ONLINE COMPUTATIONAL ALGORITHMS FOR PORTFOLIO-SELECTION PROBLEMS

by

Raphael Ndem Nkomo

Submitted to
the Faculty of Economics and Financial Sciences in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Johannesburg

2015

UNIVERSITY OF Johannesburg

This dissertation was presented

by

Raphael Ndem Nkomo

It was approved by

Dissertation Advisors: ,

Prof Alain Kabundi, South African Reserve Bank & University of Johannesburg.

ONLINE COMPUTATIONAL ALGORITHMS FOR PORTFOLIO-SELECTION PROBLEMS

Raphael Ndem Nkomo, PhD

University of Johannesburg, 2015

Abstract: This thesis contributes to the problem of equity portfolio management using computational intelligence methodologies. The focus is on generating automated financial reasoning, with a basis in computational finance research, through searching a space of semantically meaningful propositions. In comparison with classical financial modelling, our proposed algorithms allow continual adaptation to changing market conditions and a non-linear solution representations in most cases. When compared with other computational intelligence approaches, the focus is on a holistic design that integrates financial research with machine learning. The major aim of the thesis is to develop portfolio allocation techniques for learning investment-decision making that can easily adapt to changes in market processes together with speed and accuracy. We evaluate the algorithms developed in out-of-sample trading framework using historical data sets. The testing is designed to be realistic; for instance, considering factors such as transaction costs, stock splits and data snooping. To demonstrate the robustness of our approach we perform extensive historical simulations using previously untested real market datasets. On all data sets considered, our proposed algorithms significantly outperform existing portfolio allocation techniques, sometimes in a spectacular way, without any additional computational demand or modeling complexity.

Before proceeding any further, we stress that setting up abstract and complex mathematical models is neither the intention nor the scope of this thesis. Our aim rather is to investigate empirically and possibly capture any existing nonlinearities or non-stochasticities that are apparent in the dynamics of cross sectional returns of stock prices. In doing so we utilise some novel techniques, which are mostly based on such methodologies that have been used successfully in the physical sciences were the deterministic dynamics of the phenomena are more easily detected. Our intention is to provide an additional empirical analysis framework that could shed new light in the investigation of the nature of financial time-series data generating processes.

TABLE OF CONTENTS

1.0 GENERAL INTRODUCTION 1
1.1 Introduction 1
1.2 Mechanics of Online Learning for Portfolio Selection 2
1.3 Taxonomy of Online Portfolio Selection Algorithms 4
1.4 Main Contributions 6
1.5 Outline of the Thesis 7
1.5.1 Chapter II: Mathematical Formulation 8
1.5.2 Chapter III: Market Assumptions, Data Description and Performance Measurements 8
1.5.3 Chapter IV: Online Learning Algorithms for PS: An Empirical Survey 9
1.5.4 Chapter V: Kalman Filtering and Online Learning Algorithms for PS 9
1.5.5 Chapter VI: Sequential Portfolio Selection Algorithms Via Delay Coordinate Embedding 10
1.5.6 Chapter VII: Sequential Portfolio Selection Algorithms Via Multivariate Singular Spectrum Analysis 10
1.5.7 Chapter VIII: Low-Regret Stock Selection in Online Learning Algorithms 11
1.5.8 Chapter IX: Conclusion Directions for Future Research 11
2.0 MATHEMATICAL FORMULATION 12
2.1 Introduction 12
2.2 Static Portfolio Selection 12
2.3 Mean-Variance Solution to the Investment Problem 13
2.3.1 Optimisation Problem 14

2.3.2 Analytical Solution 15
2.4 The Log-Optimal Portfolio Solution 16
2.4.1 The Log-Optimal Portfolio in Continuous Time 17
2.4.2 The Log-Optimal Portfolio in Discrete Time 19
2.5 Some Benchmark Portfolio Selection Algorithms 21
3.0 MARKET ASSUMPTIONS, DATA DESCRIPTION AND PERFORMANCE MEASUREMENTS 24
3.1 Introduction 24
3.2 Another Look at the Mathematical Formulation 24
3.3 Market Assumptions 25
3.4 Data Description 26
3.5 Performance Measures and Strategy Evaluation 28
3.5.1 Jensen's Alpha 28
3.5.2 Treynor–Mazuy Market-Timing Model 29
3.5.3 Asymmetric Beta 30
3.5.4 Total Portfolio Cumulative Wealth 30
3.5.5 Annualised Percentage Yield 31
3.5.6 Sharpe Ratio 31
3.5.7 Maximum Draw Down (MDD) 32
3.5.8 Percentage of Profitable Trades 32
3.6 Combining Strategies 33
3.7 Trading Costs and Market Impacts 35
3.8 Conclusion 38
4.0 MACHINE-LEARNING ALGORITHMS FOR PORTFOLIO SELECTION: AN EMPIRICAL SURVEY 39
4.1 Introduction 39
4.2 Kernel-Based Portfolio Selection Algorithm (Gyorfi et al. (2006)) 41
4.2.1 Introduction 41
4.2.2 Pattern Matching for Kernel-Based Algorithm 42
4.2.3 Portfolio Optimisation for Kernel-Based Algorithms 42

4.2.4 Parameter Choice and Combining Portfolio of Experts 43
4.2.5 Empirical Results 46
4.2.5.1 Analysis of the Kernel Cumulative Wealth 46
4.2.5.2 Kernel Strategy Risk Analysis 48
4.2.6 Conclusion 50
4.3 Nearest-Neighbour Algorithm (Gyorfi et al. (2008)) 50
4.3.1 Introduction 50
4.3.2 Nearest-Neighbor Pattern Matching 51
4.3.3 Nearest-Neighbor Portfolio Optimisation 51
4.3.4 Combining Portfolios of Experts 53
4.3.5 Empirical Results 54
4.3.5.1 Analysis of Cumulative Wealth 54
4.3.5.2 Performance Risk Analysis 56
4.3.6 Conclusion 58
4.4 Correlation-Driven Mean Reversion Portfolio Selection Algorithm (Li et al (2011)) 58
4.4.1 Introduction 58
4.4.2 CORN Pattern Matching 59
4.4.3 CORN Portfolio Optimisation 60
4.4.4 Combining Portfolio of Experts 61
4.4.5 Empirical Results 61
4.4.5.1 Analysis of Cumulative Wealth 62
4.4.5.2 Performance Risk Analysis 63
4.5 Anticor Algorithm (Borodin et al. (2004)) 65
4.5.1 Introduction 65
4.5.2 Anticor Algorithm Design 65
4.5.3 Expert Combination 67
4.5.4 Empirical results 68
4.5.4.1 Analysis of the Anticor Cumulative Wealth 68
4.5.4.2 Anticor Performance Risk Analysis 70

4.5.4.3 Anticor Brokerage Costs Analysis 71

4.5.5 Conclusion 72

4.6 Confidence Weighted Mean Reversal (Li et al. (2013)) 73

4.6.1 Introduction 73

4.6.2 CWMR Algorithm Design 73

4.6.3 Combining Portfolio of Experts 75

4.6.4 Empirical Results 76

4.6.4.1 Analysis of Cumulative Wealth 76

4.6.4.2 Performance Risk Analysis 78

4.6.4.3 CWMR Brokerage Costs Analysis 78

4.7 Passive-Aggressive Mean Reversion (Li et al. (2012)) 80

4.7.1 Introduction 80

4.7.2 Algorithm Design and Closed-Form Solution 81

4.7.3 Expert Aggregation 82

4.7.4 Empirical Results 82

4.7.4.1 Analysis of Cumulative Wealth 82

4.7.4.2 Performance Risk Analysis 84

4.7.4.3 PAMR Brokerage Costs Analysis 85

4.8 Summary 86

5.0 KALMAN FILTERING AND ONLINE LEARNING ALGORITHMS FOR PORTFOLIO SELECTION PROBLEMS 88

5.1 Introduction 88

5.2 Mathematical Model 89

5.3 Trend Adjusted Price Relatives 91

5.3.1 Statistical Properties of the TAPR 94

5.3.2 The Scalar Kalman Filter Algorithm 95

5.3.3 TAPR Parameter Settings 99

5.4 Online Portfolio Selection Algorithms Via Trend Adjusted Price Relatives . 99

5.4.1 Momentum Anticor (MAC) Algorithm via TAPR 100

5.4.2 Combining Portfolio of Experts 104

5.5 Empirical results . . . 105
5.5.1 Analysis of Cumulative Wealth . . . 105
5.5.2 KMAC Performance Risk Analysis . . . 107
5.5.3 KMAC Brokerage Costs Analysis . . . 109
5.6 PAMR and CWMR Via TAPR . . . 111
5.7 Experimental Results . . . 112
5.7.1 Parameter choices . . . 113
5.7.2 Performance Comparison . . . 117
5.8 Conclusion . . . 118
6.0 DELAY COORDINATE EMBEDDING AND SEQUENTIAL PORTFOLIO SELECTION ALGORITHMS . . . 120
6.1 Introduction . . . 120
6.2 Mathematical Model . . . 123
6.3 Theory of State-Space Reconstruction . . . 124
6.3.1 Mathematical Foundations of the State-Space Reconstruction . . . 125
6.3.2 State-Space Reconstruction . . . 127
6.4 Embedding Parameters . . . 128
6.4.1 Chosing the Optimal Time Delay . . . 128
6.4.1.1 Autocorrelation Function . . . 128
6.4.1.2 Mutual Information . . . 129
6.4.2 Choice of the Optimal Embedding Dimension . . . 129
6.4.2.1 Singular Value Decomposition (SVD) . . . 130
6.4.2.2 False Nearest Neighbours . . . 130
6.5 Delay Coordinate Embedding for Portfolio Selection (DCEPS) . . . 130
6.5.1 Phase-Space Reconstruction of Multiple Trend-Adjusted Price Relatives 131
6.5.2 Pattern Matching . . . 133
6.5.3 DCEPS Portfolio Optimisation . . . 135
6.5.4 Discussion on Parameter Settings . . . 139
6.5.5 Mixture of Experts . . . 139
6.6 Empirical results . . . 140

6.6.1 Analysis of Cumulative Wealth 140
6.6.2 Performance Risk Analysis 142
6.6.3 DCEPS Brokerage Costs Analysis 144
6.7 Conclusion 145
7.0 MULTIVARIATE SINGULAR SPECTRUM ANALYSIS AND ONLINE PORTFOLIO SELECTION 146
7.1 Introduction 146
7.1.1 Scope of Our Study 149
7.2 Spectral Decomposition and Time-Domain Reconstruction 150
7.2.1 Literature Review of Singular Spectrum Analysis (SSA) 150
7.2.2 Embedding 151
7.2.3 Singular Value Decomposition 152
7.2.4 Rank Reduction and Grouping 153
7.2.5 Diagonal averaging 153
7.3 Multi-Channel Singular Spectrum Analysis 155
7.3.1 Mathematical Formulation of the MSSA 156
7.4 Multivariate Singular Spectrum for PS (MSSAPS) 159
7.4.1 Pattern Formation 160
7.4.2 Portfolio Optimisation 161
7.4.3 Discussion on Parameter Settings 163
7.4.4 Mixture of Experts 164
7.5 Empirical results 165
7.5.1 Analysis of Cumulative Wealth 165
7.5.2 MSSAPS Performance Risk Analysis 167
7.5.3 MSSAPS and Brokerage Costs Analysis 169
7.6 Conclusion 170
8.0 DYNAMIC STOCK SELECTION AND MACHINE LEARNING ALGORITHMS FOR PORTFOLIO SELECTION 171
8.1 Introduction 171
8.2 Literature Review 174

8.3 Mathematical Model 175
8.4 Online Stock Selection Algorithm Using Support Vector Machines 178
8.4.1 A Linear-Beta Multi-Factor Stock-Pricing Model 179
8.4.2 Eonometric Estimation of Linear-Beta Stock Pricing Models 184
8.4.3 Beta Estimation Via Principal Component Analysis 186
8.4.4 Some Practical Considerations 188
8.5 Optimal Separating Hyperplane Via Support Vector Machines 189
8.5.1 Theoretical Foundations of Support Vector Machines 190
8.5.2 Mechanics of the SVM 192
8.5.2.1 Linearly Separable Problems 193
8.5.2.2 Nonlinearly Separable Problems and Soft Margin Hyperplane . 195
8.5.2.3 Extension to nonlinear Decision Boundary 197
8.5.3 Some Practical Considerations 198
8.6 Support Vector Machine for Portfolio Selection (SVMPS) Algorithm 199
8.6.1 Mixture of Experts 202
8.6.2 Trading Costs 202
8.7 Empirical results 203
8.7.1 Analysis of Cumulative Wealth 203
8.7.2 Risk Analysis 205
8.7.3 SVMPS and Brokerage Costs Analysis 207
8.8 Conclusion 208
9.0 SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH 210
9.1 Summary 210
9.2 Directions For Future Research 213
9.2.1 Volatility Clustering and Pattern Selection 213
9.2.2 Stock Market Regime Shifts and Pattern Filtering 213
9.2.3 Directional Predictions and Online Learning for PS Algorithms 214
9.3 General Conclusion 215
BIBLIOGRAPHY 217
.0 STATISTICAL PROPERTIES OF RATIO OF RANDOM VARIABLES 228

1.0 GENERAL INTRODUCTION

1.1 INTRODUCTION

The conventional investment wisdom over the last several decades has been that securities markets were extremely efficient in reflecting information not only about individual stocks but also about the stock market as a whole. In this environment, portfolio managers were advised to Buy-and-Hold good-quality stocks for the long run with the hope that securities markets will rise over time. The accepted view was that because information arose stochastically, the news spread very quickly and was incorporated into the prices of securities without any material delay. Therefore, no systematic investment programme, including technical analysis and fundamental analysis, would enable an investor to achieve returns greater than those that could be obtained by holding a randomly selected portfolio of individual stocks, at least not without taking additional risk.

However, the amount of evidence showing the disadvantage that traditional, long-term, Buy-and-Hold investors currently face is staggering. After the 2008 financial crisis and the resulting poor performance delivered by most fund managers, there is renewed search for reliable active investment strategies that can outperform not only the market but also the best stock.

In recent years the growth of theoretically well grounded algorithms for Online PS problems has been significant. These online portfolio selection (PS) algorithms have demonstrated good finite sample properties with a performance that generally exceeds both the market and the best stock even after accounting for modest transaction costs. More specifically, algorithms such as Universal Portfolio (Cover (1991)), Exponential Gradient (Hembold et al. (1998)) and Online Newton Step (Agarwal et al. (2006)) have demonstrated that the wealth achieved through a sequential rebalancing strategy possesses explicit lower bounds given a sufficiently long period of time.

Although very elegant in their mathematical formulation, most existing state-of-the art algorithms have displayed very disappointing performance in practical applications when compared to some alternative algorithms derived from simple heuristics like the Anticor algorithm of Borodin et al. (2006). The aim of this thesis is therefore to present a set of new Online PS procedures within the context of algorithmic trading that are both theretically well founded and empirically very robust. Our approach is completely nonparametric as it assumes no knowledge of the underlying statistical distribution that generates stock prices. Our goal throughout the thesis will be to find a PS scheme such that the investor's wealth grows at a pace that could have been achieved using an optimal strategy that benefited from the full knowledge of the underlying distribution. This, of course, is a very ambitious undertaking but we will show that the newly proposed Online portfolio-selection algorithms demonstrate very good finite-horison performance when applied to a wide range of stock market datasets.

1.2 MECHANICS OF ONLINE LEARNING FOR PORTFOLIO SELECTION

Online learning for PS is a model of induction that operates in a sequence of consecutive rounds with the main goal of maximising the expected log return over a sequence of trading periods. After each observation, the portfolio manager predicts a portfolio weights vector, which is a way to allocate his available capital amongst investable assets. This portfolio-weight-vector prediction problem is based on all available information including previous market sequences of stock price relatives. The quality of the portfolio manager's prediction is generally assessed by a loss function that measures the discrepancy between the predicted answer and the correct one. The manager's ultimate goal is to minimise the cumulative wealth loss suffered over the long run, which is equivalent to maximising his total wealth. To achieve this goal, the portfolio manager may update the hypothesis after each round so as improve the odds to be more accurate in later rounds. Once the portfolio manager has made the prediction, he or she receives feedback indicating the actual outcome of stock market returns. Then, the portfolio manager calculates his or her period profit or loss and must decide to modify or not the portfolio weight prediction mechanism for the next period, presumably improving the chances of making an accurate prediction in subsequent rounds. Table 1.1 shows the general framework as followed by most Online learning algorithms for Portfolio Selection (PS).

Table 1.1	**General Mechanincs of Online PS Algorithm**
Input	matrix of price relative
Output	vector of portfolio weights
Initialise	uniform distribution of weights vector
for	$t = 1, 2, ...n$ **do**
1	portfolio manager computes portfolio weights
2	market reveals the vector of price relatives
3	portfolio manager incurs a profit or loss
4	portfolio manager updates the prediction rules
end	

Online learning for PS algorithms presents many attractive features compared to their most popular counterparts like the auto-regressive moving average (ARMA) and the auto-regressive integrated moving average (ARIMA) models. First, by readily tackling the vast amounts of historical data that are available in the financial markets, Online learning algorithms can provide insight into the forecasting problem at the core of traditional portfolio choice formulations. Second, Online learing algorithms provide a framework for bypassing stock returns forecasting issues altogether and making direct portfolio allocation choices. Third, many Online algorithms can give strong guarantees on performance even when the instances are not generated by any known distribution. As long as a reasonably good prediction function exists, the Online learning algorithm will learn to predict correct stock prices.

These good prediction functions must come from a previously determined set that depends on the algorithm. Fourth, because Online learning algorithms continually receive feedback, the algorithms are able to adapt and learn in difficult market situations. Fith, Online learning algorithms for PS problems are nonparametric in nature as they make no statistical assumptions regarding the origin of the sequence of stock prices.

However, a principal difficulty or weakness faced by Online PS algorithms is that of following a rigid (black box) methodology. Since history provides only a single realised trajectory of stock prices, making repeated use of the same historical data leads to retrospective bias and excessive optimistic estimates of past performance that vanish when a system is put into deployment. In order to avoid this well-known data snooping bias, we obtain realistic estimates of performance by following a sequential validation methodology, which attempts to reproduce as closely as is feasible the steps that would have been taken in real-life decision making processes.

1.3 TAXONOMY OF ONLINE PORTFOLIO SELECTION ALGORITHMS

Before delving into a detailed description of our proposed online algorithms for PS, we stress that setting up abstract and complex mathematical models is neither the intention nor the scope of this thesis. Our aim rather is to investigate empirically and possibly capture any existing nonlinearities or non-stochasticities that are apparent in the dynamics of cross sectional returns of stock prices. In doing so we utilise some novel techniques, which are mostly based on such methodologies that have been used successfully in the physical sciences were the deterministic dynamics of the phenomena are more easily detected. Our intention is to provide an additional empirical analysis framework that could shed new light in the investigation of the nature of financial time-series data generating processes. Another important point to stress at this early stage is that there are some "irritating" but useful repetitions throughout this thesis. The decision not to do away with these repetitions was based on maintaining a good flow of ideas as the various chapters were individually written and sent to various journals for possible publication.

