual investors.

The auction closing date and scrape date give information about changing market conditions at various points in time. The elapsed time since the closing date indicates how long the borrower has been repaying the loan, which could influence the probability of future default. This is a basic form of survival analysis, where loans that survive for a longer time are considered less risky and thus have a lower buyer rate.

Unstructured Features

The Questions and Answers section is a form of crowd due diligence, the purpose of which is to provide additional assessment of the loan application by the wisdom of the crowds. An example of this can be seen in Figure 2.4. In addition, many of the people asking questions also trade on the secondary market, giving insight into their valuation of the loan. The description section gives information about the motives of the borrower and their background, which affects the probability that they will not be able to repay the money.

TF/IDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus (Rajaraman and Ullman, 2011). It is often used as a weighting factor in information retrieval and text mining. The TF/IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others. For a full definition of TF/IDF, please refer to section 3.1.4. For example, in Table 5.3 we can see that "aircraft," "roof" and "nightclub" are highly specific and important terms to the documents, but "explain" and "small" are much more general terms and accordingly we expect them to be of much less use for our application of loan rate modelling. The presence of specific lenders in an auction is likely to correlate with the secondary market loan rates, which raises the possibility of modelling individual investors as part of our analysis. Therefore, we include the investor usernames who bid on the loan or pose questions in the Q&A section as words, which are used in computing the TF/IDF scores and LDA topics, described below.

Before computing word scores, the text is pre-processed by filtering out punctuation, and replacing numerical values with a placeholder tag. Then, word tokens are stemmed using Porter stemmer (Porter, 1980) and stop words are removed. Next, words with a count of less than 20 and the top 100 words are removed from the dataset. Finally, TF/IDF scores are calculated for each remaining word type, resulting in an approximately 3,000 dimensional vector space representation of each loan document.

aircraft	roof	nightclub
machin	scaffold	bar
aerospac	fibr	floor
program	min	tabl
agreement	tile	music
explain	explain	run
gener	larg	larg
small	go	much
level	small	circl
next	get	number

construct	hous	independ
sector	beer	currenc
score	brew	scottish
tax	pub	affect
deliv	northern	repair
debt	per	paid
contractor	week	hous
tender	breweri	run
recent	weekli	vend
may	craft	car

Table 5.3: Top/bottom 5 TF/IDF words from selected documents (left) and top 10 words from LDA topics (right).

Latent Dirichlet Allocation (LDA; (Blei et al., 2003)) is a soft document and word clustering method based on a latent document representation. It posits that each document is a mixture of a small number of topics where each word's creation is attributable to one of the document's topics.

We use LDA to infer for each loan document a distribution over 100 latent topics, where each topic is a scalar value between 0 and 1, summing to 1 across all topics. The vector of topic assignments are then used as features of the loan, which is incorporated into the model using a rational quadratic kernel, as described above. Since questions and answers are fairly sparse, we pool these fields together with the borrower description. However, it may be useful to separate these or align terms from the questions to the answers. LDA captures the semantic content of loan requests which reflects the themes lenders and borrowers are concerned about when contemplating investment decisions. In Table 5.3 we can see that Scottish independence is one such issue. The presence of certain investors in the topics may be indicative of latent investment preferences, for example, for certain industries or risk bands.

5.2.4 Results

We fit a Gaussian Process over the normalized data with 4-fold crossvalidation and measure the root mean square error and marginal likelihood. The marginal likelihood can be employed for Bayesian model selection to choose between competing hypotheses (Williams and Rasmussen, 2006), as detailed in section 3.2.4. It includes both a data fit term and a model complexity penalty, thereby penalizing overly permissive models. Moreover, this is evaluated solely on the training data, and thereby avoids the need for heldout cross-validation. As baselines we include predicting the closing rate of the auction as the secondary market buyer rate, and predicting the average buyer rate in the training set. In addition to GP regression, support vector regression (SVR) with linear and RBF kernels, and ridge regression are evaluated for comparison. The hyperparameters are tuned using grid search over a logarithmic scale.

The results in Table 5.4 show that the model confidently beats the baselines (1.352), linear regression (1.004), as well as SVR (0.970), with a root mean square error of **0.596**. Note that the scale of the output variable ranges between 4-15 as can be seen in Figure 2.11. The best GP kernel is the rational quadratic, which beats the linear and RBF kernels. This suggests that there are multiple overlapping patterns in the dataset that the RatQuad kernel can identify, but RBF cannot due to lack of flexibility. Table 5.5 shows comparative results of the GP RatQuad model. Note the close correspondence between maximizing marginal likelihood and minimizing test RMSE and that model selection will select the model with the lowest error (0.596). The addmul operator outperforms both addition and multiplication as a way to

Description	**RMSE**	**log $\mathcal{L}$**
Baseline Average Buyer Rate	1.780	-
Baseline Closing Rate	1.352	-
Ridge Regression	1.004	-
SVR Linear	0.970	-
SVR RBF	0.994	-
GP Linear	0.738	-1116
GP RBF	0.607	-847
GP RatQuad	**0.596**	-811

Table 5.4: Main results, showing baseline (top) versus modelling approaches (bottom). The columns report cross-validation RMSE and training marginal likelihood.

Description	**RMSE**	**log $\mathcal{L}$**
Operator		
Add	0.689	-1127
Prod	0.699	-1382
Addmul	**0.596**	-811
Features		
Time	1.599	-3069
Numeric	0.771	-1379
Numeric Top 6	0.764	-1373
Time+Numeric Top 6	0.602	-842
LDA	1.225	-2516
TFIDF	0.972	-2328
TFIDF+LDA	0.936	-2146
Numeric+LDA+TFIDF	0.680	-1084
Time+Numeric+LDA+TFIDF	**0.596**	-811

Table 5.5: Comparative results of the GP RatQuad model with different kernel configurations.

combine kernels, suggesting that modulation between addition and multiplication is effective. Textual features are shown to contain important information, with models using only text features (e.g., TFIDF+LDA) outperforming the baselines and competing SVR approaches. When combined with structured numeric features, they result in further improvements. The temporal kernel on its own is not very effective, but combining the time kernel with the other kernels resulted in significant improvements. Last, it can be seen that LDA and TF/IDF features have identified complementary signals in the data to the numerical and temporal components. When combining all features, the best model accuracy is achieved.

With respect to the importance of numerical features in predicting buyer rate, as can be seen in

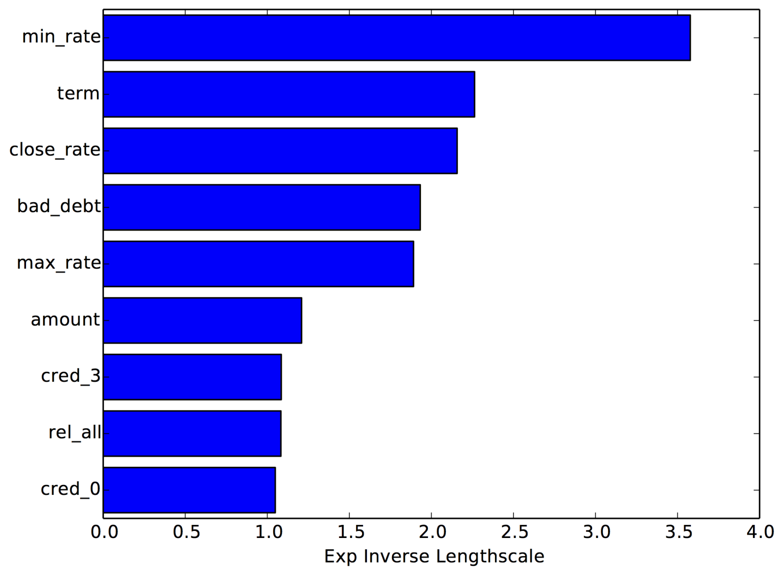


Figure 5.6: Most important numerical features, showing the log inverse length scale hyperparameters. Larger values mean greater contribution to the covariance.

Figure 5.6, we find that the most important features are the closing rate (`close_rate`) and the spread of the accepted bids, where the minimum accepted bid (`min_rate`) is weighted more than the maximum (`max_rate`). Other important features are the term length of the loan (`term`) and the bad debt ratio of the risk band for the loan (`bad_debt`). The amount requested is slightly less important (`amount`). In the credit score features, we see the average company score across all companies dominates (`rel_all`), which is an indicator of overall economic environment. With regard to monthly credit history, we find that the most important ones are the current score (`cred_0`) and the one three months ago (`cred_3`). In fact, we find that ARD is effective at feature selection, and rerunning the numeric model with only the top 6 features selected results in slightly improved predictive accuracy (see `Numeric Top 6` in Table 5.5). In addition, it appears that there is some amount of information content overlap between the various feature groups, as the key numeric features combined with time do fairly well. Traditional important financial metrics such as credit scores, profit and loss accounts and borrower personal statements however only bear marginal influence on market rates, which questions the usefulness of this aspect of fundamental analysis. Alternatively, most of this information may already be contained in Funding Circle's credit score (`bad_debt`).In the text based models, we find that the key indicators are the identities of the individual investors committed to certain loan auctions, which suggests the model is able to uncover latent correlations of investment patterns and preferences of the users.

Next, we simulate trading on the secondary market over a unit of time. This acts as a proof of concept to demonstrate how the model predictions could potentially be exploited. Profitability of arbitrage opportunities can be crudely calculated by assuming lack of short selling, no transaction costs and that market rates are distributed according to our model posterior. Note, if we had a time series of market depth for each loan, that would allow for more sound empirical evaluation here. The simulation starts by buying each loan part at the listed rate, and attempting to sell it above a new market rate using Equation 5.12. If a sale occurs, which we can estimate using the model posterior, we record the profit as the difference between the buy and sell rate. Otherwise, we hold on to the loan part and record zero profit. Using this procedure, we measure the compound return over the approximately 3,500 test loan parts starting with a unit of capital, resulting in an end capital of R = 98.35. This compares to a result of 0.005 for the baseline method of randomly pricing the loan parts under the posterior.

5.2.5 Conclusions

We have observed that P2P markets are predictable and we have shown a simple way to model them with a GP and maximize expected profits in a trading scenario. Via model and feature selection, as seen in Figure 5.6, we have identified the variables that explain moving secondary market rates on the Funding Circle market place. Text based features have successfully uncovered preferences of individual investors, as seen in Table 5.5, while traditionally important metrics such as profit and loss accounts, credit history and borrower personal statements have shown to be less informative. In the future, we plan to model other targets such as loan markup and the likelihood of default, deeper use of the categorical data using e.g., multi task learning (Cohn and Specia, 2013) and hierarchical kernels (Hensman et al., 2013b) and experiment with the Student's-T likelihood which is more resilient to outliers.

This concludes our experiments with Bayesian machine learning to validate technical and fundamental analysis on stock and lending markets. We have proposed several ways to extend the Gaussian Process framework and produce significant risk free profits on both markets in a test trading simulation. Linking back to the research questions in chapter 1, we found some evidence for the validity of fundamental analysis for loan rate prediction on a crowfunding market, while certain fundamental indicators, such as financial ratios were not that predictive. We tested models with unstructured text based input and found that they beat the baseline confidently after performing topic modelling and TF-IDF extraction from the text. Temporal features were also deemed important and improved upon the results. In our models, we were able to aggregate multiple sources of data, such as time, text, and numerical features. In addition, we showed non-linearities to be important via the Rational Quadratic kernel for loan rate prediction, though for the stock market experiments, linear kernels with automatic relevance determination proved better due to high amounts of noise. The Gaussian Process framework also allowed us to exploit uncertainty by integrating over possible hypotheses that explain the data and then later optimizing for profit on top of the GP posterior. In the sock market experiments, optimizing for profit via a pseudo likelihood and incorporating uncertainty turned out to improve the results. Last, GPs allowed us to learn better representation of the data via a kernel transformation, which supports the concept of end-to-end learning when combined with predicting trading profits or finding arbitrate opportunities.

In the next section, we will move on to representation learning, and show how we can construct deep compositional neural network learning machines that are able to capture complex non-linear interactions in the data, output actions we want to perform on the market, and directly optimise for profit or net worth of the trader.

Chapter 6

End-to-End Learning with Deep Artificial Neural Networks

In the previous chapter, we expanded the linear profit maximisation models with a Bayesian interpretation of profit, and modelled credit risk rates on a crowdfunding website. We also showed how trading profits can be optimised on top of a Gaussian Process prediction model.

In this chapter, we turn our attention towards Artificial Neural Networks. ANNs have the advantage of learning good feature representations for the data in progressively hierarchical steps. This exploits the fact that often higher level concepts can be composed of lower level ones. In addition, ANNs allow for encoding our profit objective directly, inputting arbitrary type of data and outputting likewise arbitrary actions. On the other hand, in contrast with Bayesian methods, they make a point estimate of the optimal parameters, therefore they are often thought to require large training datasets for this estimate to be meaningful.

In the first part of the chapter, we show a way to predict future box office revenue of movies from movie reviews and meta data, using only a small number of around 1000 training instances without overfitting, proving that ANNs can compete with Bayesian methods when data is sparse. Our proposed neural network architecture achieves a 40% improvement in accuracy over the previous state-of-the-art linear model. We also propose a novel way to interpret the inner state of the network for text based predictions, since ANNs previously were often viewed as a "black-box" in the research community.

In the second part of the chapter, we demonstrate and end-to-end learning framework to trade multiple stocks and optimise for the net worth of the trader. Our system takes multiples types of data as input, including historical market data, newspaper articles and economic indicators, uncovers temporal dynamics over several days focusing on both local and global patterns, and finally issues limit order trades directly in the market. This way, we are able to comprehensively simulate a real life trader that reads information from various sources, reasons over it and makes informed decisions reflecting their updated beliefs. Our robot trader is able to achieve significant profits over more than 17 years of test market data from two major stock exchanges, beating the baseline buy-and-hold with significance.

6.1 Revenue Forecasting with a Convolutional Neural Network

In this section, we tackle the problem of revenue forecasting for box office movies based on preliminary movie reviews. This system can be exploited in prediction markets to determine how successful a movie is going to be once released. Similar to previously explored Bayesian systems seen in chapter 5, we make use of multi modal data, including structured meta data and text. We take time into account by a sliding window training and validation, and by forecasting the revenue into the future. We show that by using deep representation learning over the combined data, we are able to achieve highly accurate predictions.

The problem of text regression has traditionally been tackled using linear models. Here we present a non-linear method based on a deep convolutional neural network. We show that despite having millions of parameters, this model can be trained on only a thousand documents, resulting in a 40% relative improvement over sparse linear models, the previous state of the art. Further, this method is flexible allowing for easy incorporation of side information such as document meta-data. Finally we present a novel technique for interpreting the effect of different text inputs on this complex non-linear model.

6.1.1 Introduction

Text regression involves predicting a real world phenomenon from textual inputs, and has been shown to be effective in domains including election results (Lampos et al., 2013), financial risk (Kogan et al., 2009) and public health (Lampos and Cristianini, 2010). Almost universally, the text regression problem has been framed as linear regression, with the modelling innovation focussed on effective regression, e.g., using Lasso penalties to promote feature sparsity (Tibshirani, 1996). Some preliminary work has shown strong results for non-linear text regression using Gaussian Process models (Lampos et al., 2014), however this approach has not been shown to scale to high dimensional inputs. For an overview of linear models for text regression, please refer to Section 3.1.4. Despite their successes, linear models are limiting: text regression problems will often involve complex interactions between textual inputs, thus requiring a non-linear approach to properly capture such phenomena. For instance, in modelling movie revenue conjunctions of features are likely to be important, e.g., a movie described as 'scary' is likely to have different effects for children's versus adult movies. While these kinds of features can be captured using explicit feature engineering, this process is tedious, limited in scope (e.g., to conjunctions) and – as we show here – can be dramatically improved by representational learning as part of a non-linear model.

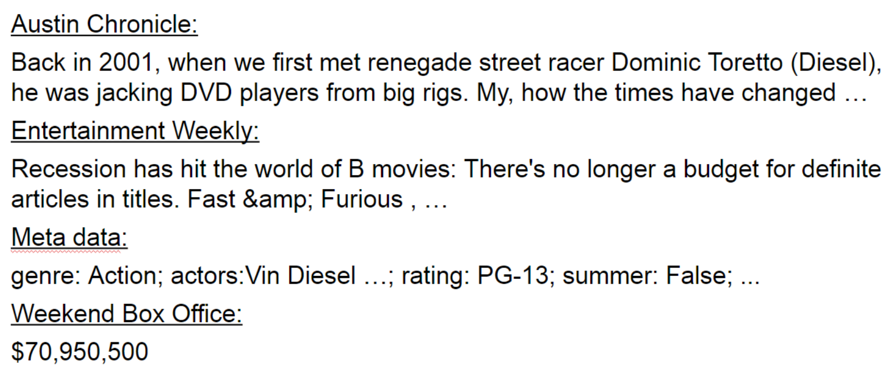
In this chapter, we propose an artificial neural network (ANN) for modelling text regression. In language processing, ANNs were first proposed for probabilistic language modelling (Bengio et al., 2003), followed by models of sentences (Kalchbrenner et al., 2014) and parsing (Socher et al., 2013) *inter alia*. These approaches have shown strong results through automatic learning dense low-dimensional distributed representations for words and other linguistic units, which have been shown to encode important aspects of language syntax and semantics. In this chapter we develop a convolutional neural network, inspired by their breakthrough results in image processing (Krizhevsky et al., 2012) and recent applications to language processing (Kalchbrenner et al., 2014; Kim, 2014). These works have mainly focused on 'big data' problems with plentiful training examples. Given their large numbers of parameters, often in the millions, one would expect that such models can only be effectively learned on very large datasets. However we show here that a complex deep convolution network can be trained on about a thousand training examples, although careful model design and regularisation is paramount.

We consider the problem of predicting the future box-office takings of movies based on reviews by movie critics and movie attributes. Our approach is based on the method and dataset of (Joshi et al., 2010), who presented a linear regression model over uni-, bi-, and tri-gram term frequency counts extracted from reviews, as well as movie and reviewer metadata. This problem is especially interesting, as comparatively few instances are available for training (see Table 6.1) while each instance (movie) includes a rich array of data including the text of several critic reviews from various review sites, as well as structured data (genre, rating, actors, etc.) (Joshi et al., 2010) found that regression purely from movie meta-data gave strong predictive accuracy, while text had a weaker but complementary signal. Their best results were achieved by domain adaptation whereby text features were conjoined with a review site identifier. Inspired by (Joshi et al., 2010) our model also operates over n-grams, $1 \leq n \leq 3$, and movie metadata, albeit using an ANN in place of their linear model. We use word embeddings to represent words in a low dimensional space (Bengio et al., 2003), a convolutional network with max-pooling to represent documents in terms of n-grams, and several fully connected hidden layers to allow for learning of complex non-linear interactions. We show that including non-linearities in the model is crucial for accurate modelling, providing a relative error reduction of 40% (MAE) over their best linear model. Our final contribution is a novel means of model interpretation. Although it is notoriously difficult to interpret the parameters of an ANN, we show a simple method of quantifying the effect of text n-grams on the prediction output. This allows for identification of the most important textual inputs, and investigation of non-linear interactions between these words and phrases in different data instances.

	train	**dev**	**test**	**total**
Austin Chronicle	306	94	62	462
Boston Globe	461	154	116	731
LA Times	610	2	13	625
Entertainment Weekly	644	208	187	1039
New York Times	878	273	224	1375
Variety	927	297	230	1454
Village Voice	953	245	198	1396
# movies	1147	317	254	1718
# reviews per movie	4.2	4.0	4.1	4.1
# sentences per movie	95	84	82	91
# words per movie	2879	2640	2605	2794

Table 6.1: Movie review dataset (Joshi et al., 2010).

In this chapter, we show how a relatively simple convolutional neural network architecture can


Austin Chronicle:

Back in 2001, when we first met renegade street racer Dominic Toretto (Diesel), he was jacking DVD players from big rigs. My, how the times have changed …

Entertainment Weekly:

Recession has hit the world of B movies: There's no longer a budget for definite articles in titles. Fast &amp; Furious , …

Meta data:

genre: Action; actors:Vin Diesel …; rating: PG-13; summer: False; ...