This section sets out our work within the broader context of some of the more recent work on Online learning for PS problems.

PS algorithms based on Online machine-learning techniques can be classified into four main groups (see Li et al. (2013) for a detailed description). This classification includes "Follow-the-Winner", "Follow-the-Loser", "Pattern-Matching" and "Meta-Learning Algorithms" algorithms.

The first group comprises the so-called, "Follow-the-Winner" strategy that increases the relative weights of better performing stocks with the belief that this performance will carry over into subsequent periods. This class is generally referred to as momentum based strategies since these algorithms implicitly bet on some form of price continuation. Of course, historical performance here could be measured along many, lines including historical price performance, historical earnings revisions or any other firm fundamental attributes etc...A well known strategy in this group is the so called Buy-and-Hold which will be employed in this thesis as an aggregation scheme to combine portfolio of strategies.

The second group is referred to as "Follow-the-Loser" approach where the portfolio manager takes a contrarian view on the recent performance of stock prices. The strategy here is simply to increase the relative weights of recent losers stocks by transferring the weights from the winners to losers. The implicit assumption in following this strategy is that market participants more often do overreact to stock price news, which ultimately results in stock prices being over/under priced with the likelihood of price reversal as the price discovery reasserts itself.

The third approach is called "Pattern-Matching"-based approach. This similarity-driven approach designs a portfolio based on instances of similarity between the most recent price sequence and the whole historical path of price relatives. This strategy is very appealing because it can capture both price reversals and price continuation that seem to cohabitate in financial markets. We will propose new similarity-driven learning to trade sequential PS algorithm that are very competitive within the class of log-optimal algorithms.

The last group is referred to as "Meta-Learning" algorithms. These algorithms essentially recombine any existing base algorithms with the objective of creating a final portfolio by adaptively combining the portfolios suggested by the base algorithms. In particular, Puja et al. (2011) show that the meta algorithms for PS will be universal; i.e., competitive with the best constant rebalanced portfolio (CRP) chosen in hindsight, if any base algorithm in the pool is universal. This result is quite an important one as it allows us to combine our proposed algorithms with those that have been shown to be universal (Blum et al. (1997), Cover (1991), Kalai et al. (2005)) and obtain an algorithm that is universal.

1.4 MAIN CONTRIBUTIONS

The main aim of this thesis is to introduce a general framework for the design and analysis of new Online learning algorithms for PS problems. Our framework includes algorithms with features that include mean reversion, pattern matching and combination strategies. These algorithms emerge from a new view on statistical learning, phase-space reconstruction and spectral analysis, which are the common thread in the analysis of Online PS algorithms as discussed here.

The first main contribution comes from the empirical analysis of some of the more promising Online PS algorithms collected from recent publications. Our analysis shows some rather surprising results when one takes into account more recent and previously untested datasets. Our main finding here is that the vast majority of selected algorithms performed remarkably well on older datasets but the claim that these existing state-of-the-art algorithms are robust and perform well on more broader datasets is rather exagerated in our view. The performance of some state-of-the-art algorithms on more recent market data including those in the UK, South Africa, the US and Canada has been poor at best.

Our next important contribution is to present a new heuristic approach to Online PS that could be used to extend all existing state-of-the-art algorithms. We introduce a state-space model via the Kalman Filter algorithm to filter price-cycle oscillations out of the current share prices and compute the trend-adjusted price relative (TAPR). The TAPR helps to de-noise the stock price data in order to account for the possibility of multi-period mean reversion in stock prices (see Li et al. (2012)). We build on the existing state-of-the-art PS algorithms in general, and the Anticor (Borodin et al.,2006) algorithm in particular and use ideas from signal processing and statistical learning to demonstrate the superiority of our new methodology. Our proposed methodology naturally results in Online PS algorithms that perform exceedingly well on all datasets, including most recent ones. To our knowledge, this is the first time that a research has combined ideas from signal processing with Online learning algorithms to select portfolios in an optimal way.

The third main contribution presents newly developed Online PS, including the Delay Coordinate Embedding Algorithm for PS (DCEPS), the Multivariate Singular Spectrum Analysis for PS (MSSAPS) and the Support Vector Stock Selection Machines (SVSS). As in Gyorfi et al. (2006) our proposed algorithms have their foundations in the similarity-driven nonparametric Online learning methodology. Our new algorithms allow us to construct asymptotically optimal investment strategies in the financial market without prior knowledge of the statistical properties of stock prices. Extensive historical simulations demonstrate that these new state-of-the-art algorithms perform exceedingly well on all datasets. We consider these new Online PS algorithms as very robust investment strategies.

1.5 OUTLINE OF THE THESIS

The present thesis is divided into nine main chapters which are organised as follows; Chapter I presents a General Introduction to the thesis while Chapter II summarises the general mathematical foundation that underpin sequential algorithms for PS. Chapter III describes in detail the data that will be used in the thesis together with the evaluation of some performance measures that will be necessary to gauge the robustness of proposed PS algorithms. In chapter IV, we present an empirical survey of some of the most promising PS algorithms found in the litterature. In Chapter V, VI, VII and VIII we present a detailed analysis of our proposed portfolio allocation strategies and their applications. The conclusion to the thesis is presented in Chapter IX.

1.5.1 Chapter II: Mathematical Formulation

This chapter lays down the theoretical foundations of the considerable task faced by machine-learning algorithms in their attempts to formulate robust and practical answers to PS problems. We start by formulating the static PS problem as a single period investment strategy. The investor in this setting distributes his capital at the beginning of each trading period according to a portfolio vector $\mathbf{b}_t = (b_t^1, b_t^2, ..., b_t^m)$. The j^{th} component b_t^j of $\mathbf{b}_t$ denotes the proportion of the investor's capital invested in asset j at time t. Starting with an initial wealth $S_0 = 1$, if one uses a portfolio $\mathbf{b}$ and the stock vector is $\mathbf{x}$, the one-period wealth relative is $S = S_0 \sum_{j=1}^{m} b^j x^j = \mathbf{b}^\top \mathbf{x}$. Because S is a random variable, there is some controversy over the choice of the best distribution for maximising S. Investors are generally constrained to constant fractions of wealth allocated across the various assets and maximising the log-mean of portfolio wealth is the best criterion to use for the long-term investment. Such a portfolio is said to be log-optimal. We formulate the log-optimal investment problem in continuous and discrete time processes. The discrete time multi-period formulation leads to a convex optimisation problem that provides the basis for most online learning algorithms for PS problems. In fact, one can achieve an even higher growth rate for long-run investments if the tuning of the porfolio is allowed to change dynamically after each trading period.

1.5.2 Chapter III: Market Assumptions, Data Description and Performance Measurements

Before testing our algorithms with data from real financial markets we make some simplifying assumptions that are not found in real-markets. As in Gyorfi et al. (2008) we assume that assets are available in the desired quantities at a given price at any trading period. We also assumed that all trades are done at the closing price of that day. Transaction costs are taken into account and we assume a round-trip trading cost per trade of 10 basis points, to incorporate an estimate of price slippage and other costs as a single friction coefficient. Also, the set of assets involved is fixed: no new assets are allowed to be introduced in the market.

1.5.3 Chapter IV: Online Learning Algorithms for PS: An Empirical Survey

This chapter provides an empirical survey of some of the more promising online PS techniques. The algorithms surveyed have all demonstrated excellent finite sample performance using older datasets (see Table 1.2). Our main aim was to provide a timely survey, using more recent datasets, to assess the acclaimed robustness of some of the earlier published works in both machine-learning and data-mining fields. Table 1.2 is drawn from Li et al. (2012) and shows a performance comparison of some of the most promising PS algorithms. Our empirical survey will therefore focus exclusively on those algorithms that have shown the most growth in portfolio wealth.

Table 1.2: Performance Comparison of Selected PS Algorithms

Methods	NYSE (O)	NYSE (N)	DJA	TSE
Market	14.50	18.06	0.76	1.61
Best-stock	54.14	83.51	1.19	6.28
BCRP	250.60	120.32	1.24	6.78
UP	26.68	31.49	0.81	1.60
EG	27.09	31.00	0.81	1.59
ONS	109.19	21.59	1.53	1.62
B^K	1.08E+09	4.64E+03	0.68	1.62
B^{NN}	3.35E+11	6.80E+04	0.88	2.27
CORN	1.48E+13	5.37E+05	0.84	3.56
Anticor	2.41E+08	6.21E+06	2.29	39.36
PAMR	5.14E+15	1.25E+06	0.68	264.86
CWMR	6.49E+15	1.41E+06	0.68	332.62

1.5.4 Chapter V: Kalman Filtering and Online Learning Algorithms for PS

This chapter proposes new Online learning algorithms for PS based on an alternative measure of price relative called the Trend-Adjusted Price Relative (TAPR). The TAPR is derived from a simple state-space model of stock prices (using the Kalman Filter recursive algorithm) and we prove that the TAPR, unlike the standard raw price relative widely used in the machine literature, has well-defined and desirable statistical properties that make it better suited for nonparametric mean-reversion strategies. We find that the statistical evidence of out-of-sample predictability of stock returns is stronger once stock price trends are adjusted for high persistence.

1.5.5 Chapter VI: Sequential Portfolio Selection Algorithms Via Delay Coordinate Embedding

This chapter presents a novel non-universal, nonparametric statistical learning algorithm for PS problems based on Takens delay coordinate embedding theorem. Our delay coordinate embedding algorithm for PS (DCEPS) allows one to construct asymptotically log-optimal strategies for sequential investment in the financial market. We find that the statistical evidence of out-of-sample predictability of stock returns is stronger once stock prices are reconstructed in a phase-space coordinates. Our DCEPS algorithm is evaluated against some existing benchmark Online portfolio allocation techniques using six up-to-date real-market datasets. The DCEPS algorithm outperforms existing state-of-the-art allocation techniques, sometime in a spectacular way in these datasets without any additional computational demand or modelling complexity.

1.5.6 Chapter VII: Sequential Portfolio Selection Algorithms Via Multivariate Singular Spectrum Analysis

In recent years a powerful statistical technique known as Singular Spectrum Analysis (SSA) has been developed and successfully applied to many real-life problems in areas such as meteorology, oceanology, market research, medicine, economics and finance. Despite these successes, there is surprisingly no research to our knowledge that has applied this robust statistical technique to online PS algorithms.This chapter bridges this gap and presents a novel non-universal, nonparametric statistical learning algorithms for PS problems based on multivariate singular spectrum analysis. Our multivariate singular spectrum algorithm for PS (MSSAPS) allows one to construct asymptotically log optimal strategies for sequential investment in the financial market. Our approach is evaluated against benchmark Online portfolio allocation techniques using six up-to-date real-market datasets. Our methods outperform existing state-of-the-art portfolio allocation techniques, sometime in a spectacular way in these datasets.

1.5.7 Chapter VIII: Low-Regret Stock Selection in Online Learning Algorithms

In this chapter we propose a nonparametric empirical PS framework that explicitely separates the stock selection from the portfolio construction. Our new Online PS methodology follows three important steps that allow us to construct optimal investment strategies in the financial market. In the first step we use state-of-the-art Online, low-regret binary classification algorithms (inspired by the Support Vector Machine algorithm) to help in the stock selection process. The second step constructs the portfolio weight vector in the usual sense using suitable heuristics along the lines of algorithms proposed by Borodin et al. (2006). In the last step we combine the stock selection and portfolio construction outputs to derive our final portfolio weights. One of the striking features of our new algorithm is that we make no assumtpions about the statistical properties of stocks prices and the model itself requires very few paramter tunings. Extensive historical simulations demonstrate that this algorithm is indeed a very robust investment strategy.

1.5.8 Chapter IX: Conclusion Directions for Future Research

In Chapters IX we draw the main conclusion of our thesis and propose some directions for future research.

2.0 MATHEMATICAL FORMULATION

2.1 INTRODUCTION

This chapter lays down the theoretical foundations of the considerable task faced by machine-learning algorithms in their attempts to formulate innovative and practical answers to the PS problems.

Consider a market of m securities: these can be stocks, bonds, foreign currencies, or commodities. In order to apply Online learning algorithms to stock selection problems we consider a stock market model that has the same characteristics as the one investigated by, amongst others, Gyorfi et al. (2008) and Algoet and Cover (1988). We consider a market of m securities such that a market vector $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^m)$ represents the vector of prices for $j = 1, 2, ..., m$ securities. The change in security prices during the t^{th} trading period is represented as a stock market vector $\mathbf{x}_t = (x_t^1, x_t^2, ..., x_t^m) \in R_m^+$ where $\mathbf{x}_t$ is the vector m of non-negative numbers representing price relatives for the trading period t. The j^{th} component $x_t^j \geq 0$ of $\mathbf{x}_t$ expresses the ratio of two consecutive closing prices (from time $t-1$ to time t) of asset j such that $x_t^j = \frac{p_t^j}{p_{t-1}^j}$. Thus an investment of d dollars in the j^{th} security just before the start of the t^{th} trading period yields dx_t^j dollars by the end of the t^{th} trading period

2.2 STATIC PORTFOLIO SELECTION

The static PS is defined as a single-period investment strategy. The investor in this setting distributes his capital at the beginning of each trading period according to a portfolio vector

$\mathbf{b}_t = (b_t^1, b_t^2, ..., b_t^m)$. The j^{th} component b_t^j of $\mathbf{b}_t$ denotes the proportion of the investor's capital invested in asset j at time t. Throughout the thesis we assume that the portfolio manager is not permitted any short sale of securities; that is the portfolio vector is such that $\mathbf{b}_t \geq 0$ and that the portfolio manager is always fully invested $\left(\sum_{j=1}^{m} b^j = 1\right)$, including reinvestment of dividends. Starting with an initial wealth $S_0 = 1$, if one uses a portfolio $\mathbf{b}$ and the stock vector is $\mathbf{x}$, the one-period wealth relative is

$$S = S_0 \sum_{j=1}^{m} b^j x^j = S_0 \mathbf{b}^\top \mathbf{x}$$

We wish to maximise S in some sense. But S is a random variable, so there is controversy over the choice of the best distribution for S. The standard theory of stock market investment is based on the consideration of the first and second moments of S. The objective is to maximise the expected value of S, subject to a constraint on the variance. Since it is easy to calculate these moments, the theory is simpler than the theory that deals wth the entire distribution of S. In this setting, the investor is assumed to make allocation decisions once and for all at the beginning of a given period, based on estimated prospects for the risk and return relationships of a universe of m investable assets over the horison. This is essentially the mean-variance investment formulation as expressed by Markowitz (1952), which is widely used by investment analysts worldwide (see Section 2.3).

2.3 MEAN-VARIANCE SOLUTION TO THE INVESTMENT PROBLEM

Markowitz (1952) introduced the basic formulation of the mean-variance investment problem. In terms of the Markowitz (1952) formulation, the investor's optimal portfolio allocation is derived from a framework that includes expressions for the expected portfolio return and variance in terms of the portfolio weights and expected returns, variances and covariances of individual assets.

Let $\mathbf{R}_{t+1} \in \mathbb{R}^m$ be a vector of random expected asset returns between times t and $t+1$. Assume that the investor makes a forecast of the first two moments of the distribution of future returns given the information available at time t

$$\boldsymbol{\mu}_{t+1|t} = \mathbf{E}_t \left[\mathbf{R}_{t+1}\right] \tag{2.1}$$

$$\boldsymbol{\Sigma}_{t+1|t} = \mathbf{Cov}_t \left[\mathbf{R}_{t+1}\right] \tag{2.2}$$

where $\mathbf{E}_t [.]$ and $\mathbf{Cov}_t [.]$ respectively denote the expectation and covariance matrix of a random variable conditioned on the information available at time t. For simplicity in this section, since single-period modelling does not explicitly consider the consequences of time, we drop the time subscripts on the above quantities, which we write simply as $\mathbf{R}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Likewise, the return on the risk-free asset during the period is denoted by $\mathbf{r}_f$.

Investors allocate their capital among the m assets, forming a portfolio $\mathbf{w} \in \mathbb{R}^m$ where each element w_i, the weight of asset i, represents the fraction of total capital held in the asset. The expected portfolio return and variance are given respectively by:

$$\boldsymbol{\mu}(\mathbf{w}) = \mathbf{E}\left[\mathbf{R}(\mathbf{w})\right] = \mathbf{E}\left(\mathbf{w}^\top \mathbf{R}\right) = \mathbf{w}^\top \mathbf{E}(\mathbf{R}) = \mathbf{w}^\top \boldsymbol{\mu} \tag{2.3}$$

$$\begin{aligned} \boldsymbol{\sigma}^2(\mathbf{w}) &= \mathbf{E}\left[(\mathbf{R}(\mathbf{w}) - \boldsymbol{\mu}(\mathbf{w}))(\mathbf{R}(\mathbf{w}) - \boldsymbol{\mu}(\mathbf{w}))^\top\right] \\ &= \mathbf{E}\left[\left(\mathbf{w}^\top \mathbf{R} - \mathbf{w}^\top \boldsymbol{\mu}\right)\left(\mathbf{w}^\top \mathbf{R} - \mathbf{w}^\top \boldsymbol{\mu}\right)^\top\right] \\ &= \mathbf{E}\left[\mathbf{w}^\top (\mathbf{R} - \boldsymbol{\mu})(\mathbf{R} - \boldsymbol{\mu})^\top \mathbf{w}\right] \\ &= \mathbf{w}^\top \mathbf{E}\left[(\mathbf{R} - \boldsymbol{\mu})(\mathbf{R} - \boldsymbol{\mu})^\top\right] \mathbf{w} \\ &= \mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w} \end{aligned} \tag{2.4}$$

2.3.1 Optimisation Problem

An investor's optimisation problem can therefore be formulated as one of maximising the expected returns of the portfolio under a volatility constraint or, equivalently, minimising the volatility of the portfolio under a return constraint. For ease of numerical computations, Markowitz (1956) transformed this nonlinear optimisation problem into a quadratic optimisation problem which can be formulated as follows:

$$\begin{aligned} \mathbf{w}^*(\lambda) &= \arg\max \mathbf{w}^\top \boldsymbol{\mu} - \tfrac{\lambda}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w} \\ s.t &\quad \mathbf{1}^\top \mathbf{w} = 1 \end{aligned} \tag{2.5}$$

or equivalently,

$$\begin{aligned} \mathbf{w}^*(\varphi) &= \arg\max \tfrac{1}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w} - \tfrac{\varphi}{2}\mathbf{w}^\top \boldsymbol{\mu} | \boldsymbol{\varphi} = \boldsymbol{\lambda}^{-1} \\ s.t &\quad \mathbf{1}^\top \mathbf{w} = 1 \end{aligned} \tag{2.6}$$

where λ and φ represent risk aversion parameters. From a numerical point of view, the second formulation has the advantage of being a standard quadratic programming problem and therefore is the one mostly used by advocates of the mean-variance framework. Of course if $\varphi = 0$ the problem reduces to one of finding the minimum variance portfolio and if $\varphi = \infty$, the investor problem is that of maximising the portfolio returns.