Weekend Box Office:

$70,950,500

Figure 6.1: Example critic reviews, meta data and target box office revenue for the movie Fast & Furious 4.

be extended to a document level task, such as text-based regression, instead of sentence level tasks often seen in previous work. Our dataset shown in Table 6.1 consists of long documents, rather than short sentences. A document instance can be seen in Figure 6.1. In addition, we make use of deep representations, which gives us further improvements over shallow bag-of-words representations. We incorporate domain information using multi task learning at the convolution level. In the end, we propose an intuitive way to interpret the network, which gives comparable level of interpretability as that of the linear model. We also show a relative improvement of over 40% prediction accuracy over the previous state of the art linear model, without the need of time consuming preprocessing of text and manual feature engineering. We demonstrate how this network with millions of parameters can be trained using only hundreds of training datapoints. In our model, the only regularization applied is early stopping despite the large number of parameters and we find that the algorithm converges very fast and starts overfitting very slowly.

In the following, we show how we can apply representation learning of movie reviews and meta data to the regression problem of box office revenue prediction.

6.1.2 Model

Unstructured Data

The outline of the convolutional network is shown in Figure 6.2. We have n training examples of the form $\{\mathbf{b}_i, \mathbf{r}_i, y_i\}_{i=1}^{n}$, where $\mathbf{b}_i$ is the meta data associated with movie i, y_i is the target gross weekend revenue, and $\mathbf{r}_i$ is a collection of u_i number of reviews, $\mathbf{r}_i = \{\mathbf{x}_j, t_j\}_{j=1}^{u_i}$ where each

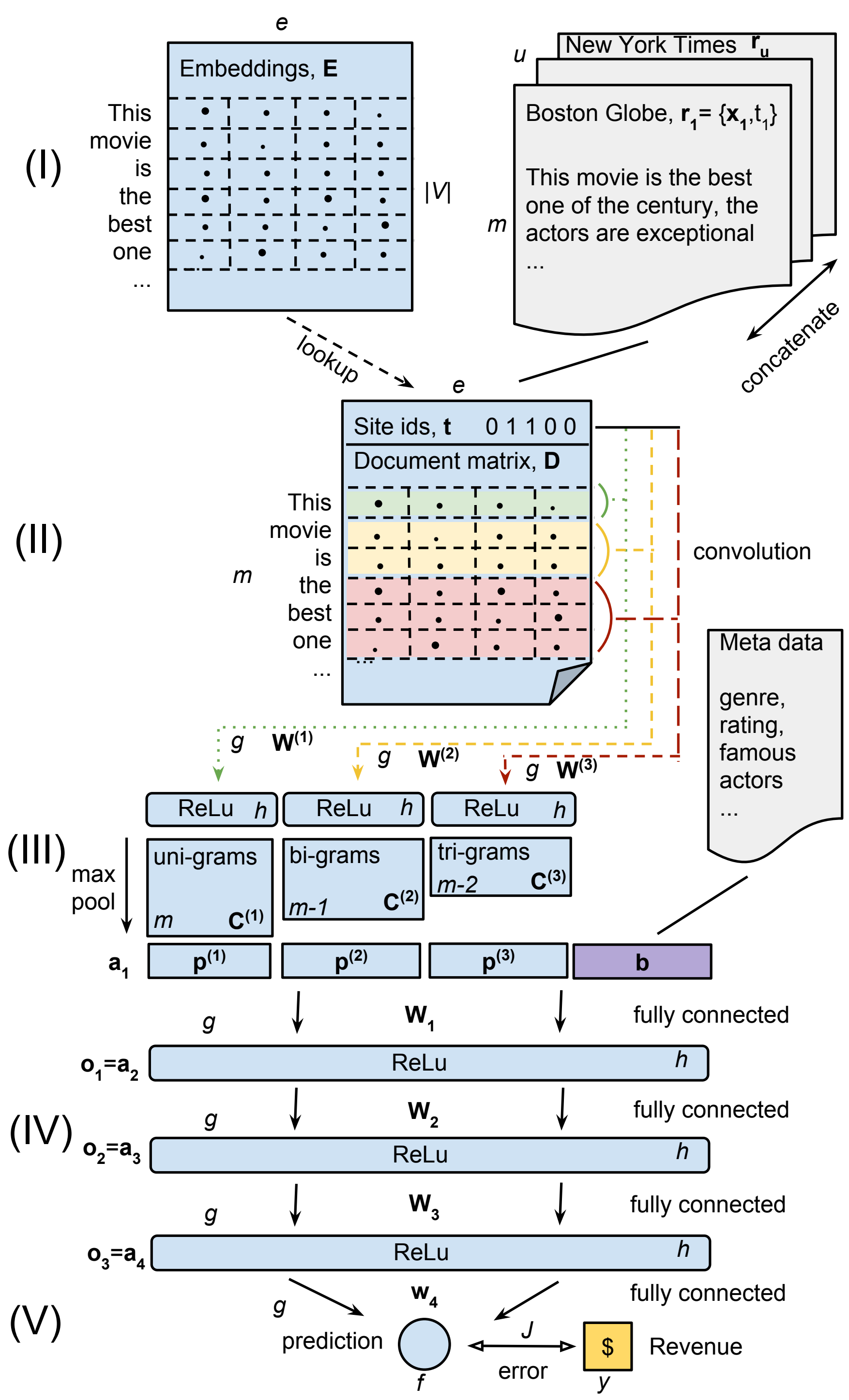


Figure 6.2: Outline of the network architecture.

review has review text $\mathbf{x}_j$ and a site id t_j. We concatenate all the review texts $d_i = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_u)$ to form our text input (see part **I** of Figure 6.2 and Figure 6.3). To acquire a distributed representation of the text, we look up the input tokens in a pretrained word embedding matrix **E** with size $|V| \times e$, where $|V|$ is the size of our vocabulary and e is the embedding dimensionality. This gives us a dense document matrix $D_{i,j} = E_{d_i,j}$ with dimensions $m \times e$ where m is the number

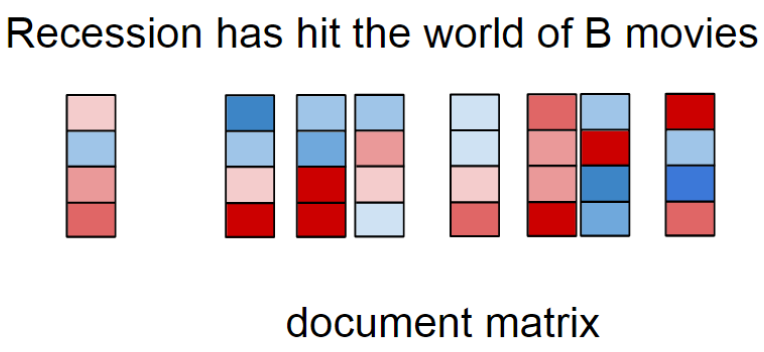


Figure 6.3: Document matrix representation with colors denoting word embedding values. Red means high, blue means low.

of tokens in the document.

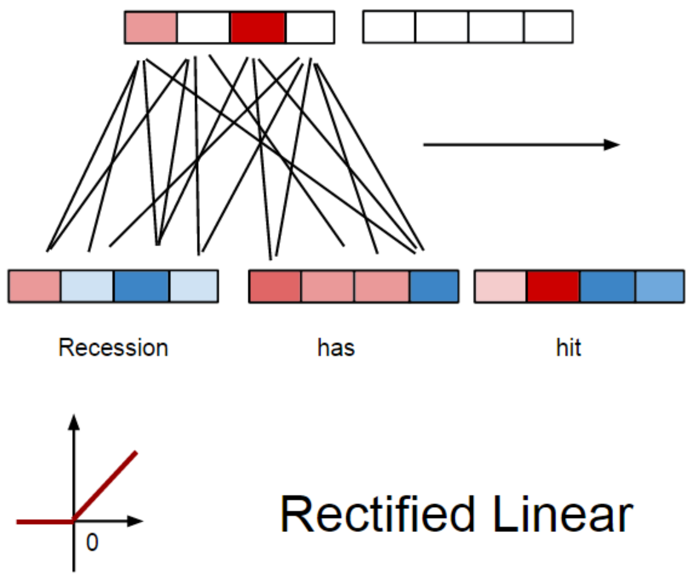


Figure 6.4: Bi-gram convolution window shifted across the text with rectifier activations applied to the output.

Since the length of text documents vary, in part **II** and Figure 6.4 we apply convolutions with width one, two and three over the document matrix to obtain a fixed length representation of the text. The n-gram convolutions help identify local context and map that to a new higher level feature space. For each feature map, the convolution takes adjacent word embeddings and performs a feed forward computation with shared weights over the convolution window. For a convolution with width $1 \leq q \leq m$ this is

$$\mathbf{S}_{i,\cdot}^{(q)} = (\mathbf{D}_{i,\cdot}, \mathbf{D}_{i+1,\cdot}, ..., \mathbf{D}_{i+q-1,\cdot})$$
$$\mathbf{C}_{i,\cdot}^{(q)} = \langle \mathbf{S}_{i,\cdot}^{(q)}, \mathbf{W}^{(q)} \rangle$$

where $\mathbf{S}^{(q)}_{i,\cdot}$ is q adjacent word embeddings concatenated, and $\mathbf{C}^{(q)}$ is the convolution output matrix with $(m - q + 1)$ rows after a linear transformation with weights $\mathbf{W}^{(q)}$. To allow for a non-linear transformation, we make use of rectified linear activation units, $\mathbf{H}^{(q)} = \max(\mathbf{C}^{(q)}, 0)$, which are universal function approximators. Finally, to compress the representation of text to a fixed dimensional vector while ensuring that important information is preserved and propagated throughout the network, we apply max pooling over time, i.e. the sequence of words, for each dimension, as shown in part **III** and Figure 6.5,

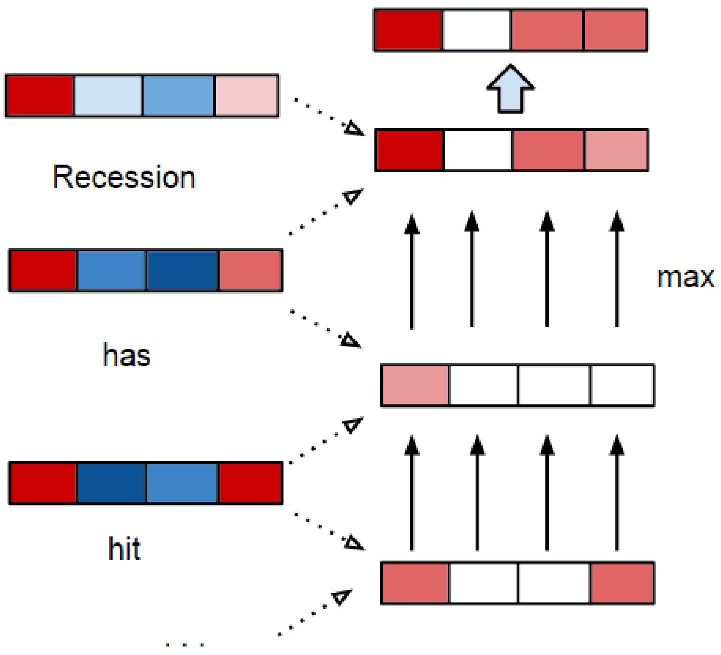


Figure 6.5: Max pooling (solid arrow) on bi-gram convolution (dotted arrow) output to collapse the representation to fixed vector (light blue arrow).

$$p^{(q)}_j = \max \mathbf{H}^{(q)}_{\cdot,j}$$

where $p^{(q)}_j$ is dimension j of the pooling layer for convolution q, and $\mathbf{p}$ is the concatenation of all pooling layers, $\mathbf{p} = (\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, ..., \mathbf{p}^{(q)})$.

Next, we perform a series of non-linear transformations to the document vector in order to progressively acquire higher level representations of the text and approximate a linear relationship in the final output prediction layer. Applying multiple hidden layers in succession can require exponentially less data than mapping through a single hidden layer (Bengio, 2009). Therefore, in part **IV** and Figure 6.6, we apply densely connected neural net layers of the form

$\mathbf{o}_k = h(g(\mathbf{a}_k, \mathbf{W}_k))$ where $\mathbf{a}_k$ is the input and $\mathbf{o}_k = \mathbf{a}_{k+1}$ is the output vector for layer k, g is

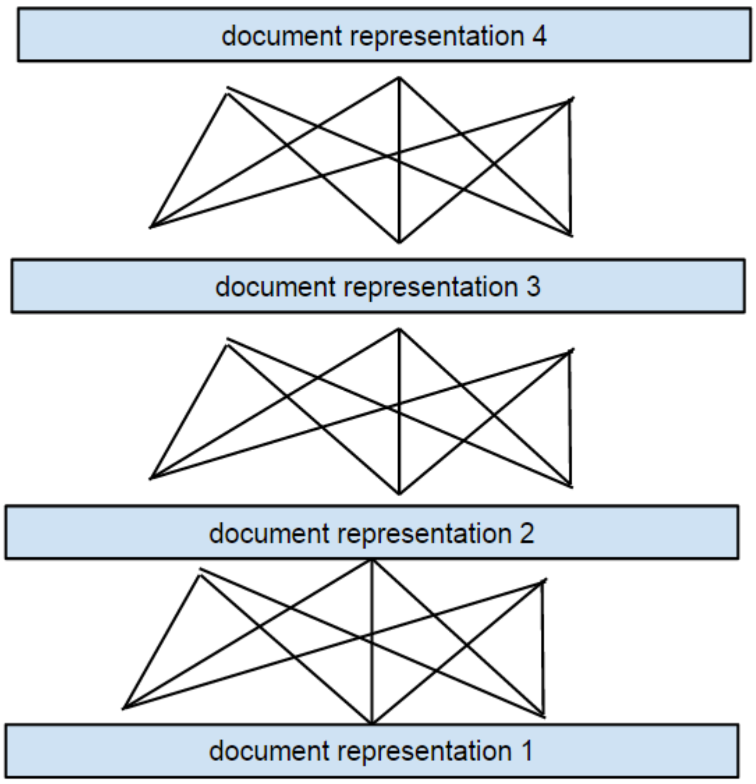

document representation 4
document representation 3
document representation 2
document representation 1

Figure 6.6: Densely connected layers allow for the learning of progressively higher level concepts.

a linear transformation function $\langle \mathbf{a}_k, \mathbf{W}_k \rangle$, and h is the activation function, i.e. rectified linear seen above. $l = 3$ hidden layers are applied before the final regression layer to produce the output $f = g(\mathbf{o}_l, \mathbf{w}_{l+1})$ in part **V** and Figure 6.7. The mean absolute error is measured between

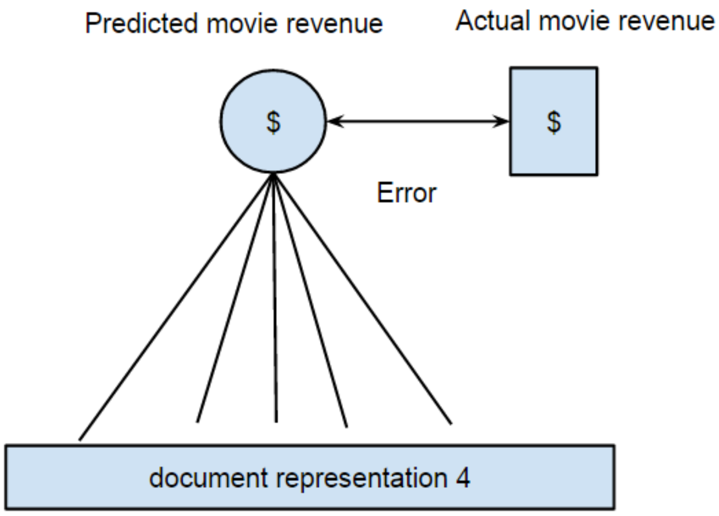


Figure 6.7: Output revenue prediction and error back propagation.

the predictions **f** and the targets **y**, which is more permissible to outliers than the squared error. The cost J is defined as

$$J = \frac{1}{n}\sum_{v=1}^{n} |f_v - y_v|.$$

The network is trained with stochastic gradient descent and the Ada Delta (Zeiler, 2012) update rule using random restarts. Stochastic gradient descent gives a noisier local estimate of the gradient than batch training, but it can start converging much faster. Ada Delta keeps an exponentially decaying history of gradients and updates in order to adapt the learning rate for each parameter, which partially smooths out the training noise. Regularisation and hyperparmeter selection are performed by early stopping on the development set. The size of the vocabulary is 90K words. Note that 10% of our lexicon is not found in the embeddings pretrained on Google News (Mikolov et al., 2013b). Those terms are initialised with random small weights. The model has around 4 million weights plus 27 million tunable word embedding parameters.

Structured Data

Besides text, injecting meta data and domain information into the model likely provides additional predictive power. Combining text with structured data early in the network fosters joint non-linear interaction in subsequent hidden layers. Hence, if meta data $\mathbf{b}$ is present, we concatenate that with the max pooling layer $\mathbf{a}_1 = (\mathbf{p}, \mathbf{b})$ in part **III** and Figure 6.8. Domain

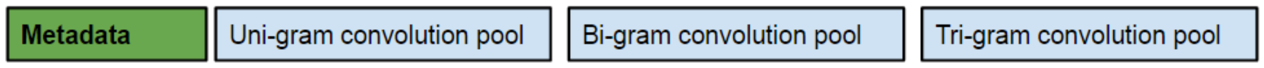


Figure 6.8: Meta data (green) appended to text representations (blue) with different convolution windows to derive the document representation.

specific information $\mathbf{t}$ is appended to each n-gram convolution input $(\mathbf{S}^{(q)}_{i,\cdot}, \mathbf{t})$ in part **II** and Figure 6.9, where $\mathbf{t}_z = \mathbf{1}_z$ indicates whether domain z has reviewed the movie. Alternatively, site information can be encoded with one-hot categorical variables. This helps the network bias the convolutions, and thus change which features get propagated in the pooling layer.

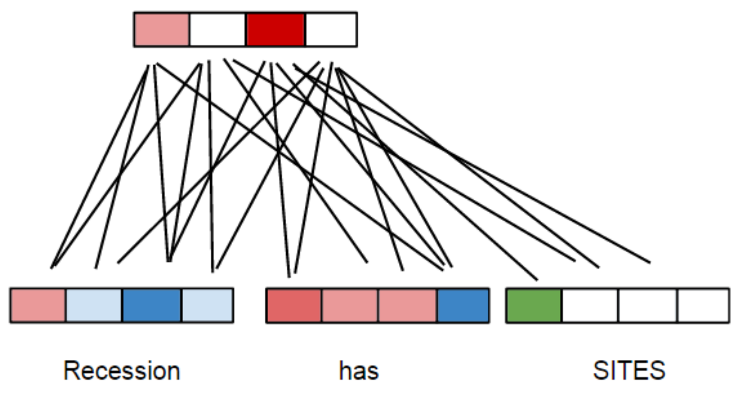


Figure 6.9: Injecting domain information as site identity (green) into bi-gram convolutions.

Extensions

Next, several extensions of the model are investigated. We have explored multi task learning for domain specific characteristic of the data. There are several options here, for example, the convolutions can be made with different weights for each task (Ghosn and Bengio, 1997), or parts of the words embeddings can be task specific while others shared (Collobert and Weston, 2008). Alternatively, the task ids can be conjoined to the convolutions either by dummy variable coding, or by a bag of words representation shown above. The convolutions can be made more complex using a hierarchical approach: the document is broken into several subsections, i.e. by reviews, over which multiple layers of convolutions are performed. K-max pooling (Kalchbrenner et al., 2014) can also increase the size of the pooling output and preserve some order information. Syntactic information can be injected by performing recursive computation over parse trees (Socher et al., 2013). In addition, dependency relation trees can be created for sentences, and convolutions can traverse this tree structure from top to bottom over n-grams. The type of dependency can be added via additional embeddings. Apart from the task bag of words extension, these experiments are not shown in Table 6.2 but some of the results for these experiments are displayed in Table 6.4.

Model Description	**MAE**($M)
Baseline mean	11.7
Linear Text	8.0
Linear Text+Domain+POS	7.4
Linear Meta	6.0
Linear Text+Meta	5.9
Linear Text+Meta+Domain+Deps	5.7
ANN Text	6.3
ANN Text+Domain	6.0
ANN Meta	3.9
ANN Text+Meta	**3.4**
ANN Text+Meta+Domain	**3.4**

Table 6.2: Mean absolute error on test set of various models. Linear models by (Joshi et al., 2010).

6.1.3 Results

Prediction Accuracy

The results in Table 6.2 show that the neural network performs very well, with around 40% improvement over the previous best results (Joshi et al., 2010). Our dataset splits are identical, and we have accurately reproduced the results of their linear model. Non-linearities are clearly helpful as evidenced by the ANN Text model beating the bag of words Linear Text model with a mean absolute test error (MAE) of 6.0 vs 8.0. Moreover, simply using structured data in the ANN Meta beats all the Linear models by a sizeable margin. Further improvements are realised through the inclusion of text, giving the lowest error of 3.4. Note that (Joshi et al., 2010) preprocessed the text by stemming, downcasing, and discarding feature instances that occurred in fewer than five reviews. In contrast, we did not perform any processing of the text or feature engineering, apart from tokenization, instead learning this automatically. Although we do make use of pretrained word embeddings in our text features.

We find that both text and meta data contain complementary signals with some information overlap between them. This confirms the finding of (Bitvai and Cohn, 2015c) on another text regression problem. The meta features alone almost achieve the best results whereas text alone performs worse but still well above the baseline. For the combined model, the performance improves slightly. This is confirmed by the analysis of the text model in the next section, which shows that high influence n-grams identified by the model many times correspond to structured data features. In Table 6.3 we can see that contrary to expectations, fine tuning the word embeddings does not help significantly compared to keeping them fixed. Moreover, randomly initialising the embeddings and fixing them performs quite well. Fine tuning may be a challenging optimisation due to the high dimensional embedding space and the loss of monolingual information. This is further exacerbated due to the limited supervision signal coming from few training data points.

One of the main sources of improvement appears to come from the non-linearity applied to the neural network activations. To test this, we try using linear activation units in parts **II** and **IV** of the network. Composition of linear functions yields a linear function, and therefore we recover the linear model results. This is much worse than the model with non-linear ReLU activations. Changing the network depth, we find that the model performs much better with a single hidden layer than without any, while three hidden layers are optimal. For the weight dimensions we find square 1058 dimensional weights to perform the best. The ideal number of convolutions are three with uni, bi and trigrams, but unigrams alone perform only slightly worse, while taking a larger n-gram window $n > 3$ does not help. Average and sum pooling perform comparatively well, while max pooling achieves the best result. Note that sum pooling recovers a non-linear bag-of-words model. A non-linear sum pooling bag-of-words model with unigrams achieves 3.6 MAE, showing that the main improvement of our approach is the non-linearity. We note that in the ANN Text+Meta model sum pooling was more unstable, and needed occasional restarts. This could be due to initialization issues arising from interacting with the meta data. Note, a max pool operator is scale invariant, whereas sum pool may need downweighing when combined with other data sources in order to not hijack the optimisation procedure due to the high magnitude feature values. Standardisation of the input and careful weight initialization may help here.

With respect to activation functions, both ReLU and sigmoid work well, and we find the sigmoid to be compatible with both min and max pooling. This is likely because of the symmetrical nature of sigmoid. However, we have encountered performance drop with tanh (5.1) and 3-way maxout units (4.8). These issues may be related to optimization problems. Replacing the convolutions with a single layer of n-gram counts results in a moderate performance drop. We have found that it was important to ensure that each layer had enough capacity for the feature transformation. For this effect, the 300 dimensional embedding space was the highest

Model Description	**MAE**($M)
fixed word2vec embeddings	**3.4***
tuned word2vec embeddings	3.6
fixed random embeddings	3.6
tuned random embeddings	3.8
uni-grams	3.6
uni+bi-grams	3.5
uni+bi+tri-grams	**3.4***
uni+bi+tri+four-grams	3.6
0 hidden layer	6.3
1 hidden layer	3.9
2 hidden layers	3.5
3 hidden layers	**3.4***
4 hidden layers	3.6
linear	5.9
non-linear	**3.4***
maxout-3	4.8
relu	**3.4***
sigm	3.5
tanh	5.1
max	**3.4***
min	3.5
avg	3.5
sum	3.6

Table 6.3: Various alternative configurations, based on the ANN Text+Meta model. The asterisk (∗) denotes the settings in the ANN Text+Meta model.

performing, and when we decreased the number of embedding dimensions the performance dropped. This suggests that it is vital to allow for a relatively large dimensional embedding space, as each dimension likely encodes a different aspect of the word, where similarities along different axes can be exploited to produce predictions. Likewise, for the forward propagation the best weight matrix dimensions were square 1058 x 1058, where the output of each convolution was a 300 dimensional vector. Whenever we increased or decreased the weight dimensions, the performance dropped. This architecture broadly corresponds to an optimal network structure automatically discovered via Bayesian hyperparameter optimisation for a prediction problem shown in (Snoek et al., 2015).

For the meta features, the ideal place to add them was after the pooling layer, where adding extra hidden layers further helped. However, after three hidden layers no improvement was observed. It seems the most important parameters of the network were the number of hidden layers, the number of hidden units, the optimization parameters, type of pooling and activations, etc. This suggests that one of the critical challenges with neural nets is to find good values for these hyperparameters and heuristics should be avoided as much possible. Therefore, one possible future extension of this work is to look at Bayesian optimization to select all the hyperparameters of the model (Snoek et al., 2012).

Model Extensions

Model Description	**MAE**($M)
multi task bag of words	3.7
multi task dummy coding	3.8
multi task sep conv weights	3.7
multi task private embeddings	3.7
hierarchical spatial convolutions	3.8
double convolution with k-max pooling	3.8
Dependency relations with type embeddings	3.9
Dependency relations	3.6

Table 6.4: Extensions of the best ANN model.

In Table 6.4 some of the extension results are presented. Even though (Kim, 2014) and (Kalchbrenner et al., 2014) applied convolution windows up to 7-grams, such a big window did not increase performance in our experiments. Furthermore, (Kalchbrenner et al., 2014) performed single dimensional convolution over the word embedding, and enabled interaction through “folding” of neighbouring dimensions. On the other hand, we convolved the entire word embedding space, which fostered interactions between various aspects of words early in the model.

Multi task learning with task identifiers, ANN Text+Domain, does improve the ANN Text model. This suggests that the tendency by certain sites to review specific movies is in itself indicative of the revenue. However this improvement is more difficult to discern with the ANN Text+Meta+Domain model, possibly due to redundancy with the meta data. Other forms of extensions described in section 6.1.2 did not result in significant gains.

The best pooling was max, which beat average, sum, min (with sigm) and k-max pooling. This suggests that what is important is whether a phrase is mentioned and not how many times.

Likewise, the word order information captured by K-max was not deemed important in the signal.

An alternative approach for multi-task learning is to have a separate convolutional weight matrix for each review site, which can learn site specific characteristics of the text. This can also be achieved with site specific word embedding dimensions. However neither of these methods resulted in performance improvements. In addition, we experimented with applying a hierarchical convolution over reviews in two steps with k-max pooling (Kalchbrenner et al., 2014), as well as parsing sentences recursively (Socher et al., 2013), but did not observe improvements.

Alternative approaches included augmenting the word embeddings with review site private dimensions, or encoding the site ids as dummy variables in the convolution. In addition, we have tried multi task learning by doubling the size of word embedding dimensions where one half of the embedding is shared between all tasks, while the other half is task specific. If the dimension of the task specific embeddings was less than the original embedding dimension, we randomly sampled the columns from the original embedding space for each task. Our task identifier was the review site. With these models, we obtained the previously best result, but no statistically significant improvement over it. To take further syntactic information into account, we replaced n-grams with uni, bi and trigrams generated by traversing the dependency tree of sentences from top to bottom. In addition, we could also inject the type of the dependency as additional embeddings into the model. These did not result in significant improvements either.

For optimisation, both Ada Grad and Steepest Gradient Descent had occasional problems with local minima, which Ada Delta was able to escape more often. In contrast to earlier work (Kim, 2014), applying dropout or max constraints on the final layer did not improve the validation error. The optimiser mostly found good parameters after around 40 epochs which took around 30 minutes on a NVidia Kepler Tesla K40m GPU. Due to the fact that random restarts were necessary, the stochastic nature of the optimization and that crossvalidation performance was only measured once every epoch, therefore a ± 0.2 error should be applied to the results presented in the tables for any given optimization run. Note that the only regularization employed was early stopping, despite the large complexity of the model, and we achieved fast improvements in around 12 epochs with slower improvements occurring until around epoch 40. After that overfitting did tend to happen but at a slow speed.

It is important to note that in (Joshi et al., 2010), site concatenated features in the text model resulted in substantial gains in performance, whereas in our case multi task learning by site did not result in big improvements. One possible explanation is that in our case the neural network model is much more powerful than a linear bag of words, therefore it managed to implicitly learn site specific information from training examples by simply building a better model of language and document. For example, if certain sites write in a certain way, then by thoroughly analyzing the language, the site identity is recoverable. That said, the joint model is powerful enough to model the distribution of tasks given the training dataset.

Phrase Impacts

Next we perform analysis to determine which words and phrases influenced the output the most in the ANN Text model. To do so, we set each phrase input to zeros in turn and measure the prediction difference for each movie across the test set. We report the min/max/average/count values in Table 6.5 and Figure 6.11. We isolate the effect of each n-gram by making sure the uni, bi and trigrams are independent as seen in Figure 6.10, i.e. we process “Hong Kong” without zeroing “Hong” or “Kong”. About 95% of phrases result in no output change, including common sentiment words, which shows that text regression is a different problem to sentiment analysis.

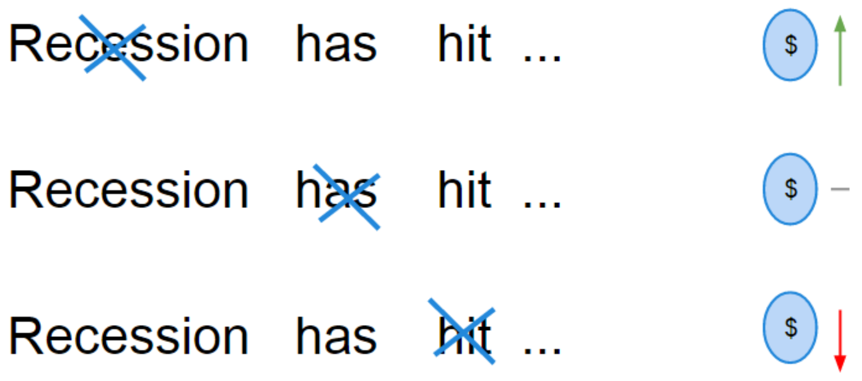


Figure 6.10: Measuring phrase impact for unigrams by blanking out each phrase’s embedding and observing the change in prediction. This process is repeated for bi- and tri-grams too treating each n-gram isolated.

We see that words related to series “# 2”, effects, awards “praise”, positive sentiment “intense”, locations, references “Batman”, body parts “chest”, and others such as “plot twist”, “evil”, and “cameo” result in increased revenue by up to $5 million. On the other hand, words related to independent films “vérité”, documentaries “the period”, foreign film “English subtitles” and negative sentiment decrease revenue. Note that the model has identified structured data in the unstructured text, such as related to revenue of prequels “39 million”, crew members, duration “15 minutes in”, genre “[sci] fi”, ratings, sexuality, profanity, release periods “late 2008 release”, availability “In selected theaters” and themes. Phrases can be composed, such as “action unfolds” and “action and horror” amplify “action”, and “cautioned” is amplified by “strongly cautioned”. “functional” is neutral, but “functional at best” is strongly negative. “Toronto Film Festival” is less negative than ‘Festival”. Some words exhibit both positive and negative impacts depending on the context. This highlights the limitation of a linear model which is unable to discover these non-linear relationships. “13 - year [old]” is positive in New in Town, a romantic comedy and negative in Zombieland, a horror. Other phrases that change sign include ”Hong Kong” which is positive in the movie Inglourious Basterds, but negative in Boondock Saints 2. ”English” is usually negative but within ”English subtitles” it is positive, even though that bigram in itself is negative. The character strings “k /” (mannerism of reviewer), “they’re” (unique apostrophe), “&#39” (encoding error) are high impact and unique to specific review sites, showing that the model indirectly uncovers domain information. This can explain the limited gain that can be achieved via multi task learning. The “k /” phrase is only used by a specific reviewer from Village Voice, which means the model mostly ties the meaning of the phrase to the domain. Note, in this experiment domain information was not explicitly input into the network.

We have further filtered the phrases such that they occur with positive, negative and neutral impact on the revenue predictions for various movies, and ordered them by the magnitude of their influence. While the vast majority of phrases are neutral, many that do have an impact which can assume both positive and negative one. This again emphasizes the limitation of a linear model which is unable to discover these non-linear relationships depending on context. One example is the word “action” which generally refers to popular action movies and has a strong positive impact. However, in the sexual context of girl-girl action, it comes with a strong negative impact. This is probably because the phrase suggests adult material, which is age restricted, and thus decreases the available audience of the movie, ultimately resulting in lower box office revenue. Another example is the unigram “design” which occurs with a positive impact in the context of “evil designs on the Vatican” relating to the successful Da Vinci Code franchise, but also occurs with negative impact in the context of “So much for Death’s grand

Top 5 positive phrases	min	max	avg	#
sequel	20	4400	2300	28
flick	0	3700	1600	22
k /	1500	3600	2200	3
product	10	3400	1800	27
predecessor	22	3400	1400	13
Top 5 negative phrases	**min**	**max**	**avg**	**#**
Mildly raunchy lang.	-3100	-3100	-3100	1
(Under 17	-2500	1	-570	75
Lars von	-2400	-900	-1500	3
talk the language	-2200	-2200	-2200	1
. their English	-2200	-2200	-2200	1
Selected phrases	**min**	**max**	**avg**	**#**
CGI	145	3000	1700	28
action	-7	1500	700	105
summer	3	1200	560	42
they're	3	1300	530	68
1950s	10	1600	500	17
hit	8	950	440	72
fi	-15	340	160	26
Cage	7	95	45	28
Hong Kong	-440	40	-85	11
requires acc. parent	-780	1	-180	77
English	-850	6	-180	41
Sundance Film Festival	-790	3	-180	10
written and directed	-750	-3	-220	19
independent	-990	-2	-320	12
some strong language	-1600	6	-520	13

Table 6.5: Selected phrase impacts on the predictions in $ USD(K) in the test set, showing min, max and avg change in prediction value and number of occurrences (denoted #). Periods denote abbreviations (language, accompanying).

designs", which is a disappointing phrase expressing negative emotion. It has neutral impact in "character designs", which conveys us little information about where the author stands in relation to the movie. Last, the word "English" generally has a big negative impact, such as in "Can you Americans speak any other language except English?" – slightly condescending; but in the context of "English subtitles" it has a large positive impact, even though the bigram "English subtitles" is negative. This is related to the fact that the movie comes with subtitles, which is a characteristic of independent movies. These movies are unpopular and often appeal to a niche audience, thus this fact substantially decreases the expected revenue of the movie. However, if the subtitles are in English rather than in some other language, that fact increases the revenue slightly, most likely because English is a widely spoken language and thus more people are able to understand the movie, as a result of which more people ultimately see it in the cinema. Other examples include "Hong Kong", which results in +$40K in Inglourious Basterds, but -$140K in Boondock Saints 2, showing a wide range of its impact.

phrase		min	max	avg
sequel	series	20	4400	2300
CGI	effect	145	3000	1700
flick	genre	0	3700	1600
action	genre	-7	1500	700
summer	release	3	1200	560
they're	domain	3	1300	530
1950s	theme	10	1600	500
hit	sentiment	8	950	440
fi	genre	-15	340	160
Cage	actor	7	95	45
Hong Kong	location	-440	40	-85
requires accompanying parent		-780	1	-180
English	foreign	-850	6	-180
Sundance Film Festival	indie	-790	3	-180
written and directed	indie	-750	-3	-220
independent	indie	-990	-2	-320
some strong language	rating	-1600	6	-520
(Under 17	rating	-2500	1	-570

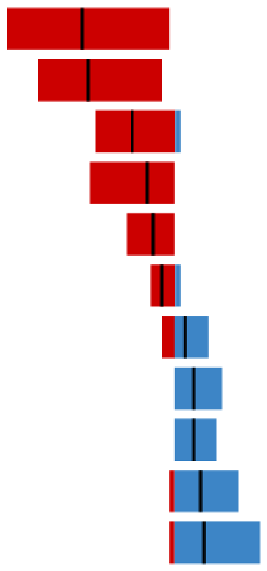


Figure 6.11: Phrase impact visualization (red: positive impact, blue: negative impact, line: mean impact, width: spread of impact across different movie instances). Red text shows manually annotated category of the phrase.

Expanding on further examples, we can see that words related to series such as "franchise", "2", "sequel"; effects such as "sound effects", "computer generated", "CGI"; awards such as "won", "international acclaim"; positive sentiment words such as "waving gang", "intense", "general beauty", "comic sidekick", "trademark", certain geographic locations "Eastern European", "Hollywood", cultural icons such as "Batman", result in increased revenue, sometimes by around even $2 million dollars. On the other hand, words related to independent films i.e "Festival", "English subtitles", "sociological vérité", "infectiously frisky documentary", "Reviewed at Venice", and negative sentiment words or words referring to inauthenticity such as "barbed portrait", "terrible filmmaker", "harrowing pure cinema", "functional at best", "compellingly staged", "accurately reconstruct" greatly decrease revenue. An interesting note is that the model has been able to identify structured data in the unstructured text, such as the phrases "39 million", "750 million". The latter refers to the gross revenue of the related Da Vinchi Code movie in the review text for Angels and Demons, and has a large positive impact on the prediction. Likewise, certain cast members such as “Lemhenyi ; music”, release periods such as “summer”, “weekend”, the duration such as “84 minutes”, genre such as “action - comedy”, “flick” have greatly positive impacts; while adult ratings such as “rating : Unrated”, “requires accompanying parent”, “under 17”, and other known people such as “Andy Warhol”, “Noureen DeWulf” along with the words “dedicated to sexual”, “their private affairs”, “profanity”, as well as delayed release such as “late 2008 release” have negative impacts. Since the training was performed only on text here, this suggests the model can learn the meta data from the review text without that being fed into it explicitly in a structured format. Another interesting aspect is the composition of phrases, for example “Festival” has a negative impact of -\$15962.1, but “Film Festival”, and “Toronto Film Festival” only have impacts of -\$7347.2 and -\$4808.36. This suggests the unpopularity of movie festivals often showcasing independent movies among the population. Since these movies do not attempt to appeal to mass audience at all, a decreased box office revenue is expected. Certain themes are more popular than others, for example “1993”, “90’s” have negative impact but “1950s” is greatly positive. In addition, famous crew members normally have positive impact in the movie, but in the case of the movie Fanboys, “Han Solo” has a negative impact due to the negative ridiculing context in which the movie depicts the Star Wars franchise.

Model Visualization

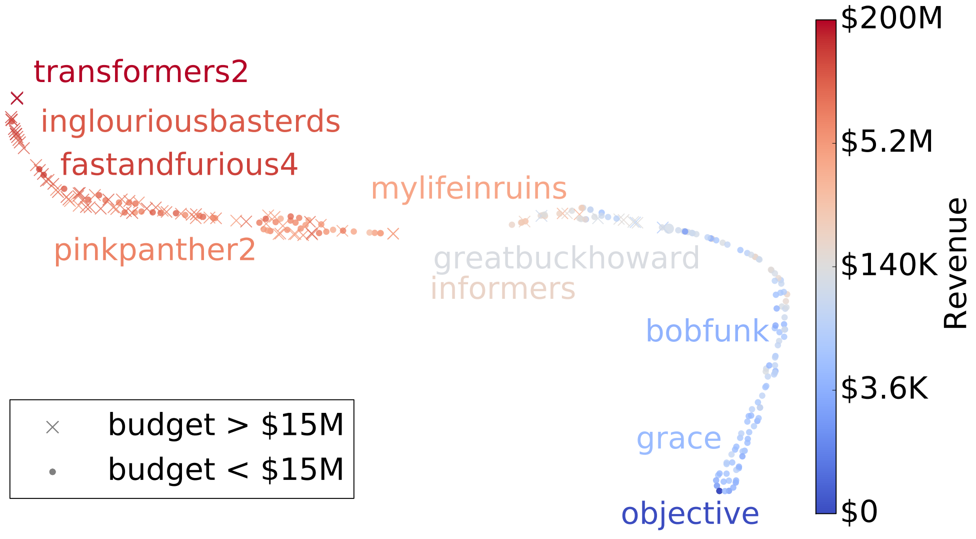


Figure 6.12: Two dimensional projection of the last hidden layer of test movies using t-SNE. Red means high and blue means low revenue. The cross vs dot symbols indicate a production budget above or below \$15M.