2.3.2 Analytical Solution

The first version of the optimisation problem in Equation 2.5 could be easily solved using the method of Lagrange multiplier. The Lagrange function of the optimisation problem is:

$$L(w; \lambda_0) = \mathbf{w}^\top \boldsymbol{\mu} - \frac{\lambda}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w} + \boldsymbol{\lambda}_0\left(\mathbf{1}^\top \mathbf{w} - \mathbf{1}\right) \tag{2.7}$$

The method of Lagrange multiplier works by taking the partial derivatives of equation 2.7 with respect to unknown variables ($\mathbf{w}$ and λ_0) and setting them to zero. The solution $\mathbf{w}^*$ therefore verifies the following first order conditions:

$$\begin{cases} \partial_{\mathbf{w}} L(w; \lambda_0) = \boldsymbol{\mu} - \lambda \boldsymbol{\Sigma}\mathbf{w} + \boldsymbol{\lambda}_0 \mathbf{1} = \mathbf{0} \\ \partial_{\boldsymbol{\lambda}_0} L(w; \lambda_0) = \mathbf{1}^\top \mathbf{w} - 1 = 0 \end{cases} \tag{2.8}$$

The first partial derivative in Equation 2.7 gives:

$$\mathbf{w} = \boldsymbol{\lambda}^{-1}\Sigma^{-1}(\boldsymbol{\mu} + \boldsymbol{\lambda}_0 \mathbf{1})$$

Substituting this last expression into the second part of Equation 2.8 gives:

$$\mathbf{1}^\top\left(\boldsymbol{\lambda}^{-1}\Sigma^{-1}\left(\boldsymbol{\mu}+\boldsymbol{\lambda}_0\mathbf{1}\right)\right)-1=0 \text{ or } \lambda_0=\frac{1-\mathbf{1}^\top\boldsymbol{\lambda}^{-1}\Sigma^{-1}\boldsymbol{\mu}}{\mathbf{1}^\top\boldsymbol{\lambda}^{-1}\Sigma^{-1}\mathbf{1}}$$

This leads to the final solution given by:

$$\mathbf{w}^*=\frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top\Sigma^{-1}\mathbf{1}}+\frac{1}{\lambda}\times\frac{\left(\mathbf{1}^\top\Sigma^{-1}\mathbf{1}\right)\Sigma^{-1}\boldsymbol{\mu}-\left(\mathbf{1}^\top\Sigma^{-1}\boldsymbol{\mu}\right)\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top\Sigma^{-1}\mathbf{1}} \tag{2.9}$$

From equation 2.9 we can deduce the global minimum variance as the following:

$$\mathbf{w}_{mv}=\mathbf{w}^*(\infty)=\frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top\Sigma^{-1}\mathbf{1}} \tag{2.10}$$

One major practical difficulty when implementing this framework is to precisely define the vector of expected returns of the risky assets and the corresponding expected covariance matrix of asset returns. As many financial professionals have experienced, using sample mean and covariance is simply not a good alternative in this case as the optimised portfolios are generally very sensitive to these inputs. In fact, using historical returns to estimate parameters that can be used as inputs to obtain the set of efficient portfolios depends on whether the underlying economies giving rise to the observed outcomes of returns are strong and stable. This, in general, is a very difficult statement to prove.

2.4 THE LOG-OPTIMAL PORTFOLIO SOLUTION

In Section 2.3 we reviewed some of the basic building blocs of the mean-variance framework for PS problems. Although the theory behind mean-variance optimisation is relatively straightforward, many finanical professionals have found its practical implementation very difficult to achieve. The theory dictates that given estimates of the returns, volatilities, and correlations of a set of investments and constraints on investment choices (for example, maximum exposures and turnover constraints), it is possible to perform an optimisation that results in the risk/return or mean-variance efficient frontier However, the theory remains silent on how to obtain these forward estimates of returns, volatilities and correlations making it of little usage for practical applications.

2.4.1 The Log-Optimal Portfolio in Continuous Time

One of the portfolio manager's main aim in PS is to obtain an optimal wealth growth. It is now well agreed that when the investment is constrained to have constant fractions of wealth allocated across the assets, then maximising the log-mean of portfolio wealth is the best criterion to use for long-term investment (see Gyorfi et al. (2007)). Such a portfolio is said to be log-optimal. These portfolios were introduced in Latane (1959) and Kelly (1956) for the case of discrete-time static portfolios and were more fully developed in Breiman (1961). Log-optimal portfolios in continuous-time dynamic case, with constraints and transaction cost can be found in textbooks such as Korn (1997).

Let's consider a market consisting of a constant single risk-free asset which has the following mathematical representation

$$dS\left(t\right) = rS\left(t\right)dt \text{ or } \mathbf{S}\left(t\right) = S\left(0\right)e^{\int_0^t r(u)du} \tag{2.11}$$

and m risky assets with the following geometric brownian motion equations

$$dS_i\left(t\right) = S_i\left(t\right)\left[\mu_i dt + \sum_{j=1}^{m} \sigma_{ij} dW_j\left(t\right)\right] \tag{2.12}$$

and $S_i\left(0\right) > 0$, $i = 0, 1, ..., m$. The interest rate r, the drift μ_i and volatility σ_{ij}, are all assumed to be constants and positive. The noise terms $dW_i\left(t\right)$ are differentials of independent standard Brownian motions and the volatility matrix σ, given by $\sigma = \left\{\sigma_{ij}\right\}_{i,j=1,}^{m}$ is assumed to be nonsingular. This is essentially the same setting as the one adopted by Fernholz (2002).

The trading strategy is defined as an adapted real-valued process

$$\left[v_0\left(t\right), v_1\left(t\right), ..., v_m\left(t\right)\right]^\top$$

where $v_i\left(t\right)$ represents the number of shares per asset. The portfolio value, i.e. the total potfolio wealth, at time t is given by

$$y\left(t\right) = \sum_{i=0}^{m} v_i\left(t\right) S_i\left(t\right) = \sum_{i=0}^{m} y_i\left(t\right) \tag{2.13}$$

Here $y_i(t), i = 0, 1, ..., m$, denotes the value of the holdings per asset. These can also be expressed in terms of the fraction $\alpha_i(t)$ of $y_i(t)$ allocated to asset i, as follows

$$y_i(t) = v_i(t) S_i(t) = \alpha_i(t) y(t), i = 0, 1, 2, ..., m \tag{2.14}$$

$$\sum_{i=0}^{m} \alpha_i(t) = 1 \tag{2.15}$$

A portfolio is self-financing if the changes in its value occurs only due to price changes, and is described by

$$dy(t) = \sum_{i=0}^{m} v_i(t) dS_i(t) \tag{2.16}$$

A well known result in stochastic portfolio theory (see Fernholz (2002)) it that

$$dy(t) \quad = \quad y(t)\left[\alpha_0 r dt + \sum_{i=0}^{m} \alpha_i \left(\mu_i dt + \sum_{j=1}^{m} \sigma_{ij} dW_j(t)\right)\right] \tag{2.17}$$

This is obtained by simply substituting (2.11) and (2.12) in (2.16) and making use of the relations (2.14) and (2.15).

Equation 2.17 is equivalent to

$$dy(t) \quad = \quad y(t)\left[rdt \sum_{i=0}^{m} \alpha_i (\mu_i - r) dt + \sum_{j=1}^{m} \left(\sum_{i=1}^{m} \alpha_i \sigma_{ij}\right) dW_j(t)\right] \tag{2.18}$$

The portfolio value (2.18) is a controlled stochastic process with fractions of wealth $\alpha_i(t), i = 0, 1, ..., m$, as control variables. Assuming these to be constant over time, the optimal growth of $y(t)$ over the long run is achieved by maximising its log mean at some instant of time t. Following the presentation in Luenberger (1998), we first derive the dynamics of $ln(y(t))$ from (2.18) using Ito's lemma as

$$d\ln(y(t)) \quad = \quad \left[r + \sum_{i=1}^{m} \alpha_i (\mu_i - r) - \tfrac{1}{2} \sum_{j=1}^{m} \left(\sum_{i=1}^{m} \alpha_i \sigma_{ij}\right)^2\right] dt + \sum_{i=1}^{m} \alpha_i \sum_{i=1}^{m} \sigma_{ij} dW_j(t) \tag{2.19}$$

$$\max E\left[\ln\frac{y(t)}{y(0)}\right] \tag{2.20}$$

is achieved for some t (or equivalently, for every $t > 0$), subject to (2.19). After integrating (2.19) and taking the expectation of the result, one obtains

$$E\left[\ln\frac{y(t)}{y(0)}\right] = \left[r + \sum_{i=1}^{m}\alpha_i(\mu_i - r) - \frac{1}{2}\sum_{j=1}^{m}\left(\sum_{i=1}^{m}\alpha_i\sigma_{ij}\right)^2\right]t \tag{2.21}$$

This shows that log-optimal fractions $\alpha_i(t), i = 0, 1, ..., m$, solve the following problem:

$$\max\left[r + \sum_{i=1}^{m}\alpha_i(\mu_i - r) - \frac{1}{2}\sum_{j=1}^{m}\left(\sum_{i=1}^{m}\alpha_i\sigma_{ij}\right)^2\right] \tag{2.22}$$

$$\Delta\ln(y(k)) = \left[r + \sum_{i=1}^{m}\alpha_i(\mu_i - r) - \tfrac{1}{2}\sum_{j=1}^{m}\left(\sum_{i=1}^{m}\alpha_i\sigma_{ij}\right)^2\right]T + \sum_{i=1}^{m}\alpha_i\sum_{i=1}^{m}\sigma_{ij}\sqrt{T}e_j(k) \tag{2.23}$$

where $e_j(k)$, $j = 1, 2, ..., n$, are gaussian i.i.d. random variables with zero mean and variance one. It is clear from (2.23) that the values of α_i, $i = 1, 2, ..., n$, such that

$$\max E\left[\Delta\ln(y(k))\right] \tag{2.24}$$

or equivalently

$$\max E\left[\Delta\ln(y(k+1))\right] \tag{2.25}$$

is achieved for every k, are the log-optimal ones. This means that optimisation problems (2.20), (2.24), and (2.25), all lead to solving (2.22).

2.4.2 The Log-Optimal Portfolio in Discrete Time

A general goal in PS problems is that of deriving an optimal growth in portfoli wealth. It is a now well established result that when the investment is constrained to have a constant fractions of wealth allocated across the assets, then maximising the log-mean of portfolio wealth is the best criterion to use for the long-term investment (see Latane (1959), Kelly (1956), Breiman (1961) and Gyorfi et al. (2007)). To fully explore this fact and its various implications, let us once more consider a market of m stock prices such that a market price vector $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^m)$ represents the vector of prices for $j = 1, 2, ..., m$.

In particular we consider a stock market model that has the same characteristics as the one investigated by amongst others, Gyorfi et al. (2007) and Algoet (1992).

We consider a market of m securities that has the same characteristics as the market investigated by Gyorfi et al. (2006,2007,2008) and Algoet (1996). In this setting, the market vector $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^m)$ represents the vector of prices for $j = 1, 2, ..., m$ stocks. The change in security prices during the t^{th} trading period is represented as a stock market vector $\mathbf{x}_t$ $= (x_t^1, x_t^2, ..., x_t^m) \in \mathbf{R}_m^+$ where $\mathbf{x}_t$ is the vector m of non-negative numbers representing price relatives for the trading period t. The j^{th} component $x_t^j \geq 0$ of $\mathbf{x}_t$ expresses the ratio of two consecutive closing prices of asset j such that $x_t^j = \frac{p_t^j}{p_{t-1}^j}$. Thus an investment of d dollars in the j^{th} security just before the start of the t^{th} trading period yields dx_t^j dollars by the end of the t^{th} trading period

The investor in our model is allowed to distribute his or her capital at the beginning of each trading period according to a portfolio vector $\mathbf{b}_t = (b_t^1, b_t^2, ..., b_t^m)$. Here the j^{th} component b_t^j of $\mathbf{b}_t$ denotes the proportion of the investor's capital invested in asset j at time t. Throughout this chapter we assume that the portfolio manager is not allowed any short sale of securities, meaning that the portfolio vector is such that $\mathbf{b}_t \geq 0$ and that the portfolio manager is always fully invested $\left(\sum_{j=1}^{m} b_t^j = 1 \right)$, including reinvestment of dividends.

Let S_0 denotes the investor's initial capital. A PS algorithm is a mechanistic procedure that produces any sequence of portfolios $\mathbf{b}_n = (\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n)$ by specifying how to reinvest the current wealth from trading period to trading period. Starting with an initial wealth S_0, after n trading periods, the investment strategy $\mathbf{B}$ achieves the wealth

$$S_n = S_0 \prod_{t=1}^{n} \sum_{j=1}^{m} b_t^j \left(\mathbf{x}_1^{t-1} \right) x_t^j = S_0 \exp \left\{ \sum_{i=1}^{n} \log \mathbf{b}_t^{\intercal} \left(\mathbf{x}_1^{t-1} \right) \mathbf{x}_t \right\} \tag{2.26}$$

this may be written as

$$S_n = S_0 \exp\left\{n\mathbf{W}_n\left(\mathbf{B}\right)\right\} \tag{2.27}$$

where $\mathbf{W}_n\left(\mathbf{B}\right)$ denotes the average growth rate and is given by

$$\mathbf{W}_n\left(\mathbf{B}\right) = \frac{1}{n}\sum_{i=1}^{n} \log \mathbf{b}_t^{\intercal}\left(\mathbf{x}_1^{t-1}\right)\mathbf{x}_t. \tag{2.28}$$

Given that the goal is to maximise $S_n = S_n\left(\mathbf{B}\right)$, this is equivalent to maximising the following average growth rate

$$\mathbf{W}_n\left(\mathbf{B}\right) : \mathbf{b}_i^{*}\left(\mathbf{x}_1^{i-1}\right) = \arg_b \max \mathbf{E}\left\{\log \mathbf{b}_t^{\intercal}\left(\mathbf{x}_1^{t-1}\right)\mathbf{x}_t \mid \mathbf{x}_1^{i-1}\right\} \tag{2.29}$$

It is important to notice that the PS problem in this setup is only a crude approximation of the corresponding real-life problem, as it includes no transaction costs and assumes that money and units of securities are arbitrarily divisible. Although there are many more assumptions that could be made in order to make the model more realistic, we nevertheless think that this model is rich enough to form the basis for studying some of the essential questions related to PS.

2.5 SOME BENCHMARK PORTFOLIO SELECTION ALGORITHMS

The complexity of equation (2.30) makes it very hard to find closed-form solutions for the PS problem unless some structure is imposed a priori on the evolution of the portfolio weights. This lack of close-form solution explains why market practitioners have adopted some rather simplistic but intuitive PS rules in an attempt to derive optimal portfolio weights. The most basic of PS algorithms is the so called Buy and Hold (BAH_b) which buys stocks using some constant portfolio b. This algorithm invests according to b on the first trading day, and then never re-invests any money after that. The proportion of capital invested in stock j at time t is given by

$$\widetilde{b}_{t+1}^{j} = \frac{x_t^j b_t^j}{\sum_{i=1}^{m} b_t^i x_t^i}. \tag{2.30}$$

In theory it is not difficult to imagine the existence of an optimal BAH_{b^*} strategy that can only be achieved "in hindsight". BAH_{b^*} is simply a portfolio b that assigns a weight of 1 to the "best" stock, and a weight of 0 to all others. Mathematically this is expressed as

$$b^* = \underset{b(.)}{\arg\max}\, ret_x\left(BAH_b\right) \tag{2.31}$$

where $ret_x\left(BAH_b\right)$ is simply the returns achieved by the Buy-and-Hold given the sequence of price relative x. When b is set such that the total available investment is initially equally distributed amongst various assets $b = \left(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}\right)$ the BAH_b is referred to as the uniform Buy-and-Hold or $UBAH_b$

The BAH_b strategy possesses at least two very attractive features. First, BAH_b incurs no additional transaction costs once the initial trade allocation has been placed in the market unless the portfolio manager decides to make changes to his holdings. Second, the Buy-and-Hold strategy does not suffer from any market impact or other stock market frictions, as it requires no rebalancing. This, to some extent, explains why this strategy has been so popularised amongst fund managers globally. However, the major drawback of the Buy-and-Hold strategiy is that it performs well only when the overall market performs well. Recent history typified by the global financial crunch of 2008 has demonstrated that severe market corrections do occur with reasonable frequency and that when this happens a naive Buy-and-Hold strategy can suffer precipitous losses.

An alternative approach to the static Buy-and-Hold is to dynamically change the portfolio during the trading period. In this case the algorithm maintains a fixed portfolio b throughout the entire trading period by appropriately (actively) re-investing money at the end of each trading day. One example of active trading is constant rebalancing; namely, fix a portfolio b and (re)invest capital each day according to b. We denote this constant rebalancing strategy by $CBAL_b$ and let $CBAL^*$ denote the optimal (in hindsight)

$$CBAL_{b^*} = \underset{b(.)}{\arg\max}\, ret_x\left(CBAL_b\right) \tag{2.32}$$

A major benefit of the constant rebalancing strategy lies in its ability to take advantage of market fluctuations to achieve returns that are sometimes significantly greater than those of BAH^*, although this might come at the expense of much higher transactions costs. $CBAL^*$ is always at least as good as the best stock and BAH^*, that is $ret_x\left(CBAL_b^*\right) \geq ret_x\left(BAH_b^*\right)$ and in some real-market sequences, a constant rebalancing strategy will take advantage of market fluctuations and significantly outperform the best stock. When b is set such that the total available investment is initially equally distributed amongst various assets $b = \left(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}\right)$ the $CBAL_b$ is referred to as the "uniform Buy-and-Hold or $UCBAL_b$".

3.0 MARKET ASSUMPTIONS, DATA DESCRIPTION AND PERFORMANCE MEASUREMENTS

3.1 INTRODUCTION

Before testing our algorithms with data from real financial markets we make some simplifying assumptions that are not found in real-markets.

3.2 ANOTHER LOOK AT THE MATHEMATICAL FORMULATION

Let S_0 denotes the investor's initial capital. Starting with an initial wealth S_0, after n trading periods, we showed in section 2.4.2 that the investment strategy $\mathbf{B}$ achieves the wealth

$$S_n = S_0 \prod_{t=1}^{n} \sum_{j=1}^{m} b_t^j \left(\mathbf{x}_1^{t-1}\right) x_t^j = S_0 \exp\left\{\sum_{i=1}^{n} \log \mathbf{b}_t^{\intercal} \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t\right\}$$

this may be written as

$$S_n = S_0 e^{n\mathbf{W}_n(\mathbf{B})}$$

where $\mathbf{W}_n(\mathbf{B})$ denotes the average growth rate and is given by

$$\mathbf{W}_n(\mathbf{B}) = \frac{1}{n} \sum_{i=1}^{n} \log \mathbf{b}_t^{\intercal} \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t.$$

Given that the goal is to maximise $S_n = S_n(\mathbf{B})$, this is equivalent to maximising the following average growth rate

$$\mathbf{W}_n(\mathbf{B}) : \mathbf{b}_i^*\left(\mathbf{x}_1^{i-1}\right) = \arg_b \max \mathbf{E}\left\{\log \mathbf{b}_t^{\intercal}\left(\mathbf{x}_1^{t-1}\right)\mathbf{x}_t \mid \mathbf{x}_1^{i-1}\right\}$$

The fundamental limits, determined in Algoet and Cover (1988), reveal that the so-called log-optimum portfolio $B^* = \{b^*(.)\}$is the best possible choice.

If $S_n^* = S_n(B^*)$ denotes the capital achieved by a log-optimum portfolio strategy B^*, after n trading periods, then for any other investment strategy B with capital $S_n = S_n(B)$ and for any stationary and ergodic process $\{X_n\}_{-\infty}^{\infty}$,

$$\lim_{n\longrightarrow\infty} \sup \frac{1}{n} \log \frac{S_n}{S_n^*} \leq 0 \text{ almost surely}$$

and

$$\lim_{n\longrightarrow\infty} \frac{1}{n} \log S_n^* = W^* \text{almost surely}$$

where

$$W^* = \mathbf{E}\left\{\log\left\langle \mathbf{b}^*\left(\mathbf{X}_{-\infty}^{-1}\right), \mathbf{X}_0\right\rangle\right\}$$

is the maximal possible growth rate of any investment strategy. Thus (almost surely), no investment strategy can have a faster rate of growth or cumulative annual growth rate (CAGR) than W^*. Of course, to determine a log-optimal portfolio, full knowledge of the (infinitedimensional) distribution of the process is required.

3.3 MARKET ASSUMPTIONS

As in Gyorfi et al. (2008) we assume that assets are available in the desired quantities at a given price at any trading period. We also assumed that all trades are done at the closing price of a given day. Of course in real-markets application, investors are likely to buy at the "bid" and sell at the "ask" resulting in a spread between the bid and the ask called the "bid-ask spread". This is essentially the difference in price between the highest price that a buyer is willing to pay for an asset and the lowest price for which a seller is willing to sell.

Another assumption we make is that all the wealth achieved in the previous period is fully invested in the next one, without any extra investment allowed. Transaction costs are taken into account and we assume a base case round-trip trading cost per trade of 10 basis points, to incorporate an estimate of price slippage and other costs as a single-friction coefficient. Also, the set of assets involved is fixed; no new assets are allowed to be introduced in the market. Another implicit assumption is that prices are not affected by our actions on the market. Clearly this is not a realistic assumption since we are trading an enormous amount of asset values, not negligible even in comparison with the full market. Finally, we adjust all stock prices for dividend payments and stock splits.

3.4 DATA DESCRIPTION

Our empirical experiments entail performing numerical evaluations on six real datasets in order to compare the performance of our proposed algorithms with some existing benchmark methods. The first four datasets summarised in Table 3.1 are obtained from Bloomberg and cover the period of January 2000 to December 2013. Bloomberg stock prices are adjusted for stock splits, dividends and are given in local currencies. To our knowledge this datasets has never been tested and possesses the advantage that it covers the subprime crisis period of 2007 to 2009. This datasets covers about 4508 trading days that exclude weekends and public holidays for the respective countries.

The first datasets is the TOP40 Index which consists of the largest 42 stocks by market capitalisation listed on the Johannesburg Stock Exchange in South Africa. This datasets contains price relatives of 4508 trading days, ranging from January 2000 to October 2013 and is by far the most liquid index available to the South African investor. The second datasets, FTSE100 Index, is a share index of the 100 companies listed on the London Stock Exchange with the highest market capitalisation. It is one of the most widely used stock indices globally and is seen as a gauge of business prosperity in the United Kingdom. The third datasets is that of the Toronto Stock Exchange referred to as the S&P/TSE 60 or simply the TSE60 Index. This index represents the stock market index of 60 large companies listed on the Toronto Stock Exchange and it currently exposes the investors to about 10 industry sectors. The fourth datasets is derived from the NASDAQ stock exchange. The NASDAQ is a stock market index of 100 of the largest non-financial companies listed on the NASDAQ. It is a modified capitalisation-weighted index. The companies' weights in the index are based on their market capitalisations, with certain rules capping the influence of the largest components. This index does not contain financial companies, and could include companies incorporated outside the United States.

Table 3.1: Data Description

Data Set	Region	Time Frame	Trade Days	Stocks
TOP40	SA	JAN-2000 - OCT-2013	4508	42
FTSE100	UK	JAN-2000 - OCT-2013	4508	100
TSE60	CANADA	JAN-2000 - OCT-2013	4508	60
NASDAQ	US	JAN-2000 - OCT-2013	4508	100
NYSE(O)	US	JUL-1962 - DEC-1984	5651	36
NYSE(N)	US	JAN-1985 - JUN-2009	6179	23

The last two datasets, NYSE (O) and NYSE (N), are from the New York Stock Exchange in the US. These NYSE datasets have been widely analyzed in many previous studies (Cover (1991); Helmbold et al. (1998); Borodin et al. (2004); Agarwal et al. (2006); Gyorfi et al. (2006; 2008)) and recently by Li et al (2012). Although not very useful for practical applications, these datasets provide a benchmark against which one can compare historical simulation and testing of empirical data across many previously published state-of-the-art Online PS strategies. The NYSE (O) for example contains 5651 daily price relatives of 36 stocks in the New York Stock Exchange (NYSE) for a 22-year period from July 3rd 1962 to December 31st 1984. The NYSE(N) datasets is the one used by Li et al. (2012) that covers the period from January 1st 1985 to June 30th 2009 and contains 6179 trading days and 23 stocks.

3.5 PERFORMANCE MEASURES AND STRATEGY EVALUATION

This section provides a review of the methods for measuring portfolio performance and the evidence on the performance of professionally managed investment portfolios. While the literature goes back to before the 1960s, recent years have witnessed an explosion of new methods for performance evaluation and new evidence on the subject. We think that several forces have contributed to this renaissance. Early studies frequently attempt to distinguish security selection versus market-timing abilities on the part of fund managers. Timing ability is the ability to use superior information about the future realisations of common factors that affect overall market returns. A PS strategy with timing ability may alter the asset allocation between available stock prices just before significant moves. Selectivity refers to the use of security-specific information, such as the ability to pick winning stocks or bonds within an asset class.

The main idea in most of the classical measures of investment performance is quite simple. The measures essentially compare the return of a managed portfolio over some evaluation period to the return of a benchmark portfolio. The benchmark portfolio should represent a feasible investment alternative to the managed portfolio being evaluated. If the objective is to evaluate the investment ability of the PS strategy, as has typically been the case, the benchmark should represent an investment alternative that is equivalent in all return-relevant aspects to the managed portfolio being evaluated, except that it should not reflect the investment ability of the manager. With this in mind, we propose the following simple yet effective measures of performance to assess the proposed PS algorithms.

3.5.1 Jensen's Alpha

Alpha is perhaps the most well-known of the classical measures of investment performance. Using the market portfolio of the CAPM to form the market benchmark, Jensen (1968) advocated the original version, or Jensen's alpha. The most convenient way to define Jensen's alphas is as the intercept, α^J, in the following time series regression:

$$r_t = \alpha^J + \beta r_{m,t} + \varepsilon_t \tag{3.1}$$

where r_t is the strategy returns at time t and $r_{m,t}$ represents the market returns at time t. α^J, β and ε_t represent the estimated intercept, the estimated slope of the regression and the error term in that order.

This version of Jensen's alpha remains the most popular performance measures in academic studies. However, the measure has some disadvantages. For example, alpha does not control for nonsystematic sources of risk that could matter to investors (e.g., Fama, 1972).

3.5.2 Treynor–Mazuy Market-Timing Model

Classical models of market-timing use convexity in the relation between the strategy's return and the "market" return to indicate timing ability. In these models the manager observes a private signal about the future performance of the market and adjusts the market exposure or beta of the portfolio at the beginning of the period. Successful timing implies higher betas when the market subsequently goes up, or lower betas when it goes down, leading to the convex relation. The Treynor–Mazuy (1966) market-timing model is a quadratic regression:

$$r_t = \alpha + \beta r_{m,t} + \gamma r_{m,t}^2 + \varepsilon_t \tag{3.2}$$

where as usual r_t is the strategy returns at time t and $r_{m,t}$ represents the market returns at time t. In this equation α represent the intercept, β shows the average sensitivity of the strategy returns to the movements in the market returns γ is the convexity estimate. ε_t is the residual term in this regression that is assumed to be white noise.

Treynor and Mazuy (1966) argue that $\gamma > 0$ indicates market-timing ability. When the market is up the strategy will be up by a disproportionate amount and when the market is down, the fund will be down by a lesser amount. Admati et al. (1986) formalised the model, showing how it can be derived from a timer's optimal portfolio weight, assuming normal distributions and managers with exponential utility functions. These authors show that the timing coefficient γ is proportional to the product of the manager's risk tolerance and the precision of the signal about the future market returns.

3.5.3 Asymmetric Beta

A similar and widely used risk management tool that describes the risk of our strategies with respect to the risk of the overall market is called Asymmetric Beta. Asymmetric Beta measures the slope of the regression (the co-movement) between the market index and the strategy for a given market mode. We define the bull and bear betas by the following equations:

$$\beta_{bull} = \frac{Cov\left(r, r_m\right)}{Var\left(r_m\right)} \left| r_m \geq 0 \right. \tag{3.3}$$

$$\beta_{bear} = \frac{Cov\left(r, r_m\right)}{Var\left(r_m\right)} \left| r_m < 0 \right. \tag{3.4}$$

The intuition behind the bull and bear betas is straightforward. In bull markets, the strategy returns should go up faster than the market and in bear markets, they should fall less than the market, not at all, or even go up in price. For a strategy with good market-timing abilities we would expect $\beta_{bull} \gg \beta_{bear}$. It should be expected that measures of β_{bull} and β_{bear} will give very similar results to those of Treynor–Mazuy market-timing model.

3.5.4 Total Portfolio Cumulative Wealth

Another criterion we use to evaluate the performance of our proposed strategy is the total portfolio cumulative wealth, S_n, achieved by the strategy until the end of the whole trading period n. Starting from a wealth of S_0 we can calculate S_n as:

$$\mathbf{S}_n = \mathbf{S}_0 \prod_{i=1}^{n} \left(1 + r_i\right) \tag{3.5}$$

where r_t represents the returns achieved by the portfolio on trading period t. Of course in a risk-neutral world the portfolio manager will aim at maximising the terminal wealth as this is directly linked to the investor's utility.

3.5.5 Annualised Percentage Yield

An equivalent criterion is annualised percentage yield (AY) which is defined as the effective annual rate of return, taking into account the effect of compounding interest. The AY can be calculated as

$$AY = \left(1 + \frac{\mathbf{S}_n}{\mathbf{S}_0}\right)^{(252/n)} - 1 \tag{3.6}$$

In this expression we assume that there are 252 trading days in a calendar year. AY measures the average wealth increment that one strategy could achieve compounded in a year. Typically, the higher the value of portfolio cumulative wealth or annualised percentage yield, the more preferable the trading strategy is.

3.5.6 Sharpe Ratio

One important way to evaluate risk-adjusted returns in a portfolio is simply to calculate the ratio between the annualised percentage yield (AY) after subtracting the risk free rate r_f and the annualised standard deviation of daily returns to measure the volatility risk. This is generally referred to as the Sharpe Ratio (SR) and measures the risk-adjusted returns achieved by a strategy. We calculate the SR as

$$SR = \frac{AY}{STD} \text{ where } STD = \sqrt{\sum_{i=1}^{n} (r_i - \mu)^2 \, (252)} \tag{3.7}$$

Higher Sharpe Ratios indicate better risk-adjusted returns performance of a trading strategy. The Sharpe Ratio may also be inappropriate when returns are highly nonnormal. For example, Leland (1999) shows that it is important to consider higher moments of the distributions if the performance measure is to accurately capture an investor's utility function. Furthermore, if the returns distributions are highly skewed, such as when options may be traded, the Sharpe Ratio can be misleading

3.5.7 Maximum Draw Down (MDD)

Another important measure of the risk embedded in the PS algorithm is the maximum cumulative loss from a market peak to the following trough, often called the "maximum drawdown (MDD)". The MDD is a measure of how sustained one's losses can be. Large drawdowns usually lead to fund redemptions, and so the MDD is the risk measure of choice for many money management professionals and a reasonably low MDD is critical to the success of any fund. Formally, let S_n denote the maximum cumulative wealth achieved by a trading strategy from the beginning to time n. The Maximum Draw Down (MDD) at any time n, is defined as

$$MDD_n = \max\left[0, \max_{1\leq i\leq n}\left(\frac{\mathbf{S}_i}{\mathbf{S}_n} - 1\right)\right] \tag{3.8}$$

This is an excellent way to measure the downside risk of different strategies.

3.5.8 Percentage of Profitable Trades

We define three other measures of risk that give an indication of how our strategy performs in various market conditions. The first one is the percentage of time a strategy actually generates a profit irrespective of the general market movements. We call this performance measure PP and calculate it from the indicator function I as follows:

$$PP = \frac{\sum_{i=1}^{n} I_{r_i>0}}{n} \tag{3.9}$$

Where I is an indicator function taking the value of 1 when the strategy returns was positive and zero otherwise. In a similar way we can calculate the percentage of time the strategy is profitable in rising markets (PPUM) and in falling markets (PPDM) as follows:

$$PPUM = \frac{\displaystyle\sum_{i=1}^{n} I_{\{r_{\mathrm{Si}}>0 \text{ and } r_{Mi}>0\}}}{\displaystyle\sum_{i=1}^{n} I_{r_{Mi}>0}} \text{ and } PPDM = \frac{\displaystyle\sum_{i=1}^{n} I_{\{r_{\mathrm{Si}}>0 \text{ and } r_{Mi}<0\}}}{\displaystyle\sum_{i=1}^{n} I_{r_{Mi}<0}} \tag{3.10}$$

where r_{S_t} is the strategy returns and r_{M_t} is the market returns in period t. It can be easily demonstrated that for the market index $PPUM = 1$ and $PPDM = 0$. These two performance measures are particularly important in so far as they show how the strategy behaves in both bull and bear markets.

3.6 COMBINING STRATEGIES

Online learning algorithms for PS problems in general have few critical parameters that need to be set/estimated by the portfolio manager. Unfortunately, the performance of these algorithms depends significantly on the optimal choice of these parameters and this choice is rather a difficult one and cannot be done with the benefit of hindsight. In traditional quantitative modelling, the optimal choice is generally performed via the in sample-training and out-of-sample testing paradigm. Of course there is nothing special about this traditional optimal parameter choice procedure, as nothing guarantees that parameters that are optimal in the training sample will be so in the more important test sample. Perhaps, a better procedure for optimal parameter choice should be done in a dynamic fashion and works as follows.

let us call $S_t\left(H^{\mathbf{w}}\right)$ the capital accumulated after t^{th} trading period using the expert $H^{\mathbf{w}}$ with initial capital $S_0 = 1$. We refer to each wealth $S_t\left(H^{\mathbf{w}}\right)$ as an expert where $\mathbf{w}$ represents the parameter or set of parameters that needs to be selected. In online trading we can try to learn the parameter or combination of parameters that will generate the maximum wealth growth. But similar to individual stocks, the various strategies, also called experts, induce a rather volatile set of terminal wealth and standard expert combination algorithms (Cesa-Bianchi et al., 1997) tend to fail. In this thesis, we will adaptively learn and invest in some weighted average of all the expert algorithms indexed by the parameter or combination of parameter choice. More specifically, we will uniformly distribute the available capital amongst all the experts at the beginning of the trading experiment and will never rebalance after the first period. This strategy reduces to a simple Buy-and-Hold of all the chosen strategies. Therefore, we form a mixture of all experts using a positive probability distribution q_w on the set of all possible parameters w of positive integers (see Gyorfi et al. (2004)). The investment strategy simply weights these experts H^w according to their past performances and the q^w such that after the t^{th} trading period the investor wealth becomes:

$$\mathbf{S}_t = \sum_w q_w S_t\left(H^w\right) \tag{3.11}$$

where $S_t\left(H^w\right)$ is the capital accumulated after t^{th} trading period using the expert H^w with initial capital $S_0 = 1$. We then form our final portfolio by weighting all expert portfolio using the following

$$\mathbf{b}\left(X_1^{t-1}\right) = \frac{\sum_w q_w S_{t-1}\left(H^w\right) \mathbf{b}^w\left(x_1^{t-1}\right)}{\sum_w q_w S_{t-1}\left(H^w\right)} \tag{3.12}$$

Table 3.2 presents a typical algorithm for combining experts when there is only one parameter to optimise and where ALG stands for algorithm under consideration. Extension to multiple parameter choice is very trivial

Table 3.2 Uniform Expert Combination $\mathbf{ALG}\left(\mathbf{X}, w\right)$

```
Input
              $\mathbf{X}_1^n$: matrix of price relative
              $W$: maximum allowed parameter value
Output
              $\mathbf{b}$: matrix of portfolio weights
Initialise    $\mathbf{b}_0 = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$: uniform distribution of weights vector
for           $t = 1, 2, ...n$ do
          1   for $w = 1, 2, ..., W$ do
          2       find the portfolio weight vector $\mathbf{b}_t = \mathbf{ALG}(\mathbf{X}_1^t, w)$
          3   end for
          4   combine the expert's portfoliios
              $\mathbf{b}_t = \frac{\sum_w q(w) s_{t-1}(H^w) \mathbf{b}_t(w)}{\sum_w q(w) s_{t-1}(H^w)}$
          5   update the portfolio wealth
              $\mathbf{S}_t = \mathbf{S}_{t-1} \times (b_t . x_t)$
          6   update the experts wealth
              $s_t\left(w\right) = s_{t-1}\left(w\right) \times \left(\mathbf{b}_t\left(w\right) . \mathbf{x}_t\right)$
end for
```

3.7 TRADING COSTS AND MARKET IMPACTS

Because our algorithms are likely to generate a large number of daily transactions and because transactions costs are very costly, we need to be concerned about the number of commissions this portfolio will incur in actual market trading. Although these costs have been significantly reduced in recent years due to technological advances and improved market liquidity (see

Table 3.3), transaction costs remain an issue to be carefully analyzed if algorithms are to be trusted for real-market applications. In this study we work on the assumption that there are charges on all transactions equal to a fixed percentage of the amount transacted. We adopt the proportional transaction cost model following Blum and Kalai (1999) and Borodin et al. (2004), that is, rebalancing the portfolio on any given day incurs transaction costs for both buy and sell orders.

Table 3.3	Broker's Commission Estimates
OptionsHouse	\$4.75 Flat-Rate Stock Trades
E*Trade	\$9.99
TD Ameritrade	Straightforward pricing with \$9.99 for Internet equity trades
Interactive Brokers	Stocks & ETFs - \$0.0005 - \$0.0035 per share. See site for details.
LOYALS3	\$0 Online Stock Trades and IPO Investments
Scottrade	\$7.00 Online Stock Trades
Trade Station	\$0.006 per share or \$6.99 flat

Source: http://www.nasdaq.com/investing/online-brokers/

At the beginning of the t^{th} trading day, the portfolio manager rebalances the portfolio from the previous closing price adjusted portfolio b_{t-1} to a new portfolio b_t. Specifically, we consider a transaction cost rate $c \in (0, 1)$, so the transaction cost will be charged according to

$$\frac{c}{2}\sum_{k=1}^{m}\left|\widetilde{b}_t^k - b_t^k\right| \tag{3.13}$$

Thus, with a transaction cost rate the total wealth achieved by the strategy becomes

$$S_T^c = S_0\prod_{t=1}^{T}\left[(\mathbf{b}_t, \mathbf{x}_t)\left(1 - \frac{c}{2}\sum_{k=1}^{m}\left|\widetilde{b}_t^k - b_t^k\right|\right)\right] \tag{3.14}$$

In this thesis, our base-case transaction costs are assumed to cost a round-trip trading of 10 basis points per trade. This is equivalent to setting $c = 0.1$ in Equation 3.14. We assume that 10 basis points per trade is enough to incorporate not only transaction costs but also compensate our estimate of price slippage and other costs like market frictions. To get a deeper understanding of the impact of transactions costs we provide simulation results that aim at determining the levels at which the net-of-cost strategy performance equal or beat both the market benchmark and the best stock.