Last, we have plotted the last hidden layer of each test set movie with t-SNE (Van der Maaten and Hinton, 2008). This gives a high level representation of a movie. In Figure 6.12 it is visible that the test set movies can be discriminated into high and low revenue groups and this also correlates closely with their production budget. We have color coded the labels to denote the revenue made by the movie, where red means high revenue and blue low revenue. Before plotting, the revenue has been log scaled and normalized to between 0 and 1. Note that the model was not trained on these movie reviews and the revenue label information was not available to the clustering algorithm. This suggests that indeed it is possible to tell from movie reviews whether a movie is going to make it big in the box office, and our model has successfully identified the critical properties of text to predict the revenue.

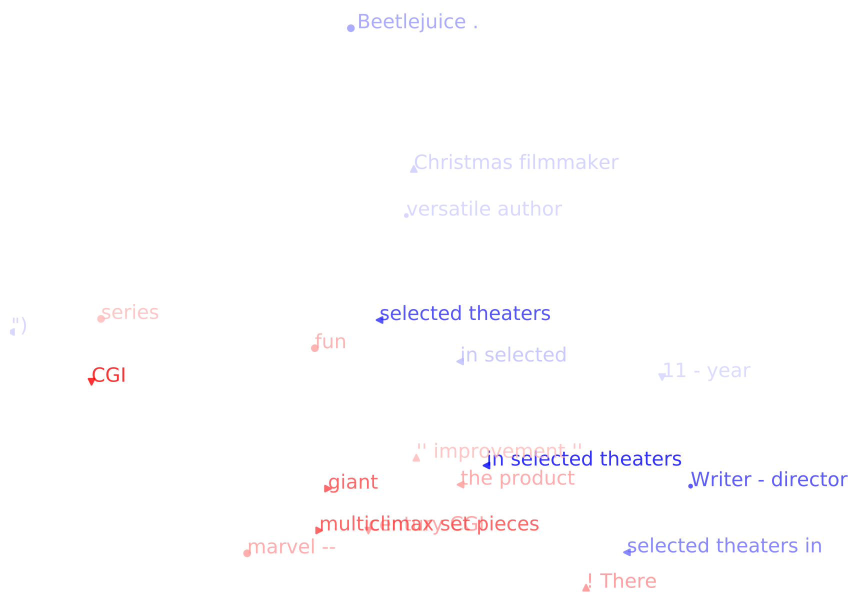


Figure 6.13: Top positive and negative phrases for test movie ”Coraline”. Opacity denotes strength of impact. Marker type shows identity of review site.

Next we look at an individual movie. In Figure 6.13, we plotted the top 10 positive and negative phrases that influence the movie Coraline revenue the most. A test example was randomly selected and for each uni, bi and trigram the effect of removing that phrase was measured in terms of prediction difference, as explained before. After filtering all phrases with no effect since our model is sparse, we reduced the dimensionality of the remaining activations of the phrase and site convolutions and plotted them with T-SNE. In the figure, the marker type denotes a movie review site and the color the normalized effect on the prediction. For example, we see that the phrase “selected theaters” greatly reduced the revenue prediction, whereas “CGI”, “multiclimax”, “fun” and “series” increased it. In addition, the Boston Globe review mainly decreased the revenue whereas Village Voice and Austin Chronicle increased it.

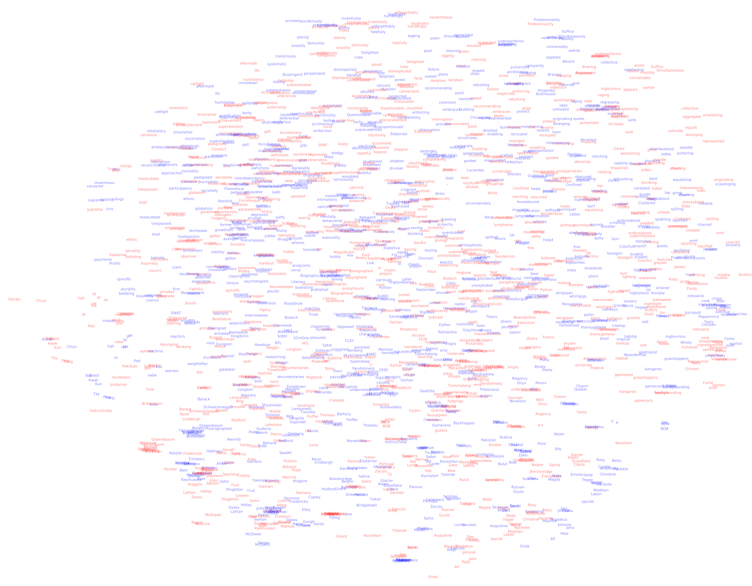


Figure 6.14: 1000 randomly selected initialized (blue) and trained (red) word embeddings, plotted with t-SNE (best viewed electronic by zooming in).

In Figure 6.14, one can see a graph of 1000 randomly selected trained word embeddings and how they have migrated from their initialized positions. The dimensionality of the embeddings was reduced with t-SNE (Van der Maaten and Hinton, 2008). We can see that while most words stayed close to their original positions, some were optimized according to their context in a movie. This corresponds with our previous finding that up to 95% of the phrases have no effect on the output. For example, ”inglourious” and ”smashfest” were originally close together, but after the optimization ”inglourious” migrated towards ”raging”, while ”smashfest” towards ”trudgers”.

6.1.4 Conclusions

In this chapter, we have shown that convolutional neural networks with deep architectures greatly outperform linear models even with very little supervision, and they can identify key textual and numerical characteristics of data with respect to predicting a real world phenomenon. In addition, we have demonstrated a way to intuitively interpret the model. In the future, we will investigate ways for automatically optimising the hyperparameters of the network (Snoek et al., 2012) and various extensions to recursive or hierarchical convolutions.

We have replicated the experiments of (Joshi et al., 2010) and demonstrated significant improvements over their results. We have tried a variety of ways to extend the model to take syntactic language information, dependency relations and task identity into account, but have found the base model to be powerful enough such that these extensions did not result in significant improvement over the previously best result. We have found AdaDelta to be an excellent optimizer for this kind of networks and that one of the key challenges is the tuning of network hyper parameters.

In the next section, we will build upon the concept of deep convolutional neural network that aggregates multi-modal data. We will start with the model presented in this chapter as basis and show how we can further incorporate time series data, measure latent temporal dynamics extracting both local and global patterns, obtain actionable output, and train the model to directly optimise for profit in a full end-to-end trading scenario over more than 500 companies across two major stock exchanges and over almost two decades of test data. We show that our trading simulator neural network achieves significant profits, beating the buy-and-hold baseline over the entire period with high significance.

6.2 An End-to-End Stock Trading Agent

In the previous section, we presented a neural network for predicting the future box of revenue of movies from their reviews and other meta data. In this section, we move on to constructing a deep end-to-end learning machine for stock trading and use the previously constructed convolutional network as basis.

We apply deep representation learning to stock market trading, a problem considered extremely difficult predicting with high levels of noise. Our end-to-end differentiable trading agent jointly models raw market data of 727 companies listed on 2 major stock exchanges; news data from 8 major news agencies and economic indicators. It outputs limit order trades, thereby fully simulating a real world trader while keeping track of its inventory and cash balance. Directly optimizing for the end value of its portfolio, the model achieves a net return of 15% p.a., significantly beating the baseline buy and hold 11% with low correlation over the test period of 18 years.

6.2.1 Introduction

Recurrent neural networks have been previously applied to discover temporal dynamics in noisy time series data (Giles et al., 2001) and deep architectures have been used to optimize for a financial criterion (Bengio, 1997b). We construct a deep neural network to automatically trade stocks and to optimize for long term profit or risk-adjusted returns. We show that it is possible to consistently outperform passive "buy-and-hold" strategies despite economic theories to the contrary.

Representation learning allows neural networks to automatically learn the best embedding for data that is the most predictive of the objective optimized (Bengio et al., 2013). They have been shown to outperform extensive manually engineered systems in computer vision for learning image filters (Krizhevsky et al., 2012); in speech processing for implicit phoneme recognition (Hinton et al., 2012); and in natural language processing for language modelling (Bengio et al., 2003). Earlier, neural networks were used in information retrieval tasks, such as text categorization, with success (Sebastiani, 2002). In technical analysis, human constructed heuristics are used to predict future returns from historical price series. Likewise in fundamental analysis, indicators are developed to gauge the health of companies and future likelihood of bankruptcy (Lo et al., 2000). For a more thorough review of financial theory, technical and fundamental analysis, please refer to chapter 2. The construction of new indicators is a time consuming, labour intensive and costly process, and in the end may not improve predictive accuracy. Instead here, we use a deep neural network to save on human costs of development and make technical and fundamental analysis redundant. We feed raw price and economic data along with the text of various news articles into our model and automatically learn embeddings that predict the future evolution of the market. Simultaneously, we also aim to improve the predictive power of our system since instead of using heuristics, our learned representations are directly optimized for our task objective. Deep architectures can also uncover complex non-linear relationships in the data, and make the learning of higher level concepts more efficient (Bengio, 2009). Our model is likewise deep, enabling the interactions between all the different data sources, as well as the modulation of a time varying component to capture long or short term dependences in the markets.

In this work we take a data driven approach in learning a model of markets purely from historical data. While doing so, we strive to make as few assumptions as possible and use realistic constraints to simulate equity trading in a real world scenario. We design an end-to-end differentiable trading agent that assesses raw information, reasons over it, and executes informed actions based on the environment. This way, our algorithm does not only model the market in isolation, but also the trading behavior, thereby viewing the whole environment as an interconnected system. In addition, the constructed model directly outputs limit order trades and optimizes the economic utility of the trader, i.e. net worth. By doing so, it does not attempt to describe a complete theory of markets, which may be prohibitively complex and expensive, but rather focuses limited computational resources to explain only those aspects that aids in maximizing its cumulative reward at the final step of the simulation. In contrast, many previously developed systems devised proxy problems for easier modelling and framed their task as regression or classification. Then they used pre or post processing with human constructed heuristics to adapt the model to the task at hand ex post facto (Schöneburg, 1990). This is often seen undesirable as their is a disparity between the training objective and the evaluation metric, and many times it is not clear what heuristics to apply to the model output. We take an active approach to investing and our goal is to consistently spot stocks that outperform our baseline strategy, without taking on excess risk.

6.2.2 Model

The model aims to optimize the value of the trader's account balance at the end of the trading simulation. The input is daily series of stock prices, economic indicators and news data. This is modelled through recurrent connections over time, after which we incorporate the current inventory state (stocks and cash) before outputting buy and sell orders for predicted volume and price. The simulator then records transactions in case the orders are filled and advances the trader to the next day. A schematic outline of the network can be seen in Figure 6.15.

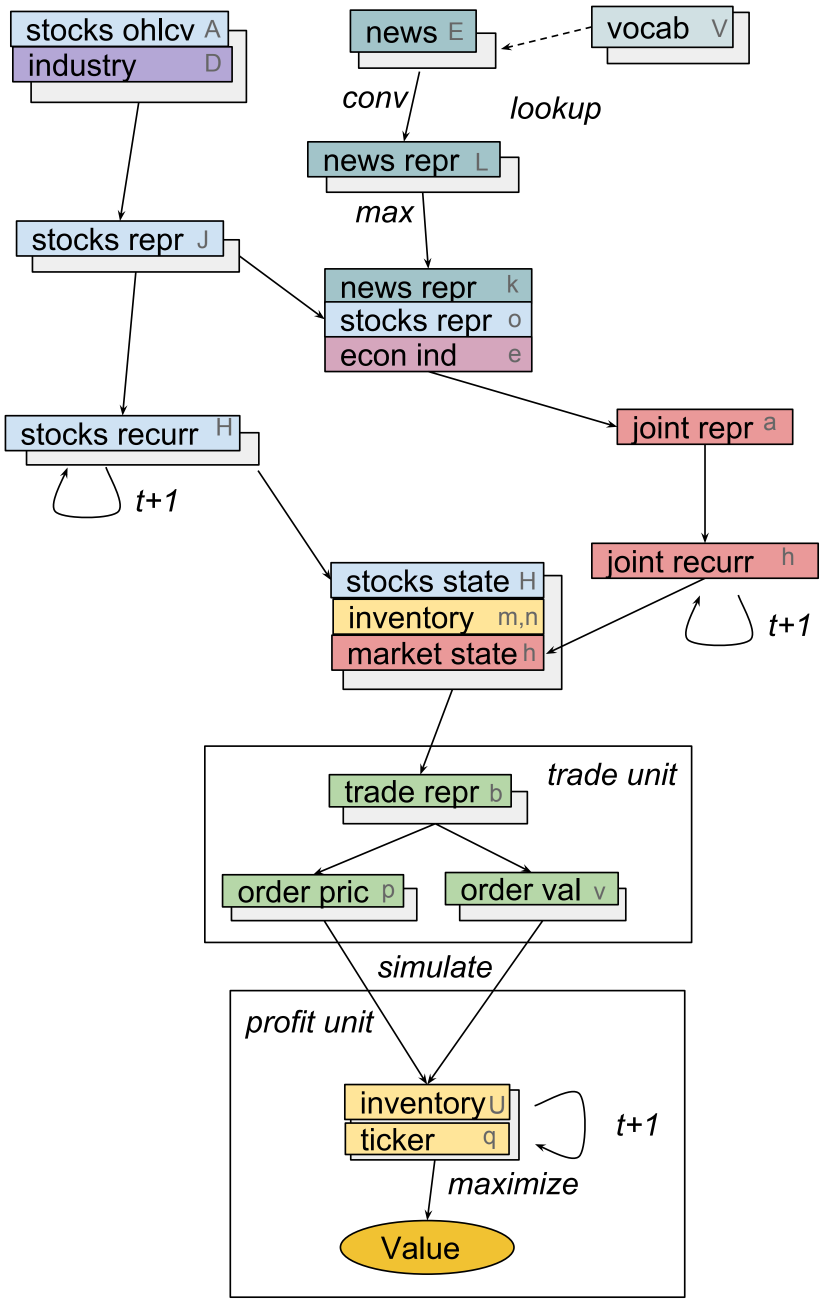


Figure 6.15: Outline of the network architecture. Shadowing denotes stacked representations. Arrows indicate fully connected layers. Variables are shown with grey letters.

Representing Financial and Text Time Series

We have daily time series data with T steps. At step t the input into the model is $\mathbf{e}_t$ a vector of economic indicators; $\mathbf{A}_{t,i}$ a matrix of stock prices containing open, high, low, close, volume, close ask, close bid, for each stock i; $\mathbf{D}_{t,i}$ a matrix of one-hot industry codes for each stock; and $\mathbf{v}_t$ a vector of word indices, where all news for the day is concatenated into one document.

We look up each word from a vocabulary, $\mathbf{V}$ matrix; and perform unigram non-linear convolution on each token in the resulting matrix $\mathbf{E}$, in order to reweigh their values and aggregate along each dimension with max pooling. This enables the propagation of few important key words throughout the rest of the network. Our activation function is the rectifier $\text{ReLu}(x) = \max(x, 0)$ that allows for faster backpropagation than sigmoid units.[1] This gives rise to the encoding of news data as

$$\begin{aligned} \mathbf{E}_{t,i} &= \mathbf{V}_{\mathbf{v}_{t,i}} \\ \mathbf{L}_t &= \text{ReLu}(\langle \mathbf{E}_t, \mathbf{W}_e \rangle) \\ \mathbf{k}_t &= \text{pool}_{\max}(\mathbf{L}_t) \end{aligned} \tag{6.1}$$

where $\text{pool}_{\max}$ is the pooling operator with the max function, i.e. $\max_i \mathbf{L}_{t,i,j}$. After obtaining the news representation, we transform the daily stock and industry data with a feed forward computation. The non-linearly transformed output matrix

$$\mathbf{J}_t = \text{ReLu}(\langle (\mathbf{A}_t, \mathbf{D}_t), \mathbf{W}_a \rangle)$$

jointly represents the stock prices combined with the company industries. To model the joint market movements and the interaction of prices with economic and news information, we flatten the price matrix $\mathbf{o}_t = \text{flat}(\mathbf{J}_t)$ and concatenate it with the news and economic vectors. We then perform a joint feed forward computation with a rectified linear unit activation, which represents the joint interaction of various market forces at time step t. This compresses all the market data into a representation

$$\mathbf{a}_t = \text{ReLu}(\langle (\mathbf{o}_t, \mathbf{k}_t, \mathbf{e}_t), \mathbf{W}_k \rangle) \tag{6.2}$$

[1]We use $(.,.)$ to denote vector or matrix concatenation and we omit bias units for clarity.

that allows for the discovery of latent correlations and relationships between various aspects of the signal.

Temporal Dynamics

Over time markets exhibit dynamic behavior which we incorporate with a **g**lobal Long Short Term Memory Unit (LSTM) (Hochreiter and Schmidhuber, 1997). The hidden unit outputs the next memory cell and hidden state $\mathbf{h}^g_t, \mathbf{c}^g_t = \mathrm{LSTM}(\mathbf{a}_t, \mathbf{h}^g_{t-1})$. The proposed memory cell $\tilde{\mathbf{c}}^g_t = \tanh(\langle(\mathbf{a}_t, \mathbf{h}^g_{t-1}), \mathbf{W}^g_c\rangle + \mathbf{b}^g_c)$ is modulated by the input, $\mathbf{i}^g_t = \sigma(\langle(\mathbf{a}_t, \mathbf{h}^g_{t-1}), \mathbf{W}^g_i\rangle + \mathbf{b}^g_i + \mathbf{c}^g_{t-1}\tilde{\mathbf{w}}^g_i)$ the forget, $\mathbf{f}^g_t = \sigma(\langle(\mathbf{a}_t, \mathbf{h}^g_{t-1}), \mathbf{W}^g_f\rangle + \mathbf{b}^g_f + \mathbf{c}^g_{t-1}\tilde{\mathbf{w}}^g_f)$ and the output $\mathbf{o}^g_t = \sigma(\langle(\mathbf{a}_t, \mathbf{h}^g_{t-1}), \mathbf{W}^g_o\rangle + \mathbf{b}^g_o + \mathbf{c}^g_{t-1}\tilde{\mathbf{w}}^g_o)$ gates. The hidden unit outputs the next memory cell $\mathbf{c}^g_t = \tilde{\mathbf{c}}^g_t\mathbf{i}^g_t + \mathbf{c}^g_{t-1}\mathbf{f}^g_t$ and hidden state $\mathbf{h}^g_t = \tanh(\mathbf{c}^g_t)\mathbf{o}^g_t$. Furthermore, we add peekback weights from each gate to the memory cell. For the full definition of LSTM, please refer to (Graves, 2013). The memory cell in the LSTM captures the global past long and short term time dependencies in the data up to the current time step t.