All our algorithms also assume that all portfolio adjustments are implemented using the quoted prices and that all transactions are implemented simultaneously using these prices. This is of course an oversimplification of what really happens in actual trading where a time delay is needed between updating of portfolio weights and actual trading. Although recent technological advances have made computerised systems a natural candidate for fast order execution, there is still no guarantee that market orders will be implemented instantly unless the portfolio manager is happy to cross the bid-ask spread at every round of trading. These trading frictions will necessarily generate discrepancies between the model returns and those realised in actual trading.

An additional caveat is our assumption that all trades could be implemented using the closing price. While in principle there is nothing special about the closing price (i.e. our algorithms can trade at any time during the trading day) practical consideration related to datasets gathering and availability dictated the use of closing prices. Our algorithms assume that all portfolio adjustments are implemented using the quoted prices they receive as inputs. This means that all transactions are implemented simultaneously using the quoted prices. With current online brokers a computerised system can issue all transaction orders almost instantly but there is no guarantee that they will be all implemented instantly. This trading "friction" will necessarily generate discrepancies between the input prices and implementation prices.

A related problem that one must face when actually trading is the difference between bid and ask prices. These bid-ask spreads (and the availability of stocks for both buying and selling) are functions of stock liquidity and are typically small for large market capitalisation stocks. We consider here only very large market cap stocks.

3.8 CONCLUSION

Any report of abnormal returns using historical markets should be suspected of "data snooping". In particular, all of our historical datasets are conditioned on the fact that all stocks were traded every day and there were no bankrupcies or stocks that became virtually worthless in any of these datasets. Furthermore, when a datasets is excessively mined by testing many strategies there is a substantial chance that one of the strategies will be successful by simple over-fitting. To mitigate the risk of data snooping we calibrate our models using the NYSE(O) data and use the same parameters for all other markets.

4.0 MACHINE-LEARNING ALGORITHMS FOR PORTFOLIO SELECTION: AN EMPIRICAL SURVEY

4.1 INTRODUCTION

In the last ten years or so, many machine-learning-inspired sequential PS algorithms have been proposed in the litterature. These proposed algorithms have demonstrated that sequential portfolio rebalancing strategies can generate substantial wealth way above what could be achieved by mere luck. Even more impressive is the fact that some of these algorithms have shown that the wealth achieved by some systmatic rebalancing strategies could be guaranteed a certain minimum given a sufficiently large amount of trading time. Although the rigourous mathematical foundations that underpin these machine-learning algorithms are now widely accepted, their finite sample properties are an ongoing challenge.

Several approaches to Online PS have been proposed in the machine-learning literature. Cover (1991), for example, proposed the Universal Portfolios (UP) strategy that weights all constant rebalanced portfolios, referred to as "experts", by their empirical probability distribution generated from the performance of each expert. The regret achieved by Cover's UP is $O\left(m \log n\right)$, where m denotes the number of stocks and n denotes the number of trading days. However, the implementation cost of the UP algorithm has been shown to be exponential in the number of stocks and thus restricts the number of assets used in real-market experiments. Although Kalai and Vempala (2002) presented a time-efficient implementation of Cover's UP based on non-uniform random walks, the performance of the UP algorithm has not been satisfactory enough in historical simulations.

The Exponential Gradient strategy (EG) of Helmbold et al. (1996, 1997) for Online PS proposes a PS algorithm using multiplicative updates. To achieve this, the EG strategy tries to maximise the expected logarithmic portfolio daily return (approximated using the last price relative) and minimise the deviation between next portfolio and last portfolio. One straightforward interpretation of the EG algorithm is that it tends to track the stock with the best performance in the last period but keep the new portfolio close to the previous portfolio weights.

Borodin et al. (2004) propose a non-universal but empirically robust portfolio strategy named Anti-Correlation (Anticor or simply AC). The Anticor strategy simply takes advantage of the statistical properties of mean-reverting stock prices where the underlying motivation is to bet on the consistency of positive lagged cross-correlation and negative autocorrelation. Although they do not provide any theoretical guarantee, empirical results showed that Anticor can outperform most existing state-of-the-art strategies in real-market historical data.

Györfi et al. (2006) introduced a framework of Nonparametric Kernel-based learning strategies for PS based on nonparametric prediction techniques of Gyorfi and Schäfer (2003). Their algorithm first identifies a list of similar historical price relative sequences whose Euclidean distance with the recent market window is smaller than a threshold and then optimises the portfolio with respect to the list of observed realised returns following instances of similarity. Following the same line, the Nonparametric Nearest Neighbor learning (NN) strategy proposed in Gyorfi et al. (2008) aims to search for the Nearest Neighbors in historical price relative sequences rather than search price relatives within a specified Euclidean ball.

Recently, Li et al. (2012, 2013) proposed the Confidence Weighted Mean Reversion (CWMR) and the Passive Aggressive Mean Reversion (PAMR) strategies. These algorithms actively exploit the mean reversion property and the second-order information of a portfolio. These author's algorithms have been empirically shown to be robust trading strategies as they outperform many earlier state-of-the-art algorithms in historical simulations.

In general, the machine-learning literature regarding PS strategies distinguishes three main categories of strategies. One of the most promising category of algorithms is the so called "reversal" strategies. These strategies tend to buy stocks as they fall, and sell stocks when they rise, giving rise to concave payoff curves. Concave strategies do very poorly in strong, down or up market, but tend to do well in oscillating markets. Another promising category is referred to as "momentum" strategies, which buy stocks when they rise or sell stocks when they fall. Naturally, these strategies do well in trending, down or up markets. Finally, the third category is capable of capturing both momentum and reversal patterns in the market.

This chapter provides an empirical survey of some of the more promising online PS techniques. The algorithms surveyed have all demonstrated excellent finite sample performance using older datasets (see Table 1.2). Our main aim is to provide a timely survey, using more recent datasets, to assess the acclaimed robustness of some of the earlier published works in both machine-learning and data mining fields. In doing so we will highlight the main limitations presented by these so-called state-of-the-art PS algorithms and create the platform against which we will introduce our own, newly developed Online PS algorithms.

4.2 KERNEL-BASED PORTFOLIO SELECTION ALGORITHM (GYORFI ET AL. (2006))

4.2.1 Introduction

Gyorfi et al. (2006) proposed a nonparametric Kernel-based sequential investment strategy for PS problems. Based on the nonparametric prediction of individual sequences of Gyorfi and Schafer (2003), the Kernel-based algorithm proves that there exist completely nonparametric investment strategies that can effectively find complex dependences in the past data and use this information to produce a rapid growth of the investor's capital. The mechanics of the Kernel-based nonparametric sequential investment strategy consist of three basic steps. The algorithm first identifies a list of similar historical price relative sequences whose Euclidean distances to the recent market windows are smaller than a threshold. In the second step, the algorithm constructs a vector of portfolio weights that maximises the returns following observation of similar market sequences. Finally, the algorithm combines the various portfolio "experts" according to the idea of UBAH. Section 4.2.2, 4.2.3 and 4.2.4 present a detailed analysis of these steps.

4.2.2 Pattern Matching for Kernel-Based Algorithm

To better understand the Kernel pattern matching algorithm, let us assume that we are given a parameter w, which represents a fixed window length and an integer called l (l will be defined below). Let us further assume that $\mathbf{x}_1, \mathbf{x}_2, ...$are realisations of the random vectors $\mathbf{X}_1, \mathbf{X}_2, ...$drawn from the vector of stationary and ergodic process $\{\mathbf{X}_t\}_{-\infty}^{+\infty}$. We call $\mathbf{x}_t^{t-w+1} \in \mathbb{R}m$ the most recent window of price relatives and $\mathbf{x}_{i-1}^{i-w+1} \in \mathbb{R}^{w\times m}, i = 1, 2, ...n$ any preceding vector of price relative of the same sise as the most recent one To determine the weight portfolio vector on the t^{th} trading period, the portfolio manager iterates all historical sequences of price relatives $\mathbf{x}_{i-1}^{i-w+1}$ of sise $w \times m$ and determines instances where $\mathbf{x}_{i-1}^{i-w+1}$ is similar in the Euclidian sense to the most recent window fo price relatives $\mathbf{x}_t^{t-w+1}$. Mathematically, the nonparametric Kernel-based sample selection of Gyorfi et al. (2006) identifies the similarity set by comparing two market windows via Euclidean distance; that is:

$$\mathbf{J}_t^{\{w,l\}} = \left\{w + 1 \leq i \leq t - 1 : \left\|\mathbf{x}_t^{t-w+1} - \mathbf{x}_i^{i-w+1}\right\| \leq r\right\} \tag{4.1}$$

where in the general case $r = G\left(c, l\right), c \in \mathbb{R}$ and l is an integer. In practice, Gyorfi et al.(2006) select r such that $r = \frac{c}{l}$,where $l = 1, 2, ...L$

4.2.3 Portfolio Optimisation for Kernel-Based Algorithms

Given fixed positive integers w, l, and all locations of historical matches, the Kernel-based algorithm constructs a fixed portfolio vector to optimise the returns for the trading periods following each matching period in the following way:

$$\mathbf{b}^{(w,l)}\left(\mathbf{X}_{t-1}^1\right) = \underset{b\in\Delta_d}{\arg\max} \prod_{i\in\mathbf{J}_t^{\{w,l\}}} \langle\mathbf{b}, \mathbf{x}_i\rangle \tag{4.2}$$

If the product in equation (4.2) is void the Kernel algorithm sets $\mathbf{b}$ simply as $\mathbf{b} = \left(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}\right)$. This nonparametric Kernel-based sequential investment strategy has been shown to guarantee an optimal asymptotic growth rate of capital for all stationary and ergodic markets by effectively exploiting hidden complicated dependences of asset prices on the past evolution of the market sequences.

In practice on can assume that the log-optimal portfolio designed to the matches is a convex programming problem, which can be simplified for semi log-optimal portfolio studied in Gyorfi (2007), where one has a quadratic programming problem.

4.2.4 Parameter Choice and Combining Portfolio of Experts

The Kernel-based sequential PS algorithm requires two important parameters (w, l) that need to be set by the portfolio manager before any trading takes place. In most instances the performance of the algorithm depends on the optimal choice of these parameters and fluctuates significantly, depending on which pairs of parameter one chooses. Because it is impossible to know before hand what parameter set will generate optimal growth in wealth, one reasonable approach is to uniformally allocate the initial capital amongst all the various experts generated by each combination of (w, l) on the first day and never rebalance afterward. This uniform Buy-and-Hold strategy of all the experts allows Gyorfi et al. (2006) to form a mixture of all experts using a positive probability distribution $q_{w,l}$ on the set of all parameter sets $\{(w, l) : \ w = 1, 2, ..., W = 10 \text{ and } l = 1, 2, ..., L = 5\}$ of positive integers. The investment strategy simply weights these experts $H^{w,l}$ according to their past performances and the $q^{w,l}$ such that after the t^{th} trading period the investor wealth becomes

$$S_t = \sum_{w,l} q_{w,l} S_t \left(H^{w,l}\right) \tag{4.3}$$

where $S_t \left(H^{w,l}\right)$ is the capital accumulated after the t^{th} trading period using the expert $H^{w,l}$ with initial capital $S_0 = 1$. Gyorfi et al. (2006) then form the final portfolio by weighting all expert portfolio using the following

$$\mathbf{b}\left(X_1^{t-1}\right) = \frac{\sum_{w,l} q_{w,l} S_{t-1}\left(H^{w,l}\right) h^{w,l}\left(x_1^{t-1}\right)}{\sum_{w,l} q_{w,l} S_{t-1}\left(H^{w,l}\right)} \tag{4.4}$$

There are, of course, many alternative ways to combine these expert portfolios. Algorithms such as the Exponential Gradient (Helmbold et al. (1998)) or the Switching Portfolios (Singer et al. (2003)) could be ideal candidates as they efficiently allocate assets away from poor performing experts into winning experts. Although it has been shown that the wealth generated by these algorithms is very competitive relative to the best constantly rebalanced portfolio (BCRP), these alternative methods have the potential to add much more transaction cots than necessary. Combining portfolios of experts in this thesis will therefore be done according to the methodology proposed by Gyorfi et al. (2006), as it has the major advantage that no further parameter tuning is required.

Table 4.2.1 shows a pseudo code implementation of the Kernel-based PS algorithm.

Table 4.2.1: KERNEL $(\mathbf{X}, w, l)$

```
Inputs   $l$ : partitioning parameter
         $w$ : maximum window sise
         $b_0$ : initial portfolio weights
         $c$: fixed constant parameter
Output   $\mathbf{b}_t$ : expert's portfolio weights
for      $t = 1, 2, ..., n$  do
      1  receive price relative $\mathbf{x}_t$
      2  if $t < w + 1$ then
      3      $\mathbf{b}_t = (\frac{1}{m}, \frac{1}{m}, \cdots \frac{1}{m})$
      4  end if
      5  for $i = w + 1$ to $t - 1$ do
      6      if $\left\| x_{i-w}^{i-1} - x_{t-w}^{t-1} \right\| \leq \frac{c}{l}$ then
      7          $\mathbf{J}_t^{\{w,l\}} = \mathbf{J}_t^{\{w,l\}} \cup \{i\}$
      8      end if
      9  end for
     10  if $\mathbf{N}_t == \varnothing$ then
     11      $\mathbf{b}_t = (\frac{1}{m}, \frac{1}{m}, \cdots \frac{1}{m})$
     12  else
     13  $\mathbf{b}_t^{(w,l)} = \arg\max_{b \in \Delta_d} \prod_{i \in \mathbf{J}_t^{\{w,l\}}} \langle \mathbf{b}, \mathbf{x}_i \rangle$
     14  Combine experts using Equation 4.4
end
```

4.2.5 Empirical Results

This section presents numerical results obtained by applying the Kernel-based algorithm to six financial market datasets described in Chapter 3. The back-testing experiments in this thesis will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed daily. For the purpose of our simulatoin we use the same settings as in Gyorfi et al. (2006). We set the $L = 10, W = 5$ and $c = 0.1$.

4.2.5.1 Analysis of the Kernel Cumulative Wealth The first experiment evaluates the total wealth achieved by the Kernel-based learning-to-trade algorithm including a 10 basis points transaction cost.

Figure 4.2.1: Cumulative Wealth of the Kernel-based PS algorithm

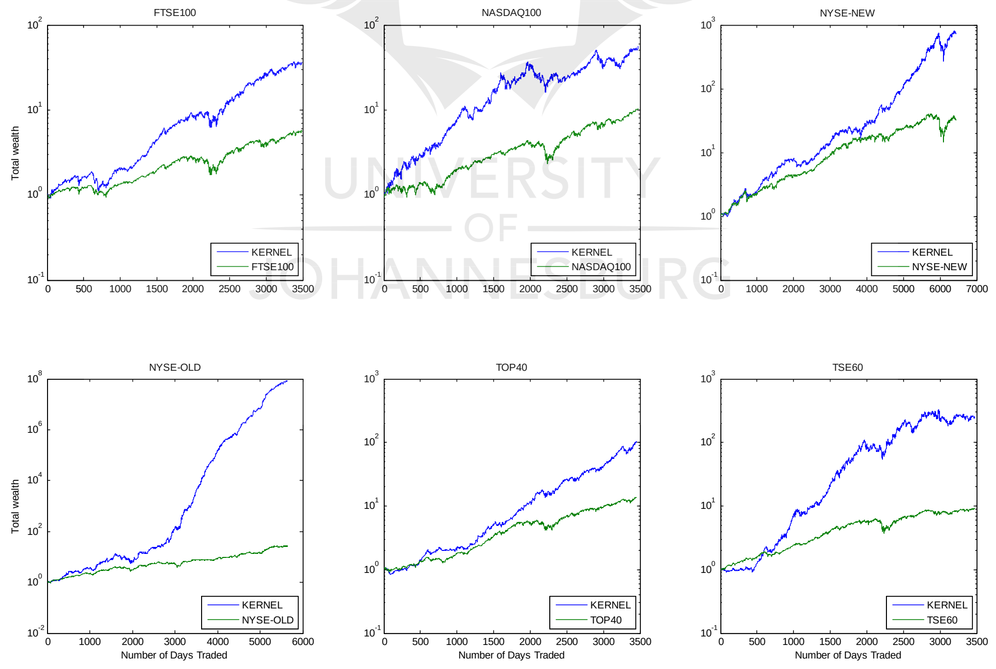


Figure 4.2.1 summarises the cumulative wealth achieved by the Kernel-based algorithms on the six market datasets. The market index is calculated as an equally weighted Buy-and-Hold portfolio on all stock available for that particular market.

There are important observations that one can draw from Figure 4.2.1. Not only does the Kernel-based PS algorithm demonstrate very good performance on old data (NYSE(O), NYSE(N)), the performance seems to be carrying over to recent and previously untested datasets. For example, on the South African TOP40 index datasets after trading for 15 years, the total wealth achieved by the Kernel strategy impressively increases from \$1 to almost \$100, which is much higher than the \$13.6 achieved by the market index in the same trading period. Although similar impressive results are achieved for all data sets, it is also evident that the growth in portfolio wealth is not as spectacular as reported by the authors using older data sets. One simple explanation could be that stock markets globally are far more efficient than they were just a couple of year ago implying that the Kernel strategy is less effective at exploiting equity market mispricings in recent data.

An alternative way to assess the performance of the Kernel-based PS strategy is to look at the Compound Annual Growth Rate (CAGR) over the full sample. The model generates a very impressive CAGR of around 40% and 49% using the TOP40 and the TSE60 datasets. These impressive results compare very favourably with a CAGR of 21% and 18% for TOP40 and TSE60 market indices respectively. Similar wealth growth can be observed from all other datasets including the NASDAQ, the NYSE(O) and the NYSE(N).