We explore a different way to summarize **l**ocal time dependency specific to each stock. Our hypothesis is that this way, the model has a more direct link between the prices of the stock and the trading action for that stock along these skip connections. We feed the stock price matrix $\mathbf{J}_t$ into another LSTM where the parameters are tied across all stocks, but the hidden states are independent $\mathbf{H}^l_{t,i}, \mathbf{C}^l_{t,i} = \mathrm{LSTM}(\mathbf{J}_{t,i}, \mathbf{H}^l_{t-1,i})$. This assumes that each stock is governed by the same underlying market forces but their individual histories may vary. The output hidden state $\mathbf{H}^l_t$ and memory cell $\mathbf{C}^l_t$ can be similarly derived as before, but this time the output is a matrix instead of a vector, with one row for each stock. The proposed memory cell $\tilde{\mathbf{C}}^l_t = \tanh(\langle(\mathbf{J}_t, \mathbf{H}^l_{t-1}), \mathbf{W}^l_c\rangle + \mathbf{b}^l_c)$ is filtered by the input $\mathbf{I}^l_t = \sigma(\langle(\mathbf{J}_t, \mathbf{H}^l_{t-1}), \mathbf{W}^l_i\rangle + \mathbf{b}^l_i + \mathbf{C}^l_{t-1}\tilde{\mathbf{w}}^l_i)$, the forget $\mathbf{F}^l_t = \sigma(\langle(\mathbf{J}_t, \mathbf{H}^l_{t-1}), \mathbf{W}^l_f\rangle + \mathbf{b}^l_f + \mathbf{C}^l_{t-1}\tilde{\mathbf{w}}^l_f)$ and the output $\mathbf{O}^l_t = \sigma(\langle(\mathbf{J}_t, \mathbf{H}^l_{t-1}), \mathbf{W}^l_o\rangle + \mathbf{b}^l_o + \mathbf{C}^l_{t-1}\tilde{\mathbf{w}}^l_o)$ gates, outputting the new memory cell $\mathbf{C}^l_t = \tilde{\mathbf{C}}^l_t\mathbf{I}^l_t + \mathbf{C}^l_{t-1}\mathbf{F}^l_t$ and the hidden states for each stock $\mathbf{H}^l_t = \tanh(\mathbf{C}^l_t)\mathbf{O}^l_t$.

Placing a trade also requires information about the current state of the inventory, i.e. how much cash and stocks we posses. Therefore, for each stock i that we trade, we concatenate the current cash n_t and stock $m_{t,i}$ balance to the global $\mathbf{h}^g_t$ and local $\mathbf{H}^l_{t,i}$ market representations, and make a non-linear transformation,

$$\mathbf{B}_{t,i} = \mathrm{ReLu}(\langle(\mathbf{n}_t, \mathbf{M}_{t,i}, \mathbf{h}^g_t, \mathbf{H}^l_{t,i}), \mathbf{W}_g\rangle).$$

The cash and global market representations are replicated for each stock, but the stock balance and local market representations are unique. This summarizes the current state of the market and the inventory jointly, which we call a trade representation.

Trading Output

The output of the network predicts a limit order trade for each active stock, which consists of a predicted price p and value v of the trade. Here, we allow the model to learn a different predictive weight per stock, enabling the model to adapt to the unique characteristics of each company. For a given stock i this is

$$p_{t,i}, v_{t,i} = \langle \mathbf{B}_{t,i}, \mathbf{W}_{p,i}\rangle. \tag{6.3}$$

We decide whether to sell $(-)$ or buy $(+)$ by looking at the sign of the value predictions. Figure 6.16 illustrates a trading scenario in which we want to sell our stocks at a higher price relative to the last close price and buy with our cash at a lower price. However, we must be careful to not set our price too high or low, otherwise our trade will not be fulfilled through falling outside the trading price range the next day. Likewise, we can also ensure that the volume of our trades is less than the market trading volume. However, this is in an idealized market. In reality the volume at the given price is likely to be less. Also, our trades for large markets like the US and small budgets barely have an effect. It might be more natural to predict number of stocks traded, however we predict value of trades instead in order to ensure symmetry of the output. Last, by allowing both buying and selling fractional units in the same output, we create a bottle neck and condense information in the network, thereby forcing it to generalize.

Next, we derive limit order trades from the predictions. The limit order price is the relative price movement from the last closing price of the stock, and the predicted amount is the value of the shares sold or bought at the limit order price. We pick this output formulation because in case the model has no information, a default zero prediction corresponds to an order for the current closing price of the stock and no amount to trade, both reasonable defaults. We push the price

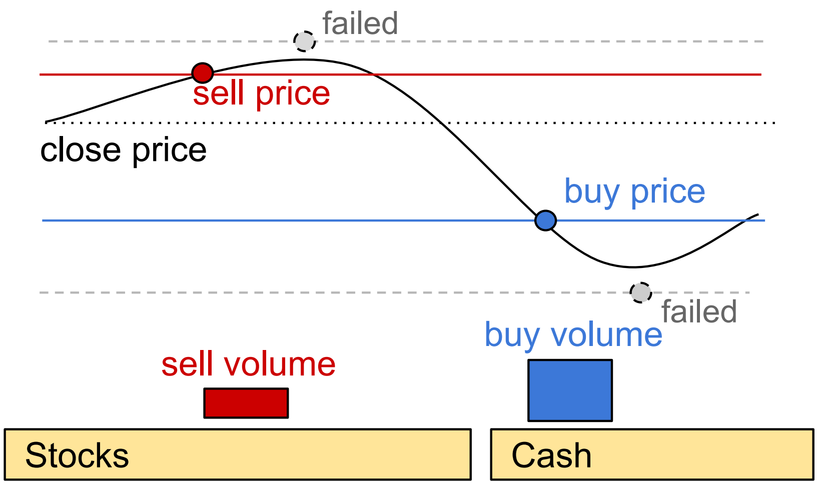


Figure 6.16: Price and volume predictions for buying with cash (blue) and selling stocks (red). Dashed line indicates failed trades (grey) due to constraint violations. Dotted line shows last closing reference price.

signal through an exponential to ensure positivity, $p^-_{t,i} = \exp(p_{t,i})q_{t,i,b}$; $p^+_{t,i} = \exp(p_{t,i})q_{t,i,a}$, and push the value signal through a rectifier to do the same, $v^-_{t,i} = \mathrm{ReLu}(-v_{t,i})$; $v^+_{t,i} = \mathrm{ReLu}(v_{t,i})$, where $q_o, q_h, q_l, q_c, q_v, q_a, q_b$ denote open, high, low, close, volume, close ask, close bid of market ticker q. We consider the closing price and spread known. However, we do not know anything about the properties, i.e. prices or volume, of the next day at the order issue time. Therefore, for a trade to be successfully executed, we must satisfy a series of constraints. We enforce soft constraints by using operators such as *rescaling* and *capping*, that is based on information that is available at the time of action. In addition, we enforce hard constraints by *failing hard* and not recording a trade, where the outcome of whether any violation has occurred is unknown at the time of action. This mimics the real life situation of a trader.

The pseudo code of the trading simulation for a single day can be seen in Algorithm 1. For selling, we cap the price at the current close bid, which has the effect of eliminating many bad trades and forcing the model to make sensible predictions, and consider the trade successfully executed if the predicted price is lower than the next day high. Next, we ensure the predicted volume of sales (valued at the traded price) is less than the available inventory of stocks. If that is not the case, we cap the predicted amounts. We do not trade in case the sell conditions do not hold. Finally, we record the cash proceeds of our sale and the number of stocks sold. Likewise, for buying, we cap the traded price at the current day close ask. We consider the purchase successful if the predicted price is higher than the next day low. Then, we uniformly rescale the purchased value for each stock in case our predicted budget is greater than the

Algorithm 1 Trading algorithm for one day

```
for each stock do
    if value < 0 then
        cap price at close bid
        if price > next high then
            continue
        end if
        cap value at inventory × price
        record sale
    else if value > 0 then
        cap price at close ask
        if price < next low then
            continue
        end if
        rescale value up to cash
        record purchase
    end if
end for
next cash = cash + sales − purchases
next stocks = stocks + purchases − sales
```

available cash balance. We ensure that our predictions satisfy our constraints, and calculate the amount of stocks we bought and record the cost of purchase. Last, we update our stock inventory and cash balance for the next day. This process is repeated for each day, building incrementally on the inventory. In the next section we present a complete outline of the trading algorithm, which defines a differentiable expression for tracking the inventory of cash, n_t, and stocks, $m_{t,i}$, as a function of the market data and the model's trading actions.

Trading Algorithm Equations

In the following, we outline the equations of Trading Algorithm 1. Assume the predicted sell $p^-_{t,i}$ and buy $p^+_{t,i}$ prices, and sell $v^-_{t,i}$ and buy $v^+_{t,i}$ volumes (denominated in USD) are given for day t and stock i. In addition, we have the current cash n_t, stock balances $m_{t,i}$ and market ticker q available.

For selling (denoted by $-$), we cap the price at the current close bid, $p^{-(1)}_{t,i} = \max(q_{t,i,b}, p^-_{t,i})$ and consider the trade successfully executed if the predicted price is lower than the next day high, $s^- = \mathbb{1}[p^{-(1)}_{t,i} \leq q_{t+1,i,h}]$, where $\mathbb{1}$ is the indicator function. We make sure the predicted amount is limited by the available inventory of stocks $v^{-(1)}_{t,i} = \min(v^-_{t,i}, m_{t,i} p^{-(1)}_{t,i})$. We trade only if the sell conditions hold, which we verify by applying the trading filter $v_{t,i}^{-(2)} = s^{-} v_{t,i}^{-(1)}$. Then we record the cash proceeds of our sale $n_t^{-(1)} = \sum_i v_{t,i}^{-(2)}$ and the number of stocks sold $m_{t,i}^{-(1)} = \frac{v_{t,i}^{-(2)}}{p_{t,i}^{-(1)}}$.

Likewise, for buying (shown as +), we cap the traded price at the current day close ask $p_{t,i}^{+(1)} = \min(q_{t,i,a}, p_{t,i}^{+})$. We consider the purchase successful if the prediction price is higher than the next day low, $s^{+} = \mathbb{1}[p_{t,i}^{+(1)} \geq q_{t+1,i,l}]$. Then, we uniformly rescale the purchased value for each stock in case the sum of our predicted budget is greater than the available cash balance $v_{t,i}^{+(1)} = v_{t,i}^{+} \min(1, \frac{n_t}{\sum_i v_{t,i}^{+}})$. We ensure that our predictions satisfy our constraints by applying the trading filter $v_{t,i}^{+(2)} = s^{+} v_{t,i}^{+(1)}$. Last, we calculate the amount of stocks we bought $m_{t,i}^{+(1)} = \frac{v_{t,i}^{+(2)}}{p_{t,i}^{+(1)}}$, and record the cost of purchase $n_t^{+(1)} = \sum_i v_{t,i}^{+(2)}$.

Finally, we update our stock inventory $m_{t+1,i} = m_{t,i} - m_{t,i}^{-(1)} + m_{t,i}^{+(1)}$ and cash balance $n_{t+1} = n_t + n_t^{-(1)} - n_t^{+(1)}$ for the next day. This process is repeated for each day, building incrementally on the inventory.

Profit Objective

The objective of the algorithm is to maximize the trading utility or returns over the entire period, where we use log returns for numerical stability. This is simply the ratio of the value of holdings (cash and stocks valued at closing price) at the last and first time steps

$$U = \log \frac{R(T)}{R(1)} \quad \text{where} \quad R(t) = n_t + \sum_i q_{t,i,c} m_{t,i}.$$

We perform the minimization of the negative utility with the L-BFGS (Byrd et al., 1995) algorithm. In order to limit overfitting, and recover from areas of the objective function where the gradient is zero – for example for predictions where one or more model constraints failed and there is no gradient signal – we add an L2 regularizer to the objective with coefficient λ

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} -U(\boldsymbol{\Theta}) + \lambda ||\boldsymbol{\Theta}||_F^2.$$

The objective is discontinuous but can be optimized using sub-gradient methods.

Alternative Formulations

In Section 6.2.2 we have introduced a model formulation for trading stocks in an end-to-end learning fashion. Here we consider alternative network architectures that offer various tradeoffs with regard to the original formulation. First, Equation 6.1 can be replaced with average pooling of the word embeddings $\mathbf{k_t} = \text{pool}_{\text{avg}}(\mathbf{E_t})$. While max pooling identifies few key phrases in the news, average pooling embeddings conveys the average document sentiment. Although convolutions allow for reweighing of the word vectors to obtain a more relevant representation to our problem domain, fixing the word vectors saves computation time.

In Equation 6.2 we flattened the stock price and industry matrices to obtain a joint market representation. To make this more efficient, we can treat companies in isolation and perform a convolution over each company independently. Then we can aggregate the resulting company vectors through max pooling, $\mathbf{o_t} = \text{pool}_{\text{max}}\,\mathbf{J_t}$. This biases the network such that a few companies can accurately describe the larger market. As many companies move together, especially in an industry, this is reasonable. When discovering temporal signal in the data, Gated Recurrent Units (Cho et al., 2014) provide an alternative to LSTMs. This may be attractive due to decreased computation time requirements, which however results in slightly lower modelling capacity. The GRU formulation of Section 6.2.2 can be implemented with update and reset gates, through which the proposed and actual hidden states for the next time step are output $\mathbf{h}_t^g = \text{GRU}(\mathbf{a}_t, \mathbf{h}_{t-1}^g)$.

In the output formulation in Equation 6.3, we predicted limit order trades directly. An alternative approach is to predict allocation percentages of all stocks plus cash with a softmax output $n_{t+1}, \mathbf{m}_{t+1}\mathbf{q}_{t,c} = \text{softmax}(\mathbf{v}_t)R(t)$. This allows us to reweigh the value of our holdings. From this output then we can derive how much we should trade for each stock to assume the new predicted position. Note that here we also predict the cash balance which was previously implicitly determined. The output can be further formulated another way. Instead of predicting sell and buy decisions in the same output unit, we can separate these two and predict buys and sells with two different rectifier units for value and price. This allows the model to learn different strategies for buying and selling and therefore it is not forced to condense this information into the behavior of a single unit. The four outputs are $p_{t,i}^-, p_{t,i}^+, v_{t,i}^-, v_{t,i}^+ = \langle \mathbf{B_{t,i}}, \mathbf{W_{p,i}} \rangle$. This also allows the network to make simultaneous buy and sell transactions at different prices. In Section 6.2.2 we did not make any assumption with regard to the execution order of trades. However, the downside of this is that the network needs multiple time steps to completely rebalance its portfolio. Therefore, we can allow the network to use not only its cash balance but also the proceeds of sales when making purchases. Another downside of the 4-way output formulation is that if the output value is below zero, then a rectifier activation caps it at 0, which means the trading signal cannot backpropagate. Note, we do not apply an exponential to the value predictions, because we want the network to be able to hold a stock and avoid trading. To mitigate the gradient issue, we therefore introduce linear penalties to the objective that punish the network for violating soft constraints during training. This allows the model to recover from badly behaving areas of the objective with a small gradient signal. The penalties are of the form

$$\Lambda(a, b) = \lambda_l \max(b - a, 0)$$

where the soft constraints P encode the previously described constraints on the profit formulation, but this time penalizing any violations linearly. Note that using an L2 penalty is insufficient in this case, since an L2 would pull back all predictions towards zero, while predicting anything below zero will still result in no gradient. This gives us our modified objective

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} -U(\boldsymbol{\Theta}) + P(\boldsymbol{\Theta}).$$

Last, we explore different formulations of the objective itself. So far we have considered optimizing for returns directly. An alternative is to optimize for risk adjusted returns, with objective $U(\boldsymbol{\Theta}) = \log \mu - \log \sigma$, where

$$\mu = \frac{1}{T} \sum_{t=2}^{T} \left(\frac{R(t)}{R(t-1)} - 1 \right)$$
$$\sigma = \sqrt{\frac{1}{T} \sum_{t=2}^{T} \left(\frac{R(t)}{R(t-1)} - 1 - \mu \right)^2}$$

are the mean and standard deviation of the returns over all trading days, respectively. We hypothesize that this formulation will result in lower returns in the test set, due to optimizing for a different training criterion than our evaluation metric, but nonetheless those returns will be more consistent since the standard deviation is also minimized. Therefore, this should result in more steady profits with lower volatility and hence be more statistically different from the market rate of return. This also ties into being able to model different investor preferences, as alluded to in Section 2.1.

6.2.3 Validation

Our dataset consists of stock chart data (open, high, low, close, volume, closing bid, closing ask), industry codes, news data and economic indicators. The data spans 17.5 years from June 1994 to January 2012 and includes 727 stocks and 288 industries listed on the Nasdaq and New York stock exchanges. We only keep companies during the periods they were part of the S&P 500 index. We filter the companies to their index membership period in order to avoid survivorship bias in the results, as predicting stocks that we know are going to be part of the index at one point considerably inflates returns.

The economic indicators include the consumer price index, government bond yields of various terms (1, 3 months, 1, 2, 5, 7, 10, 20, 30 years), risk free market rate, market momentum, size and spread, foreign exchange rates (USD/GBP, USD/EUR, USD/TWD), and the aggregate index returns. The news corpus[2] includes news articles released by the New York Times, Los Angeles Times, Associated Press, Washington Post, Bloomberg, Agence France-Presse, Central Taiwan and Xinhua New Agencies for the entire period containing 4 billion words in 10 million articles and a multitude of news genres.

Real world trading comes with many risks that are very difficult to model due to limited computational resources. One such element is systematic risk, i.e. the collapse of the financial ecosystem. These black swan events may be so rare as to not even be present in historical data. However, we note that there are multiple financial crashes in our dataset, therefore our results take these into account. Second, a very realistic simulation of markets requires fine grained data, such as orderbook information, which would make it more feasible to ascertain the impact of our trades on the market. In this work, instead of focusing on very short or long term patterns, we attempt to uncover medium term daily dependencies over several months and treat unmodelled aspects of the environment as noise. However, we note that our results also take this noise into account by measuring statistical significance of returns from the baseline buy and hold.

[2]English Gigaword 5th edition, `http://catalog.ldc.upenn.edu/LDC2011T07`

All stock data has been adjusted for splits and dividends. For news data, we only keep word tokens that have occurred at least 1000 times in the corpus. We concatenate all news articles for a given day and feed that into the model. Each word token is initialized to pretrained word2vec (Mikolov et al., 2013a) embeddings which we then fix in order to decrease memory and time requirements of model training. All model weights are initialized by drawing from a small random uniform distribution. The cash and stock balances are initialized to small numbers such that we have equal value in each stock and non-zero cash balance.

We evaluate the algorithm using a sliding window approach where we train on four months of data and test on the following month. We reinitialize the model in each window, and all hidden states and cash and stock states propagate from the training to the testing period. We validate our L2 regularizer coefficient and the number of epochs for early stopping the L-BFGS iterations on the first window. In the model we extrapolate the prices of the stocks that are not tradeable from the nearest valid trading day. This leads to an implicit zero return for the non-tradeable days with no recorded volume.

Our benchmark is the passive buy and hold strategy, which maintains a uniform investment in stocks listed in the S&P 500 index.Note that the S&P 500 index composition changes with time, as do stock prices, and accordingly this baseline method does perform regular small trades. Also, note that this benchmark differs slightly from the S&P 500 index because there were some historical companies missing from our data, and we use a uniform investment instead of weighing investment by market capitalization. The net result is that our baseline slightly outperforms the index. Buy and hold is a recommended way of investing, as it maximizes diversification of one's holding among multiple uncorrelated assets while at the same time it minimizes unsystematic risk by mitigating the effect of stock volatility. The method achieves the market rate of return, corresponding to the overall growth in the economy. A second baseline is to invest all funds into the stock that achieved the maximum return in the training set. We measure statistical significance with the Wilcoxon signed rank test applied to daily test returns. This assumes independence of returns based on economic theories, such as CAPM (Sharpe, 1964). We aim to train a model that not only achieves a large return, but also significantly differs from the buy-and-hold baseline. The reason for this is that the returns over several days are compounded, and hence the variances of daily returns also multiply, yielding a very high overall variance for the period. Therefore, it is possible to achieve a very high return in the stock market purely out of luck. Our null hypothesis states that our model is not different from the benchmark buy-and-hold. Therefore, we want to ensure that our results provide sufficient evidence of following a different returns trajectory than that of the baseline. Where companies are removed from the index, we record a constant return, i.e. we exit any trading positions at the last traded price.

We compare several models in the validation. We aim to find a model that performs reasonably well, denoted by `Designated` in Table 6.6, and plug various components in and out of this network architecture, observing how the results change. This allows us to isolate key components of the network. We identify several aspects of the network to modify as detailed in Section 6.2.2. These include the processing of economic, news and company data, the extraction of temporal dynamics, the inclusion of stock and cash inventory, the type of output units and prediction formulations and the various optimization objectives with different regularizers.