In addition to the final cumulative wealth growth over the full sample, we are also interested in examining how the cumulative wealth changes over different trading sub-periods. Table 4.2.2 shows the sub-periods CAGR generated by the proposed Kernel-based PS algorithm. In Table 4.2.2, "full" simply means the cumulative annual growth rate of returns over the full sample, while "1Yr, 2Yrs,..., 10Yrs" represent the cumulative returns generated by the Kernel-based (simply called ALG) PS algorithm over the last year, last 2 years, etc... in our trading sample

Table 4.2.2: Kernel CAGR over different trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.29	0.13	0.40	0.21	0.49	0.18	0.34	0.19	0.30	0.15	1.26	0.16
1Yr	0.19	0.22	0.44	0.19	-0.06	0.12	0.84	0.39	0.67	0.33	0.57	-0.01
2Yrs	0.14	0.16	0.65	0.22	-0.04	0.09	0.18	0.21	0.08	-0.03	0.87	0.15
3Yrs	0.24	0.13	0.48	0.18	0.05	0.07	0.21	0.20	0.20	-0.07	1.59	0.23
4Yrs	0.33	0.17	0.43	0.20	0.08	0.09	0.23	0.22	0.36	0.02	1.56	0.19
5Yrs	0.39	0.24	0.46	0.23	0.32	0.16	0.27	0.30	0.41	0.05	1.66	0.20
7Yrs	0.28	0.13	0.50	0.18	0.27	0.10	0.16	0.17	0.43	0.08	1.83	0.20
10Yrs	0.34	0.16	0.48	0.24	0.47	0.15	0.24	0.19	0.41	0.07	2.67	0.19
15Yrs	0.30	0.14	0.45	0.23	0.58	0.17	0.29	0.20	0.34	0.07	2.49	0.13

From the results, we can see that the proposed Kernel strategy consistently surpasses its respective benchmarks on all datasets, except the TSE60 in Canada. On the FTSE100, the Kernel-based strategy outperformed the market index on 12 of the 14 test samples. The performance of the Kernel-based PS algorithm has been impressive in recent times, with a 44% gain in 2013 and a 65% gain in 2012, using the TOP40 data set in South Africa. In the case of the NASDAQ, for example, the Kernel-based algorithm achieved a CAGR of 84% in 2013, which is significantly higher than the 39% achieved by the market index. This is very impressive indeed, given that this is the index of the biggest 100 companies by market capitalisation listed on the NASDAQ.

4.2.5.2 Kernel Strategy Risk Analysis We now evaluate the Kernel-based algorithm against various risk measures that could shed more light on the performance of the algorithm. These measures of risk include returns volatility, σ, maximum drawdown, MDD, and the risk-adjusted return referred to as the annualised Sharpe Ratio, SR.

In Table 4.2.3, σ represents the annualised standard deviation of strategy returns. α stands for the Jensen Alpha; β measures the sensitivity of the strategy returns relative to the market benchmark excess returns; λ is a measure of the strategy market-timing ability;

$\beta \uparrow$ is the sensitivity of the strategy in a rising market; and $\beta \downarrow$ represents the same measure in a falling market. SR indicates the strategy Sharpe Ratio while MDD respresents the strategy maximum drawdown. PP is the percentage of time the strategy generates positive returns, PP$\uparrow$ stands for the percentage of times the strategy generates positive returns in a rising market and PP$\downarrow$ is the percentage of times the strategy generates positive returns in a falling market. ALG in Table 4.2.3 simply represents the kernel-based PS strategy. These measures will be used throughout this thesis in similar tables.

Table 4.2.3: Risk Statistics of the Kernel-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.23	0.18	0.22	0.16	0.37	0.17	0.36	0.23	0.23	0.19	0.36	0.13
α	0.06	0.00	0.08	0.00	0.09	0.00	0.10	0.00	0.05	0.00	0.29	0.00
β	1.01	1.00	0.79	1.00	1.29	1.00	0.83	1.00	0.94	1.00	1.24	1.00
γ	-0.51	0.00	0.22	0.00	0.50	0.00	-1.26	0.00	0.25	0.00	3.07	0.00
$\beta \uparrow$	0.97	1.00	0.78	1.00	1.11	1.00	0.72	1.00	0.96	1.00	1.14	1.00
$\beta \downarrow$	0.99	1.00	0.79	1.00	1.09	1.00	0.79	1.00	0.93	1.00	1.13	1.00
SR	0.95	0.40	1.34	0.82	1.09	0.62	0.79	0.56	0.92	0.44	2.24	0.64
MDD	-0.41	-0.40	-0.25	-0.30	-0.51	-0.41	-0.56	-0.47	-0.63	-0.64	-0.61	-0.37
PP	0.54	0.55	0.55	0.56	0.54	0.56	0.53	0.54	0.54	0.55	0.52	0.53
PP$\uparrow$	0.81	1.00	0.77	1.00	0.75	1.00	0.74	1.00	0.78	1.00	0.72	1.00
PP$\downarrow$	0.23	0.00	0.28	0.00	0.27	0.00	0.29	0.00	0.25	0.00	0.30	0.00

Table 4.2.3 shows some risk evaluation results on the six datasets. As shown in this table the Kernel-based strategy has a volatility of returns (as measure by the annualised standard deviation of daily returns) that is higher than the one achieved by its respective benchmarks. However, when returns are adjusted by volatility, we see that the Sharpe Ratios generated by the Kernel-based algorithms are significantly higher than the respective stock market indices. This is an indication that the Kernel-based strategy generates much better risk adjusted returns.

Furthermore, we observe that the maximum drawdown on the six stock datasets.compares very well with that of their respective benchmarks except for the NYSE(O), which is twice as high as that of the benchmark. Regarding the market-timing ability of the Kernel-based sequential investment strategy we observe that the bull and bear market betas are not dissimilar on all datasets. This seems to suggests that the Kernel strategy portfolio allocation weights does anticipate very well subsequent market directions. However, please note that the γ coefficient for the Trenor-Mazuy market timing regression is negative for the FTSE100 and the NASDAQ data sets.

4.2.6 Conclusion

The cumulative wealth growth results show that the Kernel-based PS algorithm achieves very impressive cumulative returns on all datasets. Although it is commonly argued in finance that no real financial instrument can guarantee a high return without risk, we believe that the risk taken while investing with the Kernel-based PS model is fully justified by the returns that the model generates. These encouraging results show that the Kernel-base portfolio selecton algorithm is able to achieve a very good trade-off between return and risk. We therefore concur with Gyorfi et al. (2006) that the Kernel-based sequential PS algorithm is indeed a very robust investment strategy that can generate reasonable growth in portfolio wealth without any knowledge about the probability distribution of stock prices. However, we caution that the performance of the Kernel-based PS strategy has not been of the same magnitude as the one reported by Gyorfi et al. (2006) suggesting that stock markets are more efficient in recent times than what the original paper seems to have suggested.

4.3 NEAREST-NEIGHBOUR ALGORITHM (GYORFI ET AL. (2008))

4.3.1 Introduction

Gyorfi et al. (2008) proposed another universal, nonparametric Nearest Neighbor (NN)-based sequential investment strategy that has similar constructs as the Kernel-based algorithm discussed in the previous section. As in the case of the Kernel-based sequential PS algorithm, the mechanics of the NN-based nonparametric investment algorithm consist of three basic steps. First, the algorithm searches for the historical price relatives whose preceding market windows are within the l Nearest Neighbors of the latest market window in terms of Euclidean distance. In the second step the algorithm constructs a vector of portfolio weights that would have maximised the returns following observation of similar market sequences. Finally, the algorithm combines the experts according to the idea of the UBAH presented in the previous section.

4.3.2 Nearest-Neighbor Pattern Matching

As usual, we consider a fixed window length, w, and a parameter l that determines the number of NN. We assume that $\mathbf{x}_1, \mathbf{x}_2, ...$are realisations of the random vectors . $X_1, X_2, ...$drawn from the vector of stationary and ergodic process $\{X_n\}_{-\infty}^{+\infty}$. let us call $\mathbf{x}_n^{n-w+1} \in \mathbb{R}^{w \times m}$ the most recent window of price relatives and $\mathbf{x}_{i-1}^{i-w+1} \in \mathbb{R}^{w \times m}, i = 1, 2, ...n$ any preceeding vector of price relative to the same sise as the most recent one To determine the weight portfolio vector on the n^{th} trading period, the NN algorithm searches the price relatives whose preceding market windows $\mathbf{x}_{i-1}^{i-w+1}$ of sise $w \times m$ are within the l NN of the latest market window $\mathbf{x}_n^{n-w+1}$ of sise $w \times m$ in terms of Euclidean distance. Mathematically, the nonparametric NN-based sample selection algorithm identifies the similarity set by comparing two market windows via Euclidean distance, that is:

$$\widehat{\mathbf{J}}_{n,s}^{(w,l)} = \left\{ i; w + 1 \leq i \leq n \text{ such that } \mathbf{x}_{i-1}^{i-w} \text{ is amongst the } \widehat{l} \text{ NN of } \mathbf{x}_n^{n-w+1} \text{ in } \mathbf{x}_1^w, ..., \mathbf{x}_{n-1}^{n-w} \right\} \tag{4.5}$$

where $\widehat{l} = \lfloor p_l n \rfloor$ and p_l is chosen such that $\lim\limits_{l \to \infty} p_l = 0$. In practice Gyorfi (2008) selects $\widehat{l}$ such that $p_l = 0.02 + 0.5\frac{l-1}{L-1}, l = 1, 2, ...L$

4.3.3 Nearest-Neighbor Portfolio Optimisation

For fixed positive integers w, l, and after locating historical matches the Kernel-based algorithm constructs a fixed portfolio vector to optimise the returns for the trading periods following each matching period in the following way..

$$\mathbf{b}^{(k,l)}\left(\mathbf{X}_1^{n-1}\right) = \underset{b\in\Delta_m}{\arg\max} \prod_{i\in\widehat{\mathbf{J}}_{n,s}^{(w,l)}} \langle \mathbf{b}, \mathbf{x}_i \rangle \tag{4.6}$$

Of course if the product in equation (4.6) is void, the NN algorithm sets $\mathbf{b}$ simply equal to $\mathbf{b} = \left(\frac{1}{d}, \frac{1}{d}, ..., \frac{1}{d}\right)$. This nonparametric sequential investment strategy has been shown to guarantee an optimal asymptotic growth rate of capital for all stationary and ergodic markets by effectively exploiting hidden complicated dependences of asset prices on the past evolution of the market sequences. Table 4.3.1 shows a pseudo code implementation of the NN-based PS algorithm.

Table 4.3.1: **NN** $(\mathbf{X}, l, W)$

```
Inputs    $l$ : partitioning parameter
          $W$ : maximum window sise
Output    $\mathbf{b}_t$ : expert's portfolio weights for the $t^{th}$ trading day
for       $t = 1, 2, ..., n$ do
      1   receive price relative $\mathbf{x}_t$
      2   if $t < w + 1$ then
      3       $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
      4   end if
      5   for $i = w + 1$ to $t - 1$ do
      6       if $x_{i-w}^{i-1}$ is amongst the $\widehat{l}$ NN of $\mathbf{x}_t^{t-w+1}$ then
      7           $\mathbf{J}_t^{\{w,l\}} = \mathbf{J}_t^{\{w,l\}} \cup \{i\}$
      8       end if
      9   end for
     10   if $\mathbf{N}_t == \varnothing$ then
     11       $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
     12   else
     13   $\mathbf{b}_t^{(w,l)} = \underset{b \in \Delta_d}{\arg\max} \prod_{i \in \mathbf{J}_t^{\{w,l\}}} \langle \mathbf{b}, \mathbf{x}_i \rangle$
end
```

4.3.4 Combining Portfolios of Experts

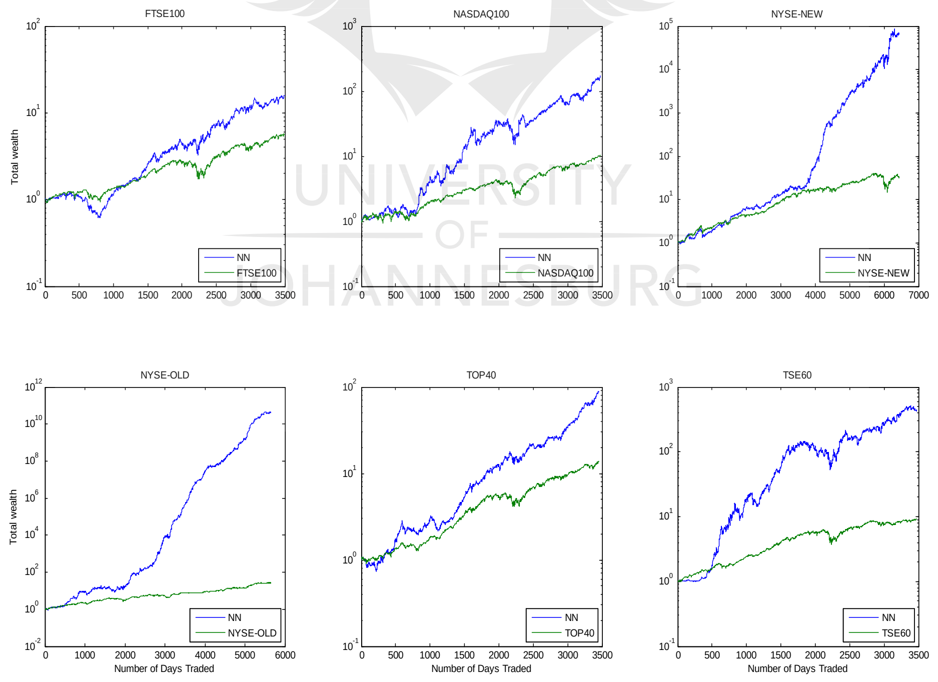
As in the case of the Kernel algorithm, the NN-based sequential PS algorithm requires two important parameters (w, l) that need to be set in advance by the portfolio manager. The experts are therefore combined in the same way as in section 4.2.4.

4.3.5 Empirical Results

This section presents numerical results obtained by applying the NN-based algorithm to six financial market datasets described in Chapter 2. For our simulatoin we use the same setting as in the original thesis where $L = 10$ and $W = 5$;

4.3.5.1 Analysis of Cumulative Wealth The first experiment evaluates the compounded wealth achieved by the NN-based learning-to trade algorithm including over the full sample. As usual this test includes a 10 basis points transaction cost. Figure 4.3.1 shows the total wealth achieved by the NN algorithm on the six market datasets. The market index is calculated as an equally weighted Buy-and-Hold portfolio on all available stocks for the market under consideration.

Figure 4.3.1: Performance of the Nearest-Neighbor algorithm for PS



From Figure 4.3.1 we observe that not only does the NN-based PS algorithm demonstrate very good performance on old data (NYSE(O), NYSE(N)), the performance has been impressive on recent and previously untested datasets. For example, on the South African TOP40 index data set, the total wealth achieved by the NN strategy impressively increases from \$1 to almost \$100. This wealth growth is much higher than the \$13.6 achieved by the market index over the same 15-years periods. Excluding the FTSE100, all other datasets display impressive growth in portfolio wealth. Despite this impressive performance, it is noticeable that the performance in recent datasets is significantly reduced compared to the results reported in the original thesis (see Gyorfi et al. (2008)). While the algorithm achieved a cumulative wealth of $\$4.32 \times 10^{10}$ in the NYSE(O), the model can only achieve \$159 using the FTSE100, \$1620 using the NASDAQ and \$87.1 using the TOP40 index in South Africa.

The NN algorithm also displays a very respectable CAGR of around 39% and 55% using the TOP40 and TE60 datasets respectively (see Table 4.3.2). This compares very farourably with a CAGR of 21% and 18% for both market indices. Similar CAGR can be observed from all other datasets, including the NASDAQ, the NYSE (O) and the NYSE (N).

Table 4.3.2: Cumulative returns over different trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.22	0.13	0.39	0.21	0.55	0.18	0.45	0.19	0.54	0.15	1.98	0.16
1Yr	0.31	0.22	0.60	0.19	0.21	0.12	1.32	0.39	0.41	0.33	0.46	-0.01
2Yrs	0.14	0.16	0.70	0.22	0.31	0.09	0.54	0.21	0.91	-0.03	1.09	0.15
3Yrs	0.23	0.13	0.53	0.18	0.27	0.07	0.41	0.20	0.88	-0.07	2.54	0.23
4Yrs	0.24	0.17	0.43	0.20	0.23	0.09	0.46	0.22	0.87	0.02	2.53	0.19
5Yrs	0.34	0.24	0.42	0.23	0.47	0.16	0.57	0.30	0.83	0.05	2.57	0.20
7Yrs	0.24	0.13	0.42	0.18	0.20	0.10	0.37	0.17	0.78	0.08	2.33	0.20
10Yrs	0.30	0.16	0.43	0.24	0.42	0.15	0.46	0.19	1.06	0.07	3.48	0.19
15Yrs	0.24	0.14	0.43	0.23	0.61	0.17	0.49	0.20	0.97	0.07	3.95	0.13

In addition to the full sample cumulative wealth and the CAGR, we also examine how the cumulative wealth changes from one trading sub-period to the next. Table 4.3.2 shows the trend of the cumulative wealth achieved by the proposed NN algorithm over different trading sub-periods. From the results, we see that the proposed NN-based PS algorithm has been quite impressive in recent years with a 60% gain in 2013 and 70% gain in 2012 using the TOP40 index in South Africa. In the case of the NASDAQ in the US, 2013 saw a returns of 132% for the NN-based algorithm as compared to about 39% for the market index. As argued earlier, this is indeed a very impressive performance given that this is the index that comprises the biggest 100 companies by market capitalisation listed on the NASDAQ. The same pattern can be seen on the Toronto Stock Exchange (TSE60) in Canada, where the algorithm achieves an annualised compounded growth rate that is about 3 times that achieved by its benchmark algorithm.

4.3.5.2 Performance Risk Analysis The NN risk measures are summarised in Table 4.3.3. As shown in this table, the NN-based strategy has a volatility of returns (as measured by the annualised standard deviation of daily returns) that is much higher than the one achieved by its respective benchmarks. However, we also notice that the Sharpe Ratio generated by the NN-based algorithms is also significantly higher than the respective stock market indices. This is an indication that the NN-based strategy generates much better risk-adjusted returns.

Table 4.3.3: Risk Statistics of the NN-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.26	0.18	0.23	0.16	0.46	0.17	0.42	0.23	0.35	0.19	0.48	0.13
α	0.04	0.00	0.06	0.00	0.11	0.00	0.14	0.00	0.13	0.00	0.38	0.00
β	1.08	1.00	0.93	1.00	1.38	1.00	0.89	1.00	1.08	1.00	1.25	1.00
γ	-0.72	0.00	0.06	0.00	1.03	0.00	-1.38	0.00	-0.04	0.00	2.83	0.00
$\beta\uparrow$	1.00	1.00	0.85	1.00	1.45	1.00	0.73	1.00	1.11	1.00	1.33	1.00
$\beta\downarrow$	1.03	1.00	0.85	1.00	1.31	1.00	0.93	1.00	1.06	1.00	1.22	1.00
SR	0.64	0.40	1.22	0.82	1.04	0.62	0.93	0.56	1.21	0.44	2.35	0.64
MDD	-0.49	-0.40	-0.32	-0.30	-0.64	-0.41	-0.60	-0.47	-0.53	-0.64	-0.54	-0.37
PP	0.53	0.55	0.55	0.56	0.53	0.56	0.53	0.54	0.54	0.55	0.54	0.53
PP$\uparrow$	0.79	1.00	0.77	1.00	0.70	1.00	0.74	1.00	0.75	1.00	0.69	1.00
PP$\downarrow$	0.21	0.00	0.27	0.00	0.30	0.00	0.29	0.00	0.28	0.00	0.36	0.00

Further, we observe that the maximum drawdown on the six stock market datasets are higher than those achieved by the respective benchmarks. When the Top40 was up, the NN-based algorithm had a positive returns 77% of the time while this number was about 80% of the time in the FTSE100 in the UK. Also impressive is the fact that the NN algorithm generates positive returns more that 20% of the time all datasets when the market indices were negative. This suggests that the value add of the NN-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market. This could indicate that the NN-based PS algorithm possesses some market-timing abilities. However, these market-timing skills displayed by the model are not felt accross all the datasets. In fact, the results shows that the γ measure is negative for the FTSE100, the NASDAQ and the NYSE(N).