6.2.4 Results

The model `Designated` consists of a company convolution with max pooling, a hidden layer, a joint GRU, another hidden layer, a ReLU value prediction unit, the trading price fixed at the last close price, and optimizing total returns. In Table 6.6 we can see that `Designated` achieves a 10.576x return on the entire test dataset, which compares favorably to the 5.442x return obtained with a uniform buy-and-hold strategy. This translates to 15.0% annual return for the model and 11.2% for buy and hold. This result is obtained with a statistical significance of $p = 0.088$. The second baseline model achieves a return of 16.266x but with the statistical significance of $p = 0.953$, which means that picking the maximum return stock in the training set over the period of four months and going all in for the next month appears to be a profitable strategy, but this profit comes mainly by chance. This can be also verified in Figure 6.17 showing the high variance in total compound returns.

Furthermore, most models beat the baseline buy and hold, but with various significance levels. We observe a small improvement over `Designated` by doing joint company modelling by flattening all company vectors into a single vector (denoted `Comp joint` in Table 6.6), as opposed to company convolutions with max pooling. This means that the algorithm can extract more joint interaction from the market this way, while with convolution this interaction can only be

Model	U	p
Buy and hold	5.442	1.0
Max train	16.266	0.953
Designated	10.576	0.088
Risk adj	8.484	0.053
Econ	10.459	0.090
News avg	10.387	0.093
News conv	10.938	0.086
Comp joint	10.980	0.087
Softmax	7.344	0.774
Exp price	3.739	0.253
4 way pred	3.628	0.006
Inventory	15.316	0.232
Tied GRU	10.626	0.092
LSTM	10.851	0.088

Table 6.6: Various model formulation test returns U (higher better) with statistical significance p (lower better) from buy and hold.

modelled in a post reduction step with pooling. Using only economic indicators (`Econ`), the model performs slightly worse and with less significance. If we exclusively feed news data into the model, then by average pooling the embeddings (`News avg`), our performance drops and significance worsens. However, by doing news convolution with max pooling (`New conv`), we see an increase in the amount of profit achieved, as well as better significance. Using convolutions, we are able to reweigh the word embedding vectors to our task at hand, and this also enables us to better model interaction of phrases with max pooling.

Optimizing for risk adjusted profits (`Risk adj`) instead of plain profits (`Designated`) achieves a better significance level than other comparative models, but at the cost of slightly lower returns. This demonstrates a more conservative strategy, as the risk adjusted returns objective also minimizes the daily variance of returns, therefore it brings a more steady stream of profits. Due to the high impact of the optimization objective on the results, in the future we intend to optimize instead the Sortino ratio (Sortino and Price, 1994), which only penalizes negative variance in the returns, but makes no statement about positive returns. In addition, borrowing from economic theory, we can construct a utility curve for the trader, and take more risks when the trading balance is high, and be more conservative when the balance is low.

Next, we analyse how changing the output formulation affects the model. We can see that using `Softmax` as the output does not help, probably due to easier overfitting in the model. With a

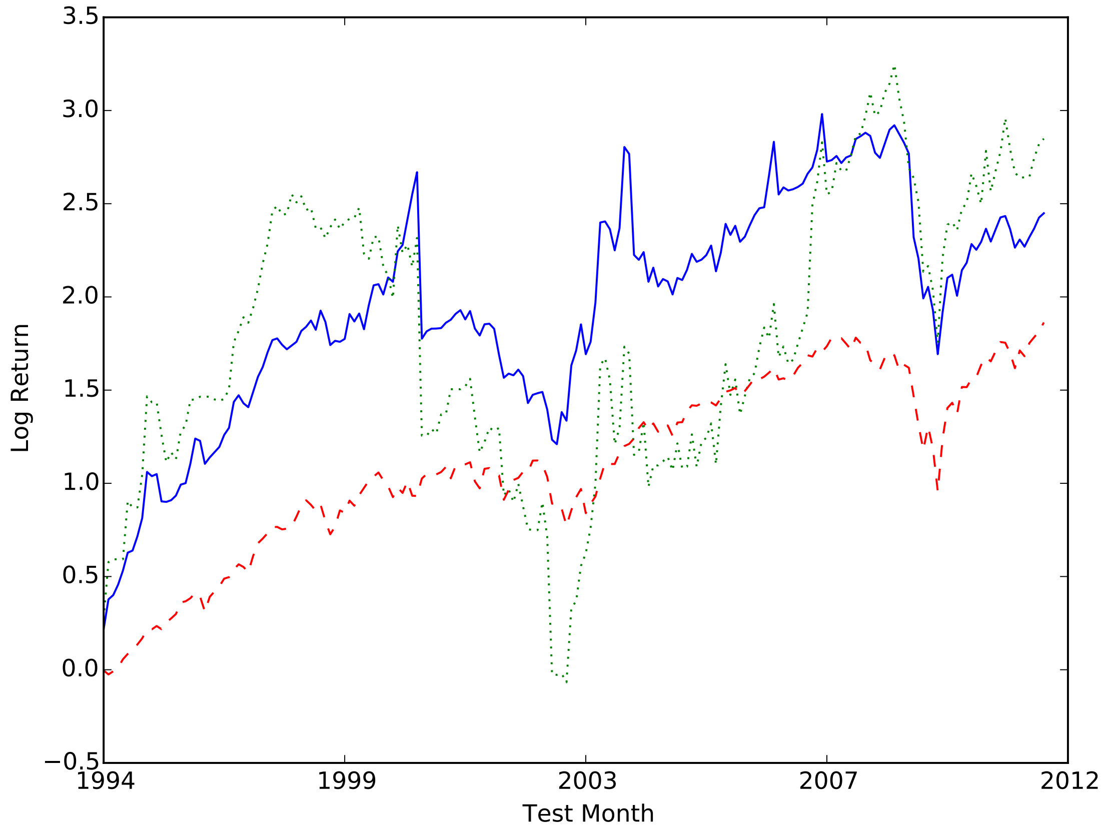


Figure 6.17: Log returns comparing designated model (blue solid) with buy-and-hold (red dashed) and maximum training (green dotted) baseline strategies, shown for 196 test months.

softmax output the model can abruptly change the composition of its portfolio, whereas a ReLu amount output predicts transitioning into a new asset portfolio in a smoother way. That is partly because the softmax predicts end balance composition, while the ReLu amount predicts the change in the composition. Furthermore, we can see that predicting future spread as a relative price movement on the current closing price does not result in improvement either (`Exp price`). This would have allowed the model to trade at a better price than the last close. However, most likely what is happening is that many trades fail due to the prediction being outside the bounds of the next day's high and low. Because this is a hard fail, there is little gradient signal to pull the model back. Exploring further regularization techniques however might help here in future work. Finally, predicting 4 way output is also difficult for the model (`4 way pred`). Due to failed trades, it is easy for the model to miss sudden upward swings in the price and remain stuck with cash only, which is a very different strategy from buy and hold, hence the significant difference. Here, we note that in earlier experiments on synthetic data as well as on a related dataset with FTSE100 companies on the London Exchange, we managed to achieve good results with this model, but these results were highly dependent on finding the ideal regularization trade-off between L2, linear penalties and the model profit objective. Predicting simultaneous buy and sell positions is akin to market making which is more beneficial in a sideways market. However, when there is a strong trend, it may miss out on price swings.

Changing intermediate structures in the network, we observe that using a GRU unit that is tied across all companies (`Tied GRU`), and attempt to model each company in isolation achieves a similar result as the designated model, though with worse significance. This way, the network could still learn embeddings for each company, which are useful for predicting in the next interval. Adding `Inventory` input to the network increases the initial magnitude of activations, therefore yields a higher variance in the amount invested, which explains the increased returns but worse significance levels observed. Using an `LSTM` unit instead of a GRU performs better due to increased modelling capacity, although the GRU is less computationally intensive. Finally, note that if we disable all inputs the performance degrades moderately, but still outperforms the buy-and-hold strategy albeit with worse significance (not shown in table). This indicates that through the individual output weights for each company, the network is able to encode information with regard to the identity of the companies.

In summary, using the designated model provides the advantage of being robust to overfitting with little regularization, while at the same time it achieves high returns that exceed the baseline buy and hold. Due to its relative simplicity, it can also be trained quickly with low memory requirements – training on one 4-month window takes around two minutes on a 2.7 Ghz multicore CPU.

Figure 6.17 shows that the model consistently outperforms the baseline buy and hold over the entire test period. It does suffer considerably during the 2008 financial crisis and during the 2003 Iraqi war. The maximum training model baseline does achieve occasionally higher returns than the buy and hold and the model, but we see a lot more variance in these returns. Just before the financial crisis, the maximum return baseline abruptly increases the revenue, only to lose almost everything in the crisis. For the model, we see similar jumps, but they are significantly less pronounced. In addition, the achieved profits are more consistent over time, steadily outperforming the market. Potentially in a real life setting, a lower maximum drawdown of the model could be useful to provide cash flow forecasts or make better use of leverage in margin trading. In addition, training on longer periods could allow the model to learn an implicit Kelly criterion (Kelly Jr, 1956) for finding an optimal trade-off between risk-seeking and risk-avoiding behaviour.

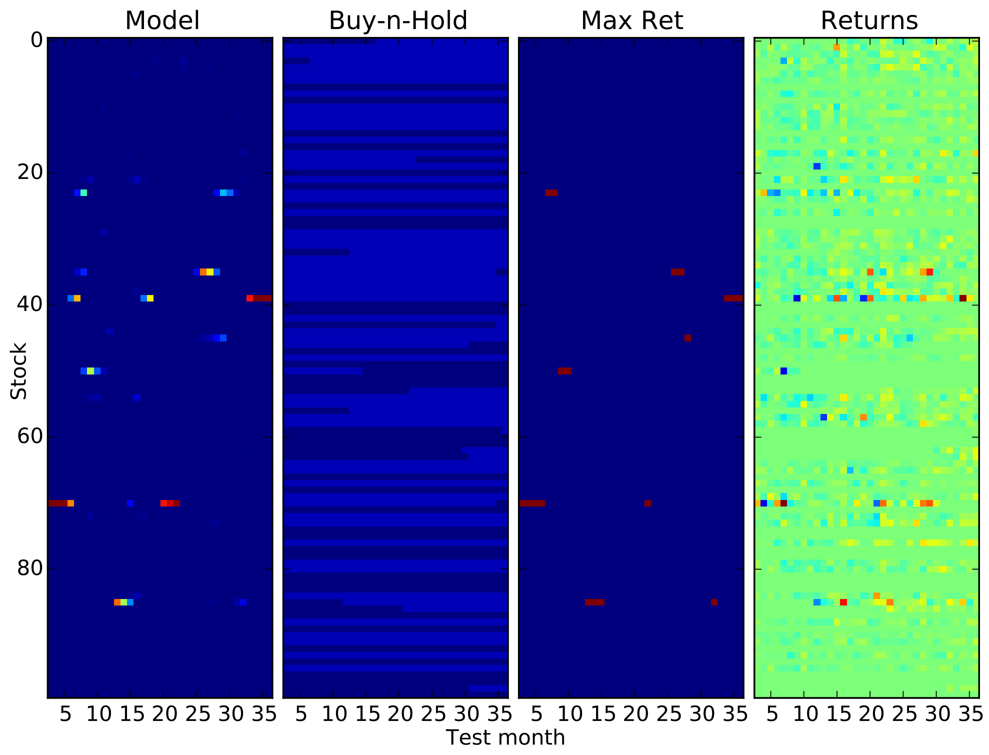


Figure 6.18: Trading strategies visualization on the portion of the test set. Figures show stocks selected by a) designated model, b) buy-and-hold, c) max return baseline, d) actual stock returns. The color denotes the size of investment within each sub plot. Non-trading stocks are masked. Brighter colors mean relatively larger size of investment.

Figure 6.18 shows a visualization of various trading strategies on randomly selected stocks and test month periods. Plot 6.18a shows that the model learns a sparse representation of stocks for investment. At the beginning of training, it starts with a uniform investment strategy and with each optimization iteration, it sparsifies its representation more before early stopping activates. As a result, it only invests in a small percentage of test set stocks and with various degrees. On the other hand, the buy and hold strategy in Plot 6.18b invests all its funds equally in all active stocks, reweighing it each month. Here, the amount of investment allocated to each stock is relatively small in order maximize diversification. The maximum return strategy in Plot 6.18c invests in only a single stock. Looking at the actual stock returns in Plot 6.18d, we can see that the model made several good trades for stocks 70 and 39. For stock 39, the model even caught the high growth period in month 17, which the maximum return baseline ignored. Although sparse, the model is still considerably more diversified than the max return baseline. In addition, we can observe a long term investing strategy where stocks are held for several months rather than exploiting short term daily price movements, and the model portfolio changes smoothly over many months in several instances. This is a more realistic real world trading scenario as it is unlikely that one could abruptly divest and reinvest in stocks following the recommendation of the max return baseline strategy without paying a large fee in trading commissions or market spread.

6.2.5 Conclusion

We have demonstrated an end-to-end differentiable learning framework for day trading stocks that takes raw input data and produces limit order trades. Our algorithm can model joint interactions of market dynamics over a variety of input modalities, achieving high returns on 18 years of stock data over 727 companies, exceeding the returns of the competitive buy and hold investment strategy. In the future, we will experiment with different network topologies and create better simulations of market trading in which our model can learn.

This summarizes our experiments with deep neural networks. We have seen that DNNs are powerful learning machines that make minimal assumptions about the problem. By automatically discovering good non-linear feature representations for multi-modal temporal data and optimising for the training criterion, i.e. profit, while producing actionable output, they can considerably improve over previous state-of-the-art machine learning models, and produce significant risk-adjusted excess returns on multiple markets consistently over a long period of time. Linking back to the research questions in chapter 1, we showed that text based forecasting systems can achieve very good results for movie revenue prediction, which demonstrates that signal can be found in text. In addition, temporal dynamics turned out to result in improvements in the recurrent net experiments for stock data. Learning a good representation of the data was also important with deep projections, which highlights the importance of sufficient depth for learning. Both in the stock market and revenue prediction experiments, domain adaptation further improved the results, though when the model could already infer the domain from the data without explicit input for the domain identity, such as via phrasing in the critic reviews, then the improvement was less pronounced. We also found evidence for the benefits of incorporating multi modal data, such as text, time and structured, all combined. As a consequence of deep structures, non-linearity turned out to be very important, with up to 40% relative gain compared to linear models. Finding the right objective, and optimizing for that also affected the results considerably, especially when noting the difference between risk adjusted and pure returns in the stock market experiments. In the end, all this demonstrates the usefulness of end-to-end learning from raw input data, such as market data and news, to actionable outputs, i.e. limit orders, while optimising for a real world metric people care about, that is net balance.

In the next section, we draw the conclusions of the thesis based on our experiment outcomes with linear, Bayesian, and deep learning models that were tested on a variety of markets under diverse conditions, and give guidelines for future work.

Chapter 7

Conclusions and Future Directions

7.1 Summary of Thesis Achievements

In this thesis, we have demonstrated that it is possible to construct trading systems based on artificial intelligence, machine learning and natural language processing that are able to achieve risk free excess trading profits above the market rate of return obtained via a buy-and-hold strategy. We have tested multiple novel learning systems coming from the linear, Bayesian and deep learning paradigms on a diverse set of markets and trading periods with positive results. This way we have provided evidence against the Efficient Market Hypothesis, and showed the benefits of active portfolio management as opposed to passive investment strategies. Therefore, we have satisfied the primary goal of this research. Below, we detail the other three major research outcomes.

In the first research question, we examined what the characteristics of our datasets are. For example, does technical analysis work? To answer this, we have created a linear profit maximisation model with input technical analysis features. We have shown that technical analysis increases predictive power of the system over just using a window of historical returns as input (section § 4). In addition, we examined whether fundamental analysis is valid. We predicted peer-to-peer loan rates using a variety of fundamental analysis indicators and have shown that some of the indicators were predictive of the interest rates of loan. These included bad debt ratio, and the interest rate selected by crowds, and the term length, which we identified via Automatic Relevance Determination (§ 5.2). This way, we supported the validity of fundamental analysis.

Next, we investigated how text analysis interacts with other forecasting methods and whether it can improve the overall forecast accuracy of markets. When predicting loan rates with topic modelling and a bag of words approach from user generated comments, we showed that we could beat the baseline and achieved good accuracy (§ 5.2). In addition, we found the textual data to contain complementary signal to meta data with some overlap. Likewise, when the task was to predict future box office revenue from movie reviews, we achieved a high accuracy compared to the baseline. In this case, we highlighted that some of the text representations captured structured data in the unstructured document, while others provided additional predictive power not found in meta data. We also showed a novel interpretation technique where we were able to identify high impact phrases in the reviews (§ 6.1). Finally, when predicting stock markets, we showed that using news paper articles, our model was able to beat the buy-and-hold baseline (§ 5.2). In conclusion, we can say that text information has predictive power for financial markets, by providing both complementary and substitutionary signal to structured data.

Following text analysis, we inspected what the temporal characteristics of the dataset are. We found that financial data is non-stationary in that underlying dynamics change over time and we investigated several ways to incorporate time. With a linear profit maximisation model, we created a multi task model with regularizers that acted on the temporal dimension of the data, regularizing subsequent months to be close each other in the weight space, while allowing for smooth variation. This resulted in significant improvements in the test profits compared to the pooled or independent single task models (§ 4). For predicting loan rates, we incorporated several time dimensions via stationary kernels and showed that this positively affected the prediction accuracy compared to models without temporal information (§ 5.2). In the neural stock trader, we constructed several temporal recurrent connections over subsequent trading days by taking both local company and global wider market context into account (§ 6.2). Finally, in our experiment setups we used sliding window validation, where we trained on earlier datapoints and validated and tested on later ones, for example in both stock forecasting experiments as well as in the movie revenue prediction experiment (§ 6.2, § 5.1, § 6.1). By adapting the models to changing market conditions this way, we were properly able to account for temporal dynamics in financial markets.

The second research outcome dealt with how to best prepare the data and feed it into the learning systems. First, we investigated in what way data should be represented. We found that automatically learning representations of text gives superior results to bag of word representations. This way, we were able to save considerable development time in skipping feature engineering based on manually specified heuristics. Our learned representations were optimal for the task at hand, while human engineered features were based on heuristics that were sub-optimal for the task. For this, we compared a linear bag of words model to a deep convolutional neural network with learned word embeddings, and showed that the neural network considerably outperformed the linear model (§ 6.1, § 6.2).

Next, we looked at whether domain adaptation and multi task learning can be beneficial for improved forecasting accuracy. We showed that we can break down our primary task into individual sub tasks, but at the same time share common knowledge between these tasks and keep private information separate. In the FTSE 100 stock prediction experiment, the company regularizers we introduced resulted in big improvements in test profit. We performed the same method over time, where we constructed sub-tasks over subsequent months and forced them to be close together while allowing for individual variations. This increased predictive performance (§ 4). In the final S&P 500 stock experiment, we fed industry codes along with company prices into the neural network and this way, we were able to model the different domains of markets in a non-linear fashion. We showed that we can extract relevant information from multiple related markets that can improve predictive accuracy when forecasting any single market (§ 6.2). In the text regression experiment, we were able to uncover domain information indirectly from the text, and therefore gaining limited improvement by explicitly injecting domain information into the network (§ 6.1). This shows that improved predictive performance can be achieved by taking domain information into account.

Moreover, we showed how to integrate all the different sources of information into the models. This dealt with multi-modal data, such as structured data, unstructured text or temporal information. First for predicting FTSE 100 companies, we represented company histories via technical analysis or by a window of returns history (§ 4). In the peer-to-peer lending experiment, we incorporated different modalities of data via a novel kernel combination technique, and showed that using all information combined in our system, we achieved a good predictive accuracy (§ 5.2). In predicting, movie revenues, we combined movie review text along with structured data for the movies in a hierarchical manner, and found this to improve test set performance (§ 6.1). Finally, in predicting S&P 500 companies, we found that by inputting economic, news, and company data with industry information, we were able to beat the buy and hold baseline and thereby achieve excess profits (§ 6.2). These experiments showed the benefits of modelling multi-modal data individually and jointly.

In the final third research area, we explored what the intricacies of the modelling process itself are. First, we investigated how much effect non-linearities have. We found evidence that financial markets are fairly non-linear and therefore we achieved improved predictive accuracy by explicitly modelling these non-linearities. For this, we again compared to a linear model and when predicting movie revenue, we achieved better performance via non-linear projections. In loan rate prediction for social lending, our rational quadratic non-linear kernel considerably outperformed the simpler linear kernel in a Gaussian Process or Support Vector Machine (§ 5.2). For predicting stocks, using multiple non-linear transformations over time, we were able to outperform the buy and hold baseline (§ 6.2). Finally, in the multi task learning setting, breaking the main prediction task down into sub-tasks by trading month and company and constructing separate locally linear model for globally non-linear market dynamics, we were able to improve the test profit earned (§ 4). This highlights the importance of avoiding underfitting complex financial data, and constructing sufficiently high variance models that can capture this complexity, while at the same time limiting overfitting with adequate regularisation.

Taking the concept of non-linearity further, we investigated whether deep hierarchical structures could be beneficial for our task. We found that modelling the problem as a series of deep non-linear transformations that can be composed of one another considerably improved generalisation performance. In the movie review prediction task, we found that using multiple hidden layers in the neural network achieved the best test set performance. In contrast, shallow one layer representation of bag of words data did not perform that well in the movie revenue task (§ 6.1). For predicting the S&P 500 companies, our model was deep over the multi-modal data as well as over time via recurrent connections, and this way we were able to beat the baseline trading strategy (§ 6.2). This gives support for breaking down the problem into small parts and learning a good composition function and higher level concepts efficiently via stacked deep representations.

Next, we asked the question as to how to model uncertainty in market dynamics. We demonstrated that using a probabilistic interpretation of trading profit, we were able to incorporate uncertainty in the models and were less likely to overfit to a single overconfident solution. We first introduced uncertainty by interpreting the linear FTSE 100 model prediction where we could denote the predictions as model confidence, and using this interpretation, we were able to achieve significant profits in the test data. We then expanded on the linear MAP solution using a Bayesian Laplace approximation and by exercising multiple hypotheses, we were able to improve over our test set profits. Putting our approximation into the Gaussian Process framework, we were further able to improve on the test set profits by integrating over all possible hypotheses, as opposed to selecting a single hypothesis (§ 5.1). In the peer-to-peer loan experiments, we found that Gaussian Process outperformed Support Vector Machine by integrating over all latent functionals that can explain the data. The likelihood of the model also significantly correlated with the test set error, even though the marginal likelihood was maximised on the training set alone, with no validation. Last, by having a probabilistic output, we were able to construct a profit maximisation objective on top of the GP posterior, resulting in significant profits in a simulation of arbitrage (§ 5.2). These experiments highlight the benefit of incorporating uncertainty when modelling financial markets.

In addition to uncertainty, we investigated what the best ways of optimizing our models are. For this, we examined the effect of encoding the trading profit directly into the objective function of the system, instead of optimizing for a proxy problem via regression or classification. We showed that optimizing for profits directly boosts test set profits over other optimizing measures. In the FTSE 100 experiment, we compared to a ridge regression model, in which case we were not able to beat the baseline buy and hold, but with optimizing for profits we were (§ 4). We also encoded our objective via Laplace approximation in the Gaussian Process framework, and achieved considerable returns (§ 5.1). When predicting peer-to-peer loan rates, optimizing for arbitrage opportunities on top of the Gaussian Process model we trained, we were able to earn excess trading profits in a trading simulation (§ 5.2). In addition, optimizing for the final net worth of our artificial neural stock trader, we were able to beat the baseline buy-and-hold over several decades of test data with significance (§ 6.2). Finally, by deciding to be more permissible to outliers, we were able to accurately forecast movie revenues with mean absolute rather than squared error (§ 6.1). These experiments show that we must accurately encode the metric we care about, i.e. profit or revenue, when designing new learning algorithms, as test set performance will improve this way.

Finally, we tested out whether improvements can be achieved and development costs reduced in an end-to-end learning framework. We made the argument for end-to-end learning, by which we fed raw economic, news, company and industry data into our neural stock trading system and directly output limit order trades. This way, we were able to skip pipelining several disparate systems together, preventing error propagation, or skip constructing human heuristics for pre and post processing our data, i.e. via feature engineering. This saved us considerably development time, and additionally our end-to-end learning framework was able to achieve significant profits on the test set by directly optimizing for the net worth of its balance at the final step of the simulation instead of classification or regression accuracy. This way, we were able to simulate a real life human stock trader that reads news, economic and market data each day, performs some trades by issuing limit orders, while trying to increase the worth of their portfolio. This system generated excess profits by beating the baseline buy and hold strategy with statistical significance over almost 20 years of data by trading almost a 1000 companies during the period, and while controlling for any kind of survivorship bias in our simulation (§ 6.2). With the movie revenue prediction task, we similarly skipped all preprocessing of the input, combined multi-modal data effectively and directly predicted for revenue in an end-to-end learning fashion. This resulted in a 40% relative improvement to the previous state-of-the-art system relying on model pipelining (§ 6.1). In peer-to-peer lending, we applied a kernel method to learn a non-linear infinitely high dimensional projection of various input data, ranging from temporal data, i.e. raw timestamps, structured fundamental indicators and unstructured loan descriptions and user comments. By applying a novel kernel combination technique, we were able to learn a good joint representation for the data and predict loan rates by taking into account inherent uncertainty. Then, on top of this, we were able to optimize for arbitrage opportunities directly (§ 5.2). These experiments highlighted the need for joint modelling of the problem end-to-end, with raw data as input, actionable predictions as output, and optimization for the right objective.

7.2 Future Work

In this section, we give some possible directions for future work.

First, we can extend the simulations to further markets under even more diverse set of conditions. In addition to traditional markets, such as other developed stock markets and fiat peer-to-peer lending sites, possible new datasets could include the very data rich, fairly open and transparent social peer-to-peer lending sites enabled by cryptocurrency technology, such as Bitcoin. These sites provide not only rich unstructured text in the form of social interactions, along with structured data, but also the activity history of individual investors and borrowers. Aside from peer-to-peer lending, cryptocurrency can also be employed for operating decentralized asset exchanges, where anybody can list their own assets at negligible cost. It would be an interesting question to validate the performance of the proposed algorithms under these circumstances and construct new ways of modelling additional data available.

Second, our systems can directly be applied to old or new emerging electronic markets and tested on live data. For trading in a live environment, a further research field that can be examined is combining supervised learning with reinforcement learning. In general, training a supervised learning framework based on a simulated environment can be used as a baseline trading algorithm, which can be gradually adapted to the peculiarities of live market feedback with a reinforcement learning framework. This enables the algorithm to detect deficiencies in any of the assumptions made by the original supervised learning algorithm, and narrow the gap between simulation and reality. This follows the curriculum learning (Bengio et al., 2009) paradigm which progressively makes the prediction task more difficult and thus more realistic. Recent work has successfully applied this concept to playing old video games (Mnih et al., 2015) or the game of Go (Silver et al., 2016).

Last, we can make some extensions directly based on the work presented in this thesis. In particular, in the Gaussian Process framework, the Intrinsic Model of Coregionalization can be performed for multi-task learning over related datasets. Further, online learning methods can be explored to allow for a different modelling of non-stationarity. The modelling targets can also be modified to encode different properties of financial markets, such as loan markups or likelihood of default in the peer-to-peer lending sphere. To make regression models more robust to outliers, the Student-T likelihood can be used. For better modelling of multiple tasks and categories of data, hierarchical kernels are another avenue to explore. For neural networks, we can explore hierarchical or recursive convolutions to model sub-sets of text data incrementally, as well as investigate automatic ways of optimising for the hyperparameters of the networks, i.e. using Bayesian sequential optimisation. By modifying the network topologies in our end-to-end learning framework and creating better simulations of market trading, our models may be able to learn better. In the future, we aim to explore additional methods of addressing the above listed challenges of financial market modelling.

In conclusion, we have made the argument for computational modelling of financial markets successfully. Our data driven techniques based on artificial intelligence have incorporated time, structured data and unstructured text for forecasting markets, and traded on them in a completely automated manner. With this, we have showed that we can achieve risk free excess profits on multiple markets over long stretches of time, and thus we have provided evidence against the Efficient Market Hypothesis.

Appendix A

Appendix

A.1 List of Company Symbols

The following companies were used in the experiments reported in the chapter. Shown below are the stock symbols on the London Stock Exchange.

CPI REX BG AGK BA AZN AAL AMEC CNA LGEN BARC BP REL STAN CRH GSK PRU LAND MGGT RR DGE SAB NG BLND RIO WTB SMIN RB RDSB LLOY SGE SHP NXT IMI BLT MKS MRW RBS WEIR WOS HMSO SRP IMT GKN ADN SVT BSY VOD TSCO SDR UU GFS HSBA RSA OML TATE SN AV KGF PSON BATS ULVR

A.2 Technical Indicators

A.2.1 Overlap Studies

```
BBANDS               Bollinger Bands
An exponentially decaying moving average plotted together with price
series one standard deviation above and below the average. In case the price
goes beyond the standard deviation threshold, that could signal a buy or
sell trading action
```

DEMA Double Exponential Moving Average

2xEMA() - EMA(EMA()). Moving average of moving average

EMA Exponential Moving Average

Moving average with an exponentially decaying factor

HT_TRENDLINE Hilbert Transform - Instantaneous Trendline

Formed by removing the dominant phase from a real valued analytic like detrended signal, such as a price series. Involves analysis of the instantaneous trendline.

KAMA Kaufman Adaptive Moving Average

Moving average designed to account for market noise and volatility. When volatility is high, it increases the backward looking horizon, while when volatility is low, it decreases it.

MA Moving average

Simple moving average of returns with fixed window size

MAMA MESA Adaptive Moving Average

Adapt to price movement based on the rate change of phase as measured by the Hilbert Transform Discriminator

MAVP Moving average with variable period

Moving average with period that can be adapted to volatility and smoothness

MIDPOINT MidPoint over period

Average of highest and lowest value over period

MIDPRICE Midpoint Price over period

Average of lowest and highest close value over period.

SAR	Parabolic SAR
SAREXT	Parabolic SAR - Extended

SAR indicator aims to identify the momentum of price movements, where it
may have a higher than normal probability of switching directions.

SMA	Simple Moving Average

Moving average of close given a target period window

T3	Triple Exponential Moving Average (T3)

Double exponential moving average with a smoothing factor, called three times.

TEMA	Triple Exponential Moving Average

Similar to DEMA, but called three times.
3 x EMA() - 3 x EMA(EMA()) + EMA(EMA(EMA()))

TRIMA	Triangular Moving Average

Moving average where weights are assigned following a triangular pattern,
i.e. more weight towards middle of period.

WMA	Weighted Moving Average

Weighted moving average with custom weights

A.2.2 Momentum Indicators

ADX	Average Directional Movement Index

Measures the strength of the prevailing trend and denotes whether
movement exists in the market for that trend. High number indicates
a strong whereas a low number a weak trend.

ADXR	Average Directional Movement Index Rating

Average ADX at current timestamp and at n timestamps ago

APO Absolute Price Oscillator

Difference between two moving averages. One is slow, other is fast moving average.

AROON Aroon

Indicates the trend of the security and likelihood it will reverse.
It assumes uptrends result in all time highs and downtrends in all time lows.

AROONOSC Aroon Oscillator

Calculated by subtracting Aroon up from Aroon down.

BOP Balance Of Power

Close minus Open divided by High minus Low. Strength of buyers and sellers is measured by how much they can push the price to extreme levels

CCI Commodity Channel Index

Price minus moving average divided by normal deviation. Measures investment overbought or oversold.

CMO Chande Momentum Oscillator

Sum of recent gains minus sum of recent losses divided by sum of all price movement.

DX Directional Movement Index

Shows whether an asset is trending or not. Measures upward and downward trends separately

MACD Moving Average Convergence/Divergence

Trend following indicator that shows the relationship between two moving averages of prices. It subtracts the 26 day EMA from the 12 day EMA.

MACDEXT MACD with controllable MA type

As MACD, but the type of moving average can be changed

MFI Money Flow Index

Measured the inflow and outflow of security over a period

MINUS_DI Minus Directional Indicator

Percentage of true range that is down

MINUS_DM Minus Directional Movement

Measures strong downward trend

MOM Momentum

Gradient of price change compared to n steps ago. (price-prevPrice_n)

PLUS_DI Plus Directional Indicator

Percentage of true range that is up

PLUS_DM Plus Directional Movement

Measures strong upward trend.

PPO Percentage Price Oscillator

Shows relationship between two moving averages. 9 day EMA minus 26 day EMA divided by 26 day EMA.

ROC Rate of change

Last return ((price/prevPrice)-1)

ROCP Rate of change Percentage:

Last return in percentage (price-prevPrice)/prevPrice * 100

ROCR	Rate of change ratio

Last return ratio (price/prevPrice)

RSI	Relative Strength Index

Used to measure oversold or overbought securities. Mark of 30 and 70 is the threshold for trading action in the range of 0-100 for a trading period.

STOCH	Stochastic

Measures where the closing price is relative to previous highs and lows.
If at the extreme, the security is over-bought or -sold. It is
smoothed by a 3 day moving average.

STOCHF	Stochastic Fast

STOCH, but more sensitive to volatility, unsmoothed.

STOCHRSI	Stochastic Relative Strength Index

Stochastic oscillator applied to the relative strength index (RSI). Oversold under 0.2 and overbought abobe 0.8.

TRIX	1-day Rate-Of-Change (ROC) of a Triple Smooth EMA

Rate of change of Triple exponential moving average of price.

ULTOSC	Ultimate Oscillator

Momentum oscillator that incorporates three different time periods.

WILLR	Williams’ %R

Shows close price relative to high and low over a period of time. Used for establishing entry and exit points. Same as STOCHF.

A.2.3 Volume Indicators

AD	Chaikin A/D Line

Measure the accumulation / distribution line (volume times relative price) over a period and apply MACD on that. Calculates money flow index, which is current price relative to high and low.

ADOSC	Chaikin A/D Oscillator

Momentum of AD.

OBV	On Balance Volume

Increasing volume without price change will result in price jump.

A.2.4 Cycle Indicators

The Hilbert Transform is a technique used to generate inphase and quadrature components of a de-trended real-valued signal in order to analyze variations of the instantaneous phase and amplitude.

HT_DCPERIOD	Hilbert Transform - Dominant Cycle Period

Analyze variations of the instantaneous cycles.

HT_DCPHASE	Hilbert Transform - Dominant Cycle Phase

Analyze variations of the instantaneous phase and amplitude.

HT_PHASOR	Hilbert Transform - Phasor Components

In phase and quadrature components of the price series.

HT_SINE	Hilbert Transform - Sine Wave

Sine of Phase advanced by 45 degrees or at specific bar.

HT_TRENDMODE Hilbert Transform - Trend vs Cycle Mode

Analyzes trend vs cycle patterns.

A.2.5 Price Transform

AVGPRICE Average Price

high + low+ close + open divided by 4.

MEDPRICE Median Price

high + low divided by 2

TYPPRICE Typical Price

high + low + close divided by 3.

WCLPRICE Weighted Close Price

Average price with more weight given to the close. 2*close+high+low/4

A.2.6 Volatility Indicators

ATR Average True Range

Measure of volatility, average of the true range (highest - lowest times previous time step TR). Low when price is flat.

NATR Normalized Average True Range

ATR / close * 100. Normalizes ATR across instruments.

TRANGE True Range

High minus low, including previous close. High volatility followed by low volatility.

A.2.7 Pattern Recognition

CDL2CROWS	Two Crows

Bearish trend reversal. On big increase, and two slight decreases in price indicates the peak of price has been reached.

CDL3BLACKCROWS	Three Black Crows

Bearish moderate trend reversal with increasing decreases. Opposite of three advancing soldiers.

CDL3INSIDE	Three Inside Up/Down

Confirmed bullish Harami pattern. One big decrease with two slight increases, not entirely surpassing the decrease. Can be reversed for down.

CDL3LINESTRIKE	Three-Line Strike

Three bullish increases with a big decrease, striking through. Indicates reversal.

CDL3OUTSIDE	Three Outside Up/Down

Confirmed bullish engulfing. One decrease and two big increases.

CDL3STARSINSOUTH	Three Stars In The South

Three bearish decreases which slows down. Indicates reversal.

CDL3WHITESOLDIERS	Three Advancing White Soldiers

Three large bullish increase, usually after a reversal. Opposite of three black crows.

CDLABANDONEDBABY	Abandoned Baby

Bearish and bullish candle with a doji (same open and close) in the middle that is out of range of both neighbour candles. Indicates reversal. Can be also bearish.

CDLADVANCEBLOCK Advance Block

Similar to 3 advancing soldiers but the last increase candle is weaker, indicating that the bulls are running out of momentum.

CDLBELTHOLD Belt-hold

After several bearish candles, a long bullish candle forms where the open price is low and the close price is high.

CDLBREAKAWAY Breakaway

Strong downwards candle with several minor bearish candles finishing with a bullish candle recovering some losses.

CDLCLOSINGMARUBOZU Closing Marubozu

Strong selling in control with a long black candle that has no lower shadow. Can signal panic or a dump.

CDLCONCEALBABYSWALL Concealing Baby Swallow

Two strong downtrends with a smaller third down candlestick, finishing with a large down candlestick enveloping almost all previous ones.

CDLCOUNTERATTACK Counterattack

Large uptrend followed by a large downtrend, closing at the same price as the opening.

CDLDARKCLOUDCOVER Dark Cloud Cover

Bullish candle followed by a reversal and bearish candle that is confirmed by an even lower third bearish candle.

CDLDOJI Doji

A candle that is only shadow, and has no body. That means the opening and closing prices are the same. Signals reversal.

CDLDOJISTAR Doji Star

Preceeding the doji, there's a bullish or bearish candle. Can indicate trend continuation.

CDLDRAGONFLYDOJI Dragonfly Doji

Doji where the price is at the high of the day. Indicates indecision or trend reversal.

CDLENGULFING Engulfing Pattern

A small bearish candle followed by a larger bullish one that completely engulfs it. The bulls have taken control.

CDLEVENINGDOJISTAR Evening Doji Star

Doji star at the end of a bullish cycle, followed by a strong bear candle.

CDLEVENINGSTAR Evening Star

Large bullish candle followed by a small bull candle and a large bearish candle. Indicates trend reversal.

CDLGAPSIDESIDEWHITE Up/Down-gap side-by-side white lines

Bearish candle followed by lower side by side bullish patterns, or bullish candle followed by higher side by side bullish patterns.

CDLGRAVESTONEDOJI Gravestone Doji

Doji where the opening and closing is at the low of the day. Indicates bearish reversal.

CDLHAMMER Hammer

Hammer formed with a lower stick shadow and a bearish or bullish head.

CDLHANGINGMAN Hanging Man

Significant selloff after an uptrend with buyers pushing price back to top. Indicates bearish reversal.

CDLHARAMI	Harami Pattern

Two almost side by side bearish candles with a small bullish candle following.

CDLHARAMICROSS	Harami Cross Pattern

Large bearish or bullish candle followed by a doji cross in the middle.

CDLHIGHWAVE	High-Wave Candle

Three small body candles with large shadows forming a wave. Indicates end of trend.

CDLHIKKAKE	Hikkake Pattern

Starts with a bar inside the previous one, followed by a large breakout.

CDLHIKKAKEMOD	Modified Hikkake Pattern

Like Hikkake pattern, but preceding the inside bar closes at top of range, indicating sell, or low of the range, indicating buy.

CDLHOMINGPIGEON	Homing Pigeon

Large bearish candle with a small bearish candle inside the first one.

CDLIDENTICAL3CROWS	Identical Three Crows

Uptrend followed by three large descending bearish candles. Indicates reversal again.

CDLINNECK	In-Neck Pattern

Large bearish candle followed by a small bullish candle below the bearish one.

CDLINVERTEDHAMMER	Inverted Hammer

Like hammer pattern but the head of the hammer is at the bottom. Occurs at bottom of downtrends.

CDLKICKING   Kicking

A large marubozu (bearish candle) followed by a large bullish candle with a disconnect between them.

CDLKICKINGBYLENGTH   Kicking - bull/bear determined by the longer marubozu

Kicking with bull or bear is determined by which candle is longer.

CDLLADDERBOTTOM   Ladder Bottom

The bearish candles with fourth one losing momentum and fifth being strongly bullish.