4.3.6 Conclusion

In summary, the simulation results demonstrate that the NN-based PS algorithm achieves very impressive cumulative wealth growth on most datasets. These encouraging results show that the NN-based PS algorithm is capable of achieving a very good trade-off between return and risk. We therefore concur with Gyorfi et al. (2006), who claim that the NN-based sequential PS algorithm is indeed a very robust investment strategy that can generate substantial growth in portfolio wealth without any knowledge about the probability distribution of stock prices.

4.4 CORRELATION-DRIVEN MEAN REVERSION PORTFOLIO SELECTION ALGORITHM (LI ET AL (2011))

4.4.1 Introduction

In sections 4.2 and 4.3 we presented both the nonparametric Kernel-based algorithm (Gyorfi et al. (2006)) and the NN-based (Gyorfi et al. (2008)) algorithm for sequential PS problems. We argued that these pattern-matching PS algorithms follow some basic steps. First, the algorithms identifiy an index of historical price relative sequences whose similarity to the recent market windows are smaller than a pre-determined threshold. Then the algorithms optimise the portfolio with respect to the list of similar sequences using the idea of the BCRP. Although we argured that these methods are indeed empirically robust trading strategies, their over reliance on the Euclidean distance for similarity measures between the latest market window and the historical market windows present some important limitations. The Euclidean distance measure exploits only the magnitude and not the direction of information of the market windows' movements. As argued by Li et al. (2011), the use of the Euclidean distance could result in signal missclassification. In other words, the Euclidian distance may detect some similar appearances, that in fact may be in the opposite direction to what the portfolio manager may be trying to achieve. To ensure that similarities are well aligned in magnitude and direction, Li et al. (2011) introduce a novel learning-to-trade strategy for

PS problems, termed "Correlation-driven Nonparametric learning" (CORN) algorithm. As in the case of the Nearest Neighbor and the Kernel-based PS algorithms, CORN first seeks to locate the market windows that are similar to the latest market window via a correlation coefficient metric, and then constructs a log-optimum portfolio according to the idea of the best constant rebalanced portfolio (Cover 1991) strategy.

4.4.2 CORN Pattern Matching

As usual we assume that $\mathbf{x}_1, \mathbf{x}_2, ...$are realisations of the random vectors $\mathbf{X}_1, \mathbf{X}_2, ...$drawn from the vector of stationary and ergodic process $\{\mathbf{X}_n\}_{-\infty}^{+\infty}$. let us call $\mathbf{x}_n^{n-w+1} \in \mathbb{R}^{w\times m}$ the most recent window of price relatives and $\mathbf{x}_{i-1}^{i-w+1} \in \mathbb{R}^{w\times m}, i = 1, 2, ...n$ any preceeding vector of price relative of the same sise as the most recent one where w is a fixed window length To determine the weight portfolio vector on the n^{th} trading period, the algorithm searches for the price relatives whose preceding market windows $\mathbf{x}_{i-1}^{i-w+1}$ of sise $w \times m$ are similar to the latest market window $\mathbf{x}_n^{n-w+1}$ of sise $w \times m$ via a correlation coefficient metric ρ. Mathematically, the nonparametric CORN-based sample selection Li et al. (2011) identifies the similarity set by comparing two market windows via the correlation coefficient. The vector $\widehat{\mathbf{J}}_{n,i}^{w,\rho}$ is defined as

$$\widehat{\mathbf{J}}_{n,i}^{w,\rho} = \left\{w < i < n; correl\left(\mathbf{x}_{i-w}^{i-1}, \mathbf{x}_{t-w}^{n-1}\right) \geq \rho \mid -1 \leq \rho \leq 1\right\} \tag{4.7}$$

is called matching time. If $\widehat{\mathbf{J}}_{n,i}^{w,\rho} == \varnothing$,we simply set $\mathbf{b} = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$. Although Li et al. (2011) did not explicitly address how higher dimentional correlations should be measured, we propose the use of a multi dimentional correlation measure defined as follows. Given two vectors X and Y of dimension $w \times m$, our correlation coefficient is given by the following equation

$$correl\left(X, Y\right) = \frac{\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)\left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)}{\sqrt{\left(\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)^2\right)\left(\sum_i \sum_j \left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)^2\right)}}$$

where $\overline{\mathbf{X}}$ and $\overline{\mathbf{Y}}$ represent the average of $\mathbf{X}$ and $\mathbf{Y}$ of dimension $w \times m$.

4.4.3 CORN Portfolio Optimisation

For the fixed positive integer w, the CORN-based PS strategy first locates intances of historical matches via a multi dimentional correlation measure. Once these instances are identified, the CORN-based algorithm constructs a fixed portfolio vector to optimise the returns for the trading periods following each matching period in the following way:

$$\mathbf{b}^{(w,\rho)}\left(\mathbf{X}_1^{n-1}\right) = \underset{b\in\Delta_m}{\arg\max} \prod_{i\in\widehat{\mathbf{J}}_{n,i}^{w,\rho}} \langle \mathbf{b}, \mathbf{x}_i \rangle \tag{4.8}$$

As usual, if the product in equation (4.8) is void the CORN algorithm sets $\mathbf{b}$ simply as $\mathbf{b} = \left(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}\right)$. This nonparametric sequential investment strategy has been shown to effectively exploit hidden and complicated dependences of asset prices on the past evolution of the market sequences (see Li et al. (2011). Table 4.4.1 shows a pseudo code of an implementation of the CORN-based PS algorithm.

Table 4.4.1: CORN $(\mathbf{X}, w, \rho)$

```
Inputs   $w$ : the window length
         $\rho$ : correlation threshold
Output   $\mathbf{b}_t$ : expert's portfolio weights for the $t^{th}$ trading day
for      $t = 1, 2, ..., n$ do
    2   receive price relative $\mathbf{x}_t$
    3   if $t < w + 1$ then
    4       $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, \dots \frac{1}{m}\right)$
    5   end if
    6   for $i = w + 1$ to $t - 1$ do
    7       if $correl\left(\mathbf{x}_{i-w}^{i-1}, \mathbf{x}_{t-w}^{n-1}\right) \geq \rho$ then
    8           $\mathbf{J}_t^{\{w,\rho\}} = \mathbf{J}_t^{\{w,\rho\}} \cup \{i\}$
    9       end if
   10   end for
   11   if $\mathbf{N}_t == \varnothing$ then
   12       $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, \dots \frac{1}{m}\right)$
   13   else
   14   $\mathbf{b}^{(w,\rho)} = \underset{b \in \Delta_d}{\arg\max} \prod_{i \in \mathbf{J}_t^{\{w,\rho\}}} \langle \mathbf{b}, \mathbf{x}_i \rangle$
end
```

4.4.4 Combining Portfolio of Experts

The CORN-based sequential PS algorithm requires two important parameters, w and ρ, that need to be set in advance by the portfolio manager. The portfolio of experts can therefore be mixed as presented in Chapter 3.

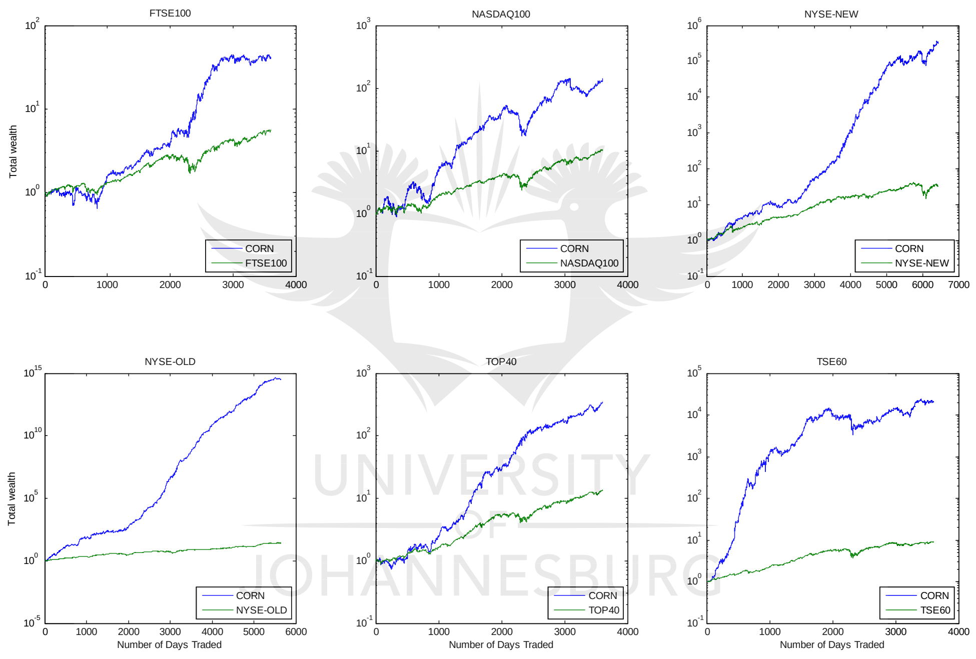
4.4.5 Empirical Results

For our empirical simulations we use slightly different parameter settings compared to those found in the original thesis of Li et al. (2011). We set the pearson correlation coefficient to

$\rho = 0.30$ and a maximum $W = 10$.

4.4.5.1 Analysis of Cumulative Wealth Figure 4.4.1 summarises the total wealth achieved by the CORN algorithm, with the usual 10 basis points of transaction costs, on the six market datasets. The market index is calculated as equal weighted Buy-and-Hold portfolio on all stock available for that particular market.

Figure 4.4.1: Cumulative Wealth of the CORN PS Algorithm



While the CORN-based PS algorithm demonstrates very good performance on old data (NYSE(O), NYSE(N)), its performance on recent and previously untested datasets seems to have been very mixed and disappointing to some extent. For example, using the FTSE100 data set, it took more than 2000 trading days before the algorithm achieved a cumulative wealth that surpassed that of the simple Buy-and-Hold. Another disappointing fact is that the recent performance on our previously untested data set is also very poor, as evidenced by the TSE60. In fact the algorithm has not been able to generate any returns at all over the past 5 years or so using the TSE60 index in Canada.

Although CORN recent performance using some previously untested data set has been disappointing, we note that the CORN algorithm has, however, achieved good CAGR over the entire testing sample on all datasets. The CAGR is around 50% and 101% using the TOP40 and TE60 datasets respectively (see Table 4.4.2). This compares very farourably with a CAGR of 20% and 17% for both market indices. Similar CAGR patterns can be observed from all other datasets, including the NASDAQ, the NYSE (O) and the NYSE (N).

Table 4.4.2: Cumulative returns over different trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.30	0.13	0.50	0.20	1.01	0.17	0.41	0.18	0.64	0.15	3.46	0.16
1Yr	0.10	0.21	0.29	0.17	-0.01	0.11	0.70	0.39	0.89	0.29	0.24	-0.01
2Yrs	0.06	0.22	0.31	0.22	0.36	0.10	0.19	0.26	0.35	-0.04	0.87	0.16
3Yrs	0.02	0.12	0.30	0.17	0.25	0.05	0.13	0.18	0.37	-0.07	2.27	0.23
4Yrs	0.18	0.16	0.27	0.19	0.32	0.09	0.27	0.21	0.30	0.03	2.28	0.19
5Yrs	0.42	0.23	0.30	0.23	0.32	0.17	0.46	0.30	0.33	0.05	2.42	0.20
7Yrs	0.44	0.12	0.46	0.16	0.12	0.09	0.23	0.17	0.51	0.08	3.02	0.20
10Yrs	0.37	0.15	0.59	0.22	0.30	0.14	0.36	0.17	0.81	0.07	4.44	0.19
15Yrs	0.37	0.13	0.57	0.21	0.57	0.15	0.38	0.18	0.93	0.07	5.80	0.13

4.4.5.2 Performance Risk Analysis Next, Table 4.4.3 shows an evaluation of some risk measures that an investor might be exposed to while using the CORN-based PS algorithm. As shown in Table 4.4.3, the CORN-based strategy has a volatility of returns that is unsurprisingly much higher than the one achieved by its respective benchmarks. At close inspection, the portfolio weight output generate by the CORN-based strategy appears to be very highly concentrated. The Sharpe Ratio generated by the CORN-based algorithms is also somewhat higher than the respective stock market indices. On balance we could argue that the CORN-based strategy is taking relatively higher risk to generate its superior outperformance.

Table 4.4.3: Risk Statistics of the CORN-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.30	0.18	0.28	0.16	0.58	0.17	0.42	0.23	0.38	0.19	0.54	0.13
α	0.01	0.00	0.06	0.00	0.25	0.00	0.12	0.00	0.14	0.00	0.53	0.00
β	1.19	1.00	1.14	1.00	1.62	1.00	1.23	1.00	1.09	1.00	1.45	1.00
γ	3.68	0.00	2.52	0.00	-1.21	0.00	-1.92	0.00	0.96	0.00	4.35	0.00
$\beta\uparrow$	1.29	1.00	1.18	1.00	1.42	1.00	1.13	1.00	1.11	1.00	1.55	1.00
$\beta\downarrow$	1.04	1.00	1.02	1.00	1.56	1.00	1.25	1.00	0.98	1.00	1.23	1.00
SR	0.78	0.37	1.33	0.78	1.37	0.59	0.87	0.53	1.32	0.44	2.90	0.64
MDD	-0.44	-0.40	-0.34	-0.30	-0.77	-0.41	-0.68	-0.47	-0.61	-0.64	-0.37	-0.37
PP	0.54	0.56	0.55	0.57	0.56	0.58	0.55	0.56	0.55	0.55	0.59	0.53
PP$\uparrow$	0.79	1.00	0.78	1.00	0.75	1.00	0.81	1.00	0.75	1.00	0.72	1.00
PP$\downarrow$	0.22	0.00	0.26	0.00	0.30	0.00	0.23	0.00	0.30	0.00	0.43	0.00

Recall that the maximum drawdown encompasses both the period from the fund's peak to the fund's valley (length), and the time from the fund's valley to a new fund high (recovery). From Table 4.4.3 , we observe that maximum drawdown on the six stock datasets are higher when compared with that achieved by their respective benchmarks. The FTSE100 and the NASDAQ experienced drawdowns of 60% and 68% over the sample period. These drawdowns are significantly higher than those experienced by those respective market indices (40% and 47%). We also note that the percentage of time the CORN-based algorithm generate positive returns (PP) is very similar to the percentage of time the varous indices generate positive returns. This suggests that the superior growth in portfolio wealth achieved by the CORN-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market, as confirmed by the Gamma coefficients and the asymmetric betas.

4.5 ANTICOR ALGORITHM (BORODIN ET AL. (2004))

4.5.1 Introduction

In its original form, the Anticor (AC) algorithm of Borodin et al. (2004) provides results on historical stock prices that show that an algorithm derived from simple heuristics can significantly outperform those that provide theoretical guarantees Unlike competing state-of-the-art algorithms that are derived from sound mathematical and statistical learning theory, the AC algorithm is derived from simple heuristics. The algorithm evaluates changes in stocks' performance by dividing the historical sequence of past returns series into equal-sized periods called windows, each with a length of w days where w is an adjustable parameter. According to the AC algorithm the wealth is transferred from recently high-performing experts to anti-correlated low-performing experts. Specifically, whenever the algorithm detects that stock i outperformed stock j during the last window, but $i's$ performance in the last window is anti-correlated to $j's$ performance in the second-to-last window ($\mu_2(i) \geq \mu_2(j)$ and $Mcorr(i,j) > 0$), it transfers wealth from stock i to stock j and calculate new portfolio weights. Borodin et al. (2004) show historical simulation results or some real-market datasets that demonstrate that the AC algorithm is indeed very robust in those datasets based solely on the mean reversion principal.

4.5.2 Anticor Algorithm Design

For a window length w, we consider LX_1 and LX_2 as two $w \times m$ matrices over two consecutive time windows, which we compute as follows:

$$LX_1 = \left(\log\left(X_{t-2w+1}\right), ..., \log\left(X_{t-w}\right)\right)^T \tag{4.9}$$

and

$$LX_2 = \left(\log\left(X_{t-w+1}\right), ..., \log\left(X_t\right)\right)^T \tag{4.10}$$

The j^{th} column of LX_k is denoted by $LX_k(j)$ and simply tracks the performance of stock j in window k where $k = 1, 2$. Let $\mu_k(j)$ be the mean of $LX_k(j)$ and $\sigma_k(j)$ be the corresponding standard deviation. The cross-covariance matrix between the columns vectors of LX_k is defined as follows

$$Mcov\left(i,j\right)=\frac{1}{w-1}\left[LX_1\left(i\right)-\mu_1\left(i\right)\right]^T\left[LX_2\left(j\right)-\mu_2\left(j\right)\right] \tag{4.11}$$

and the corresponding cross-correlation matrix is given by

$$MCorr\left(i,j\right)=\begin{cases}\frac{MCov(i,j)}{\sigma_1(i)\sigma_2(j)}, & \sigma_1\left(i\right),\sigma_2\left(j\right)\neq 0\\ 0 & otherwise\end{cases} \tag{4.12}$$

The reversion to mean strategy of Borodin et al. (2004) states that if $\mu_2\left(i\right)\geq\mu_2\left(j\right)$ and $Mcorr\left(i,j\right)>0$ the proportion of wealth to be moved from stock i to stock j is defined as:

$$claim_{i\to j}=Mcorr\left(i,j\right)+\max\left(-MCorr\left(i,i\right),0\right)+\max\left(-MCorr\left(j,j\right),0\right) \tag{4.13}$$

Therefore whenever the anticor algorithm detects that stock i has outperformed stock j during the last window but $i's$ performance in the last window is not anti-correlated to $j's$ performance in the second-to-last window ($\mu_2\left(i\right)\geq\mu_2\left(j\right)$ and $Mcorr\left(i,j\right)>0$), it transfers wealth from stock i to stock j and calculate new portfolio weights. The simple logic here is that there will be price reversal in the direction of the underperforming stock.

From the reversal to the mean conditions the model calculates the transfers of stock i to stock j as

$$transfer_{i\to j}=b_{t-1}\left(i\right)\frac{claim_{i\to j}}{\sum_j claim_{i\to j}} \tag{4.14}$$

Using these transfer values, the portfolio is defined to be:

$$b_t\left(i\right)=b_{t-1}\left(i\right)+\sum_{i\neq j}\left(transfer_{j\to i}-transfer_{i\to j}\right) \tag{4.15}$$

In Table 4.5.1, we show a pseudo code implementation of the proposed Anticor (AC).