CDLLONGLEGGEDDOJI   Long Legged Doji

Doji with a very long shadow. Signals stronger indecision.

CDLLONGLINE   Long Line Candle

A candle with a long body. May signal reversal.

CDLMARUBOZU   Marubozu

A long candle without any shadow. Signals strong trend setting. Either bullish or bearish.

CDLMATCHINGLOW   Matching Low

Two bearish candles that match at their lower parts. Indicates strong support.

CDLMATHOLD   Mat Hold

One candle in one direction followed by three candles in opposite direction, followed by a final candle in same direction. Indicates trend continuation.

CDLMORNINGDOJISTAR   Morning Doji Star

Like morning star but the in between candle is a doji.

CDLMORNINGSTAR Morning Star

Large bearish candle followed by a large bullish candle and a small candle lower in between. Indicates bullish reversal.

CDLONNECK On-Neck Pattern

A brearish candle followed by a bullish one straight below it. Indicates bearish continuation.

CDLPIERCING Piercing Pattern

Two side by side candles with large bodies, first bearish, second bullish and slightly below first. May signal reversal.

CDLRICKSHAWMAN Rickshaw Man

A long legged doji pattern with the cross in the middle and long shadows. Indicates strong indecision.

CDLRISEFALL3METHODS Rising/Falling Three Methods

Long bearish candles with three small rising bullish candles and large bearish candle again. Rising can be reversed for falling.

CDLSEPARATINGLINES Separating Lines

Uptrend with a bearish candle at the end, and a bullish candle jumping above the one before. Signals trend continuation.

CDLSHOOTINGSTAR Shooting Star

Similar to inverted hammer pattern that follows a strong bullish candle. May indicate reversal.

CDLSHORTLINE Short Line Candle

A candle with a short body. Can signal indecision.

CDLSPINNINGTOP Spinning Top

A short line candle with a longer top or bottom. Signals reversal.

CDLSTALLEDPATTERN Stalled Pattern

Like three advancing soldiers or three black crows, but the last candle diminishing in importance. Signals the trend has run out of momentum.

CDLSTICKSANDWICH Stick Sandwich

Three candles with the middle being in between the side and having opposite color and shadows on both ways. Looks like a sandwich.

CDLTAKURI Takuri (Dragonfly Doji with very long lower shadow)

Single candle with long down shadow and small upper body. Indicates bullish reversal.

CDLTASUKIGAP Tasuki Gap

Two bullish candles with a gap between them and followed by a bearish candle in between. Indicates trend continuation.

CDLTHRUSTING Thrusting Pattern

Bearish candle followed by a slightly lower bullish candle. Bearish continuation.

CDLTRISTAR Tristar Pattern

Three doji stars with the middle being lower. Indicates bullish reversal.

CDLUNIQUE3RIVER Unique 3 River

Large bearish candle, with small bearish candle with long shadow and small bullish candle. Reversal or continuation.

CDLUPSIDEGAP2CROWS Upside Gap Two Crows

Bullish candle followed by two higher bearish ones. Indicates bearish reversal.

CDLXSIDEGAP3METHODS Upside/Downside Gap Three Methods

Similar to upside gap two crows, but with the first bearish candle in fact being bullish, and the third one being large bearish. Indicates reversal.

Bibliography

Abarbanell, J. S. and Bushee, B. J. (1997). Fundamental analysis, future earnings, and stock prices. *Journal of Accounting Research*, 35(1):1–24.

Adams, R. P. and MacKay, D. J. (2007). Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

Akita, R., Yoshihara, A., Matsubara, T., and Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *15th International Conference on Computer and Information Science*, pages 1–6. IEEE.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609.

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. (2011). Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*.

Atiya, A. F. (2001). Bankruptcy prediction for credit risk using neural networks: A survey and new results. *IEEE Transactions on Neural Networks*, 12(4):929–935.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics.

Becker, G. S. (2013). *The economic approach to human behavior*. University of Chicago Press.

Beddington, J., Furse, C., Bond, P., Cliff, D., Goodhart, C., Houstoun, K., Linton, O., and Zigrand, J.-P. (2012). *Foresight: The future of computer trading in financial markets: Final project report.* The London School of Economics and Political Science, Systemic Risk Centre.

Bengio, Y. (1997a). Training a neural network with a financial criterion rather than a prediction criterion. In *Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM'96), World Scientific Publishing*, pages 36–48.

Bengio, Y. (1997b). Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, 8(04):433–443.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems.*

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., et al. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, volume 4.

Bitvai, Z. and Cohn, T. (2015a). Day trading profit maximization with multi-task learning and technical analysis. *Machine Learning*, 101(1-3):187–209.

Bitvai, Z. and Cohn, T. (2015b). Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, page 180.

Bitvai, Z. and Cohn, T. (2015c). Predicting peer-to-peer loan rates using Bayesian non-linear regression. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2203–2210.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *The Journal of Political Economy*, pages 637–654.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Bondt, W. F. and Thaler, R. (1985). Does the stock market overreact? *The Journal of Finance*, 40(3):793–805.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. (2007). Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 20, pages 153–160.

Boser, B. E., Sackinger, E., Bromley, J., Le Cun, Y., and Jackel, L. D. (1991). An analog neural network processor with programmable topology. *Solid-State Circuits, IEEE Journal of*, 26(12):2017–2025.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149.

Bousquet, O. and Bottou, L. (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, pages 161–168.

Box, G. E. (1976). Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Cadieu, C. F., Hong, H., Yamins, D. L., Pinto, N., Ardila, D., Solomon, E. A., Majaj, N. J., and DiCarlo, J. J. (2014). Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Computational Biology*, 10(12):e1003963.

Campbell, J. Y., Lo, A. W.-C., MacKinlay, A. C., et al. (1997). *The econometrics of financial markets*, volume 2. Princeton University.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Cha, S.-M. and Chan, L. (2000). Trading signal prediction. In *International Conference on Neural Information Processing—ICONIP*, pages 842–846.

Chakoumakos, R., Trusheim, S., and Yendluri, V. (2012). Automated market sentiment analysis of Twitter for options trading. Master's thesis, Stanford University.

Chamberlain, G. (1983). A characterization of the distributions that imply mean—variance utility functions. *Journal of Economic Theory*, 29(1):185–201.

Chen, R. and Lazer, M. (2012). Sentiment analysis of Twitter feeds for the prediction of stock market movement. Master's thesis, Stanford University.

Chen, X., He, J., Lawrence, R., and Carbonell, J. G. (2012). Adaptive multi-task sparse learning with an application to fMRI study. In *SDM*, pages 212–223. SIAM.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Christensen, B. J. and Prabhala, N. R. (1998). The relation between implied and realized volatility. *Journal of Financial Economics*, 50(2):125–150.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., and Olukotun, K. (2006). Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems*, volume 6, pages 281–288.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chyan, A., Hsieh, T., and Lengerich, C. (2012). A stock-purchasing agent from sentiment analysis of Twitter. Master's thesis, Stanford University.

Ciodaro, T., Deva, D., De Seixas, J., and Damazio, D. (2012). Online particle detection with neural networks based on topological calorimetry information. *Journal of Physics: Conference Series*, 368(1):1203.

Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338.

Cockburn, A. (2002). *Agile software development*, volume 177. Addison-Wesley Boston.

Cohn, T. and Specia, L. (2013). Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Csató, L. and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Damianou, A., Ek, C., Titsias, M., and Lawrence, N. (2012). Manifold relevance determination. *arXiv preprint arXiv:1206.4610.*

Damianou, A. C. and Lawrence, N. D. (2012). Deep Gaussian processes. *arXiv preprint arXiv:1211.0358.*

Daniel, K. and Titman, S. (2006). Market reactions to tangible and intangible information. *The Journal of Finance*, 61(4):1605–1643.

Dauphin, Y. N., de Vries, H., Chung, J., and Bengio, Y. (2015). RMSProp and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390.*

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941.

De la Vega, J. (1688). *Confusión de confusiones.* Editorial Maxtor.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231.

Debbini, D., Estin, P., and Goutagny, M. (2012). Modeling the stock market using Twitter sentiment analysis. Master's thesis, Stanford University.

Degutis, A. and Novickyte, L. (2014). The efficient market hypothesis: A critical review of literature and methodology. *Ekonomika*, 93(2):7.

Deterding, S. (2012). Gamification: Designing for motivation. *Interactions*, 19(4):14–17.

Dooley, M. P. and Shafer, J. (1976). *Analysis of short-run exchange rate behavior: March, 1973 to September, 1975.* Board of Governors of the Federal Reserve System.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, 9:155–161.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Elder, A. (1993). *Trading for a living: Psychology, trading tactics, money management*, volume 31. John Wiley & Sons.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117. ACM.

Fama, E. F. (1998). Market efficiency, long-term returns, and behavioral finance. *Journal of Financial Economics*, 49(3):283–306.

Fama, E. F. and French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Felleman, D. J. and Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47.

Frencha, C. W. (2003). The Treynor capital asset pricing model. *Journal of Investment Management*, 1(2):60–72.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian data analysis*, volume 2. CRC Press.

Gerber, E. M., Hui, J. S., and Kuo, P.-Y. (2012). Crowdfunding: Why people are motivated to post and fund projects on crowdfunding platforms. In *Proceedings of the International Workshop on Design, Influence, and Social Technologies: Techniques, Impacts and Ethics.*

Ghani, R. and Simmons, H. (2004). Predicting the end-price of online auctions. In *Proceedings of the International Workshop on Data Mining and Adaptive Modelling Methods for Economics and Management*.

Ghosn, J. and Bengio, Y. (1997). Multi-task learning for stock selection. *Advances in Neural Information Processing Systems*, pages 946–952.

Gilbert, E. and Karahalios, K. (2010). Widespread worry and the stock market. In *The International AAAI Conference on Web and Social Media - ICWSM*, pages 59–65.

Giles, C. L., Lawrence, S., and Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning*, 44(1-2):161–183.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press.

Gopikrishnan, P., Plerou, V., Amaral, L. A. N., Meyer, M., and Stanley, H. E. (1999). Scaling of the distribution of fluctuations of financial market indices. *Physical Review E*, 60(5):5305.

Graham, B. and Dodd, D. L. (1934). *Security analysis: Principles and technique*. McGraw-Hill.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Greenberg, M. D., Pardo, B., Hariharan, K., and Gerber, E. (2013). Crowdfunding support tools: Predicting success & failure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1815–1820. ACM.

Grinold, R. C. and Kahn, R. N. (2000). *Active portfolio management*. McGraw Hill.

Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Haugen, R. A. and Lakonishok, J. (1987). *The incredible January effect: The stock market's unsolved mystery.* Dow Jones-Irwin.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013a). Gaussian processes for big data. *arXiv preprint arXiv:1309.6835.*

Hensman, J., Lawrence, N. D., and Rattray, M. (2013b). Hierarchical Bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BioMed Central Bioinformatics*, 14(1):252.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hsu, E., Shiu, S., and Torczynski, D. (2012). Predicting Dow Jones movement with Twitter. Master's thesis, Stanford University.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.

Jensen, M. C. (1968). The performance of mutual funds in the period 1945–1964. *The Journal of Finance*, 23(2):389–416.

Jensen, M. C. (1978). Some anomalous evidence regarding market efficiency. *Journal of Financial Economics*, 6(2/3):95–101.

Jensen, M. C., Black, F., and Scholes, M. S. (1972). The capital asset pricing model: Some empirical tests. *Studies in the Theory of Capital Markets.*

Joshi, M., Das, D., Gimpel, K., and Smith, N. A. (2010). Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296. Association for Computational Linguistics.

Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, pages 263–291.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.*

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468.*

Kelly Jr, J. L. (1956). A new interpretation of information rate. *IRE Transactions on Information Theory*, 2(3):185–189.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319.

Kim, S.-D., Lee, J. W., Lee, J., and Chae, J. (2002). A two-phase stock trading system using distributional differences. In *Database and Expert Systems Applications*, pages 143–152. Springer.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., and Smith, N. A. (2009). Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280. Association for Computational Linguistics.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Kuleshov, V. (2012). Can Twitter predict the stock market? Master's thesis, Stanford University.

Lampos, V., Aletras, N., Preoţiuc-Pietro, D., and Cohn, T. (2014). Predicting and characterising user impact on Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 405—413.

Lampos, V. and Cristianini, N. (2010). Tracking the flu pandemic by monitoring the social web. In *Cognitive Information Processing*, pages 411–416.

Lampos, V., Preotiuc-Pietro, D., and Cohn, T. (2013). A user-centric model of voting intention from social media. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 993–1003.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Le Cun, B. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*. NIPS.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H. (2012). Using Twitter to estimate and predict the trends and opinions. Master's thesis, Stanford University.

Lerman, K., Gilder, A., Dredze, M., and Pereira, F. (2008). Reading the markets: Forecasting public opinion of political candidates by news analysis. In *Proceedings of the 22nd Inter-*

national Conference on Computational Linguistics-Volume 1, pages 473–480. Association for Computational Linguistics.

Liew, J. K.-S. and Mayster, B. (2017). Forecasting ETFs with machine learning algorithms. *SSRN eLibrary*.

Lo, A. W. (2002). The statistics of Sharpe ratios. *Financial Analysts Journal*, pages 36–52.

Lo, A. W., Mamaysky, H., and Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4):1705–1770.

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274.

Magdon-Ismail, M., Nicholson, A., and Abu-Mostafa, Y. S. (1998). Financial markets: Very noisy information processing. *Proceedings of the IEEE*, 86(11):2184–2195.

Mairal, J. (2012). SPAMS: A SPArse Modeling Software, v2.3. *Inria*.

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *The Journal of Economic Perspectives*, 17(1):59–82.

Mandelbrot, B. and Hudson, R. L. (2014). *The misbehavior of markets: A fractal view of financial turbulence*. Basic Books.

Markowitz, H. M. (1968). *Portfolio selection: Efficient diversification of investments*, volume 16. Yale University Press.

Maslow, A. H., Frager, R., Fadiman, J., McReynolds, C., and Cox, R. (1970). *Motivation and personality*, volume 2. Harper & Row New York.

McClelland, J. L. and Rogers, T. T. (2003). The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience*, 4(4):310–322.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

Mittal, A. and Goel, A. (2012). Stock prediction using Twitter sentiment analysis. Master's thesis, Stanford University.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Neely, C., Weller, P., and Dittmar, R. (1997). *Is technical analysis in the foreign exchange market profitable? A genetic programming approach*. Cambridge University Press.

Nelson, C. R. and Plosser, C. R. (1982). Trends and random walks in macroeconmic time series: Some evidence and implications. *Journal of Monetary Economics*, 10(2):139–162.

Nesterov, Y., Nemirovskii, A., and Ye, Y. (1994). *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436. IEEE.

Nicholas, H. (2011). *Marx's theory of price and its modern rivals*. Palgrave Macmillan Basingstoke.

Nicholson, S. F. (1960). Price-earnings ratios. *Financial Analysts Journal*, pages 43–45.

Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371.

Nison, S. (2001). *Japanese candlestick charting techniques: A contemporary guide to the ancient investment techniques of the Far East.* Penguin.

Obozinski, G., Taskar, B., and Jordan, M. I. (2010). Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252.

O'Hara, M. (2003). Presidential address: Liquidity and price discovery. *The Journal of Finance*, 58(4):1335–1354.

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20.

Owen, J. and Rabinovitch, R. (1983). On the class of elliptical distributions and their applications to the theory of portfolio choice. *The Journal of Finance*, 38(3):745–752.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Rajaraman, A. and Ullman, J. D. (2011). *Mining of massive datasets.* Cambridge University Press.

Ravi Kumar, P. and Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review. *European Journal of Operational Research*, 180(1):1–28.

Resnick, P. and Zeckhauser, R. (2002). Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *Advances in Applied Microeconomics*, 11:127–157.

Ridley, M. (1994). *The red queen: Sex and the evolution of human nature*. Penguin.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591. IEEE.

Ro, D. and Pe, H. (1973). *Pattern classification and scene analysis*. Wiley.

Robert, E. and John, M. (1948). Technical analysis of stock trends. *Springfield, MA: Stock Trend Service*.

Rogers, S. and Girolami, M. (2011). *A first course in machine learning*. CRC Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Schöneburg, E. (1990). Stock price prediction using neural networks: A project report. *Neurocomputing*, 2(1):17–27.

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, pages 2503–2511.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442.

Sharpe, W. F. (1970). *Portfolio theory and capital markets*. McGraw-Hill College.

Sharpe, W. F. (1998). The Sharpe ratio. *Streetwise - The best of the Journal of Portfolio Management*, pages 169–185.

Shiller, R. J., Fischer, S., and Friedman, B. M. (1984). Stock prices and social dynamics. *Brookings Papers on Economic Activity*, 1984(2):457–510.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *IEEE*, page 958. IEEE.

Simonoff, J. S. and Sparrow, I. R. (2000). Predicting movie grosses: Winners and losers, blockbusters and sleepers. *Chance*, 13(3):15–24.

Skolidis, G. (2012). *Transfer learning with Gaussian processes.* PhD thesis, The University of Edinburgh.

Smith, A. and Garnier, M. (1838). *An inquiry into the nature and causes of the wealth of nations.* T. Nelson.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Ali, M., Adams, R. P., et al. (2015). Scalable Bayesian optimization using deep neural networks. *arXiv preprint arXiv:1502.05700.*

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 129–136.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Sornette, D. (2009). *Why stock markets crash: Critical events in complex financial systems.* Princeton University Press.

Sortino, F. A. and Price, L. N. (1994). Performance measurement in a downside risk framework. *The Journal of Investing*, 3(3):59–64.

Sperry, R. W. (1968). Hemisphere deconnection and unity in conscious awareness. *American Psychologist*, 23(10):723.

Sprenger and Welpe (2012). Tweets and trades: The information content of stock microblogs. Master's thesis, Stanford University.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Taleb, N. N. (2007). Black swans and the domains of statistics. *The American Statistician*, 61(3):198–200.

Temin, P. and Voth, H.-J. (2004). Riding the South Sea bubble. *American Economic Review*, 94(5).

Terry, N., Butler, M., et al. (2005). The determinants of domestic box office performance in the motion picture industry. *Southwestern Economic Review*, 31(1):137–148.

Tesfatsion, L. (2002). Economic agents and markets as emergent phenomena. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7191–7192.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Tihonov, A. N. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Math*, 4:1035–1038.

Tikhonov, A. N. (1943). On the stability of inverse problems. *Comptes Rendus (Doklady) de l'Academie des Sciences de l'URSS*, 39(5):195–198.

Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *science*, 185(4157):1124–1131.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85.

Ver Hoef, J. M. and Barry, R. P. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339.

Wang, W. Y. and Hua, Z. (2014). A semiparametric Gaussian copula regression model for predicting financial risks from earnings calls. In *Proceedings of the 52nd Annual Meeting on Association for Computational Linguistics*.

Wang, Y.-F. (2002). Predicting stock price using fuzzy grey prediction system. *Expert Systems with Applications*, 22(1):33–38.

Weil, P. (1989). The equity premium puzzle and the risk-free rate puzzle. *Journal of Monetary Economics*, 24(3):401–421.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

Williams, C. K. and Rasmussen, C. E. (2006). Gaussian processes for machine learning. *MIT Press*, 2(3):4.

Winters, L. A. (2004). Trade liberalisation and economic performance: An overview. *The Economic Journal*, 114(493):F4–F21.

Wolkenhauer, O. and Ullah, M. (2007). All models are wrong. *Systems Biology: Philosophical Foundations.*

Xidonas, P., Mavrotas, G., Krintas, T., Psarras, J., and Zopounidis, C. (2012). *Multicriteria portfolio management.* Springer.

Yu, C.-C. and Liu, B.-D. (2002). A backpropagation algorithm with adaptive learning rate and momentum coefficient. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1218–1223. IEEE.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701.*

Zhang, W. and Skiena, S. (2009). Improving movie gross prediction through news analysis. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 301–304. IEEE Computer Society.

Zhang, W. and Skiena, S. (2010). Trading strategies to exploit blog and news sentiment. In *The International AAAI Conference on Web and Social Media - ICWSM.*

Zhang, X., Fuehres, H., and Gloor, P. A. (2011). Predicting stock market indicators through Twitter "I hope it is not as bad as I fear". *Procedia-Social and Behavioral Sciences*, 26:55–62.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.