Table 4.5.1: Anticor $(\mathbf{X}, w)$

```
$w$: window sise
$\mathbf{b}_0$: initial portfolio weights $\mathbf{b}_0 = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$
$\mathbf{X}$: matrix of price relatives
for $t = 1, 2, ...$
    1  Return the current portfolio $\mathbf{b}_t$ if $t < 2w$
    2  compute $\mathbf{LX}_1 = \left(\log\left(X_{t-2w+1}\right), ..., \log\left(X_{t-w}\right)\right)^T$
    3  compute $\mathbf{LX}_2 = \left(\log\left(X_{t-w+1}\right), ..., \log\left(X_t\right)\right)^T$
    4  compute $\boldsymbol{\mu}_1 = average\left(LX_1\right)$ and $\boldsymbol{\mu}_2 = average\left(LX_2\right)$
    5  compute $Mcov\left(i,j\right) = \frac{1}{w-1}\left[LX_1\left(i\right) - \mu_1\left(i\right)\right]^T\left[LX_2\left(j\right) - \mu_2\left(j\right)\right]$
    6  compute $MCorr\left(i,j\right) = \begin{cases} \frac{MCov(i,j)}{\sigma_1(i)\sigma_2(j)}, & \sigma_1\left(i\right), \sigma_2\left(j\right) \neq 0 \\ 0 & otherwise \end{cases}$
    7  Initialise $claim_{i\to j} = 0$
    8  if $\mu_2\left(i\right) \geq \mu_2\left(j\right)$ and $Mcorr\left(i,j\right) > 0$
    9  $claim_{i\to j} = Mcorr\left(i,j\right) + \max\left(-MCorr\left(i,i\right), 0\right) + \max\left(-MCorr\left(j,j\right), 0\right)$
    10 $transfer_{i\to j} = b_{t-1}\left(i\right)\frac{claim_{i\to j}}{\sum_j claim_{i\to j}}$
    11 $b_t\left(i\right) = b_{t-1}\left(i\right) + \sum_{i\neq j}\left(transfer_{j\to i} - transfer_{i\to j}\right)$
end
```

4.5.3 Expert Combination

The AC-based sequential PS algorithm requires one important parameter , the window length w, that needs to be set in advance by the portfolio manager. Because historical simulations demonstrate that the performance of the AC algorithm critically depends on the choice of this window sise parameter we form the expert combination in the usual way.

4.5.4 Empirical results

This section presents numerical results obtained by applying the AC-based algorithm to six financial market datasets described in Chapter 3. As usual, the back-testing experiments in this section will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed on a daily basis. For our simulatoin we set the maximum window sise $W = 30$.

4.5.4.1 Analysis of the Anticor Cumulative Wealth The first experiment evaluates the compounded wealth achieved by the AC-based learning-to trade algorithm, including a 10 basis points transaction cost over the entire sample period. Figure 4.5.1 summarises the total wealth achieved by the AC algorithm on the six market datasets. As in previous cases, the market index is calculated as equal weighted Buy-and-Hold portfolio on all stock available for that particular market.

Figure 4.5.1: Cummulative Wealth of the Anticor algorithm

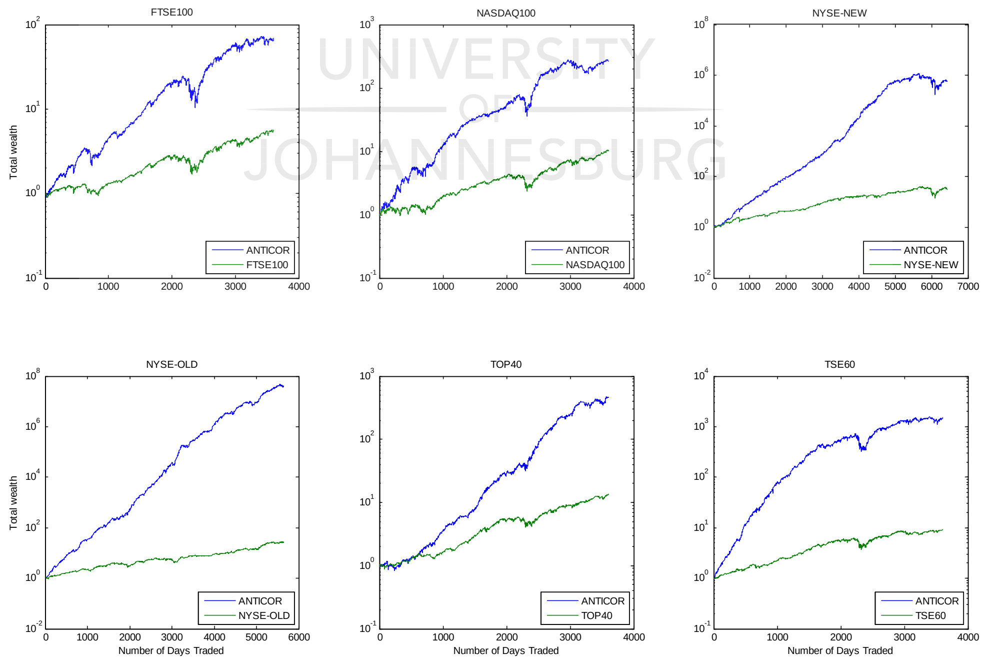


Figure 4.5.1 shows that the AC-based PS algorithm demonstrates very good performance on old data (NYSE(O), NYSE(N)) as well as on more recent and previously untested datasets. For example, on the South African TOP40 index data set, the total wealth achieved by the AC strategy impressively increases from \$1 to almost \$454. This wealth growth is much higher than the \$13.6 achieved by the market index over the same 15-year periods. During that period, the best stock generates \$87.8 and the best constantly rebalanced portfolio in hindsight only achieves a growth in wealth of \$115. We also notice that all other datasets display similar impressive growth in portfolio wealth. Figure 4.5.1 also shows that, from a start of \$1, the algorithm achieved a cumulative wealth of $\$4.17 \times 10^7$ in the NYSE(O), \$682 using the FTSE100, \$2670 using the NASDAQ.

In Table 4.5.2 we show the cumulative returns achieved by the AC algorithm over some selected trading periods. The AC algorithm also displays a very impressive CAGR of around 53% and 67% using the TOP40 and TE60 datasets respectively (see Table 4.5.2). This compares very farourably with a CAGR of 21% and 18% for both market indices. Similar CAGR can be observed from all other datasets, including the NASDAQ, the NYSE (O) and the NYSE (N).

Table 4.5.2: Cumulative returns over selected trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.34	0.13	0.53	0.20	0.67	0.17	0.48	0.18	0.68	0.15	1.19	0.16
1Yr	0.01	0.21	0.18	0.17	0.01	0.11	0.29	0.39	0.69	0.29	0.25	-0.01
2Yrs	0.17	0.22	0.16	0.22	0.10	0.10	0.14	0.26	-0.12	-0.04	0.50	0.16
3Yrs	0.13	0.12	0.26	0.17	0.11	0.05	0.07	0.18	-0.15	-0.07	0.73	0.23
4Yrs	0.20	0.16	0.39	0.19	0.14	0.09	0.13	0.21	-0.06	0.03	0.57	0.19
5Yrs	0.34	0.23	0.58	0.23	0.33	0.17	0.39	0.30	-0.01	0.05	0.65	0.20
7Yrs	0.23	0.12	0.55	0.16	0.20	0.09	0.30	0.17	0.13	0.08	0.81	0.20
10Yrs	0.29	0.15	0.59	0.22	0.32	0.14	0.32	0.17	0.43	0.07	0.89	0.19
15Yrs	0.29	0.13	0.62	0.21	0.45	0.15	0.39	0.18	0.56	0.07	1.07	0.13

Despite this impressive performance on all datasets and over the full sample, recent performance of the AC algorithm has been subdued. The AC-based PS algorithm has been pedestrian in recent times, with a 1% gain in 2013 and a 20% gain in 2012 using the TSE60 index in Canada. The same pattern can be seen in the FTSE100 in the UK. where the algorithm achieves an annualised compounded growth rate that was much less than its benchmark algorithm in 2013.

4.5.4.2 Anticor Performance Risk Analysis Table 4.5.3 shows some of the AC risk measures for the six datasets under consideration. From the results, we see that the AC-based strategy has a volatility of returns that is higher than the one achieved by its respective benchmarks. However, the Sharpe Ratio generated by the AC-based algorithms are also significantly higher than the respective stock market indices. This is an indication that the AC-based strategy generates much better risk-adjusted returns.

Table 4.5.3: Risk Statistics of the AC-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALGC	MKT
σ	0.29	0.18	0.25	0.16	0.32	0.17	0.34	0.23	0.37	0.19	0.30	0.13
α	0.04	0.00	0.07	0.00	0.14	0.00	0.08	0.00	0.13	0.00	0.24	0.00
β	1.22	1.00	1.11	1.00	1.31	1.00	1.13	1.00	1.22	1.00	1.29	1.00
γ	1.85	0.00	2.34	0.00	-0.61	0.00	0.82	0.00	1.61	0.00	1.88	0.00
$\beta\uparrow$	1.28	1.00	1.19	1.00	1.33	1.00	1.18	1.00	1.31	1.00	1.31	1.00
$\beta\downarrow$	1.23	1.00	1.07	1.00	1.31	1.00	1.09	1.00	1.13	1.00	1.28	1.00
SR	0.93	0.37	1.54	0.78	1.55	0.59	1.10	0.53	1.40	0.44	2.53	0.64
MDD	-0.58	-0.40	-0.30	-0.30	-0.55	-0.41	-0.54	-0.47	-0.85	-0.64	-0.32	-0.37
PP	0.57	0.56	0.58	0.57	0.58	0.58	0.56	0.56	0.56	0.55	0.56	0.53
PP$\uparrow$	0.83	1.00	0.81	1.00	0.82	1.00	0.84	1.00	0.80	1.00	0.79	1.00
PP$\downarrow$	0.23	0.00	0.26	0.00	0.25	0.00	0.20	0.00	0.27	0.00	0.29	0.00

Further, we observe that maximum drawdown on the six stock datasets are somewhat higher when compared with that achieved by its respective benchmarks. We also note that the percentage of time the AC-based algorithm generate positive returns (PP) is very similar to the percentage of time the varous indices generate positive returns. However, when the market index was up, the AC-based algorithm generated positive returns more than 80% of the time on all datasets. This suggests that the value add of the AC-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market. This argument is well supported by the positive sign and significant value of the estimated γ coefficients for the majority of data analysed.

4.5.4.3 Anticor Brokerage Costs Analysis Figure 4.5.2 shows the performance of the AC algorithm net of brokerage commissions against some benchmark strategies on our six datasets.

Figure 4.5.2: Transaction Cost Analysis for Anticor Algorithm

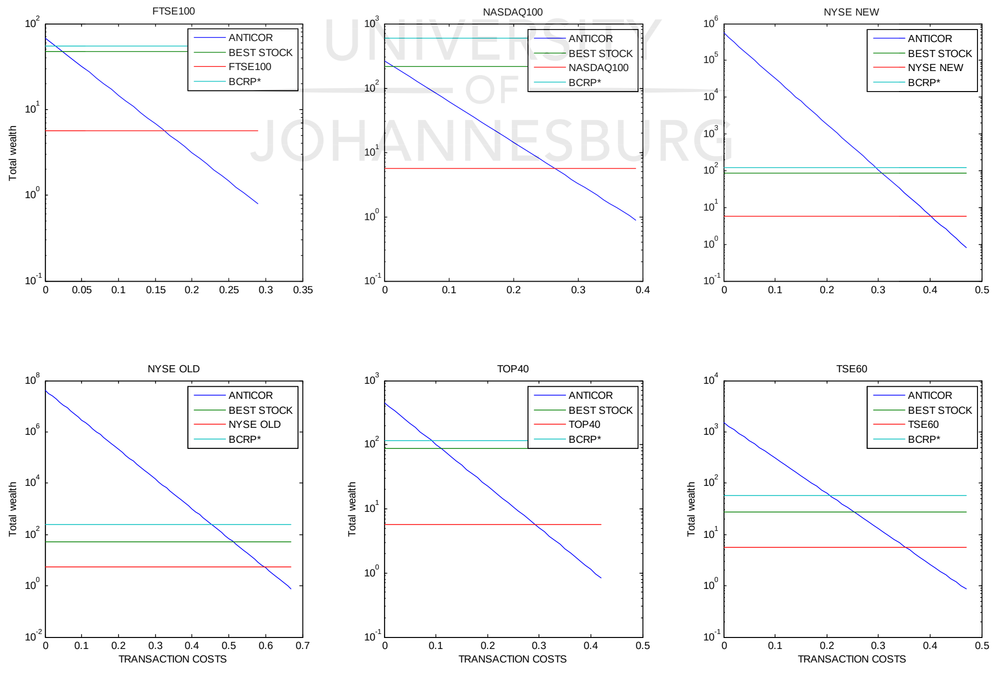


Figure 4.5.2 clearly demonstrates that the AC investment algorithm can tolerate reasonable proportional commission rates and still beat competing benchmarks, including the best stock, the equally weighted market index and best constantly rebalanced portfolio in hindsight called the BCRP*. The graphs in Figure 4.5.2 depict the total returns of the AC algorithm for a varying proportional commission factor, $c = 0.1\%, 0.2\%$; For example, with a commission cost of $c = 0.1\%$ or 10 basis points (10 bps), the algorithm still beat the best stock, the market and the BCRP* portfolio in three out of the six markets data under consideration. In fact, even with $c < 0.15\%$ our algorithm beats all its respective market indices on all datasets.

4.5.5 Conclusion

In summary, the simulation results have demonstrated that the AC-based PS algorithm achieves very impressive cumulative wealth growth on all datasets, even after accounting for moderate brokerage commissions. These encouraging results show that the AC-base PS algorithm is capable of achieving an excellent trade-off between return and risk. In our view the AC-based PS algorithm is indeed a very robust investment strategy that can effectively exploit the mean reversion properties of various stock market data.

Despite this impressive empirical performance, an extensive body of behavioural finance literature has documented that price reversals are hardly the only feature at play in equity markets. It has been argued that price momentum and reversals tend to coexist in world stock markets in the short term. In a comprehensive investigation, Conrad and Kaul (1998) find both momentum and contrarian profits in the U.S. market, depending on the time horison investigated. Balvers and Wu (2006) also demonstrate that mean reversion and momentum can simultaneously occur on the same set of assets in 18 developed countries. It is therefore possible that in the presence of both price reversal and price continuation, the original AC algorithm will fail to perform optimally. To correct this shortcoming we provide in Chapter 5 an important modification to the AC algorithm that can deal with both momentum as well as reversal in a comprehensive way.

4.6 CONFIDENCE WEIGHTED MEAN REVERSAL (LI ET AL. (2013))

4.6.1 Introduction

Li et al. (2013) proposed the Confidence Weighted Mean Reversion (CWMR) PS algorithm. The proposed strategy is inspired by Confidence Weighted learning originally proposed for classification problems (Dredze et al. 2008; Dredze et al. 2010; Crammer et al. 2008; Crammer et al. 2009). The basic idea of Confidence Weighted algorithm is to maintain a Gaussian distribution for the classifier, and to sequentially update the classifier distribution according to the Passive Aggressive Online learning algorithm. CWMR helps exploit not only the first order information but also the second order portfolio information in terms of the variance of portfolio weights.

4.6.2 CWMR Algorithm Design

On the i^{th} day the idea of the CWMR is to model the portfolio vector $\mathbf{b}_i$ as a Gaussian distribution, $\mathbf{b}_i \backsim \mathbb{N}\left(\mu, \Sigma\right)$. Σ is a diagonal covariance matrix with nonzero diagonal elements σ^2 and zero for off-diagonal elements. Furthermore, Σ could be understood as the confidence the portfolio manager has in the portfolio mean value $\boldsymbol{\mu}$. After the price relative $\mathbf{x}_i$ is revealed, the portfolio increases its wealth by a factor of $\mathbf{b}_i\mathbf{x}_i$. The portfolio daily total return can be viewed as a random variable of a Gaussian distribution, $\mathbf{D} \backsim \mathbb{N}\left(\boldsymbol{\mu}.\mathbf{x}_i, \mathbf{x}_i^\top \boldsymbol{\Sigma} \mathbf{x}_i\right)$.

For the mean reversion principle to apply the idea behind the CWMR is to select a portfolio in such a way that $D \leq \varepsilon$ in a probabilistic sense. The portfolio manager must adjust the distribution to ensure the probability of a profitable portfolio is higher than a confidence level $\theta \in [0, 1]$. Therefore

$$\Pr_{\mathbf{b}\backsim\mathbb{N}(\boldsymbol{\mu},\Sigma)} \left(D \leq \varepsilon\right) = \Pr_{\mathbf{b}\backsim\mathbb{N}(\boldsymbol{\mu},\Sigma)} \left(\mathbf{b}.\mathbf{x}_i \leq \varepsilon\right) \geq \theta. \tag{4.16}$$

The intuition behind this is quite simple. If the expected return using the i^{th} price relative is less than a threshold with high probability, the actual return for the $i^{th} + 1$ trading day is most likely to be high with correspondingly high probability, since the price relative tends to reverse. Then, the algorithm chooses the distribution closest (in the KL divergence sense) to the current distribution $\mathbb{N}(\boldsymbol{\mu}_i, \Sigma_i)$. On the $i^{th}+1$ trading day, the algorithm sets the parameters of the distribution by solving the following optimisation problem

$$\begin{aligned} \left(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}\right) &= \underset{\mu,\Sigma}{\arg} \max D_{KL}\left(\mathbb{N}(\boldsymbol{\mu}, \Sigma) \| \mathbb{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\ \text{s.t} \quad & \operatorname{Pr}_{\mathbf{b} \backsim \mathbb{N}(\boldsymbol{\mu}, \Sigma)}\left(\boldsymbol{\mu}.\mathbf{x}_i \le \varepsilon\right) \ge \theta \\ & \boldsymbol{\mu} \in \Delta_m \end{aligned} \tag{4.17}$$

After some simple manipulations Li et al. (2013) show that the final optimisation problem becomes

$$\begin{aligned} \left(\mu_{i+1}, \Sigma_{i+1}\right) &= \underset{\mu,\Sigma}{\arg} \max \tfrac{1}{2}\left[\log\left(\tfrac{\det \Sigma_i}{\det \Sigma}\right) + \mathbf{tr}\left(\Sigma_i^{-1}\Sigma\right) + \left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right)^{\top} \Sigma_i^{-1}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right)\right] \\ s.t \quad \varepsilon - \log\left(\boldsymbol{\mu}.\mathbf{x}_i\right) &\ge \Phi^{-1}(\theta)\, \mathbf{x}_i^{\top} \Sigma \mathbf{x}_i \\ & \boldsymbol{\mu}.\mathbf{1} = \mathbf{1},\ \boldsymbol{\mu} \ge 0 \end{aligned} \tag{4.18}$$

Table 4.6.1 shows the pseudo code of the CWMR algorithm

Table 4.6.1: CWMR $(\mathbf{X},\phi,\varepsilon)$

```
Inputs      $\mathbf{X}$ : the matrix of price relative
            $\phi$ : the confidence parameter
            $\varepsilon$ : the sensitivity parameter
Initialise  $\boldsymbol{\mu}_0 = \frac{1}{m}1, \boldsymbol{\Sigma}_0 = \frac{1}{m^2}I, \mathbf{S}_0 = 1$
Output      $\mathbf{b}$ : expert's portfolio weights
for         $i = 1, 2, ..., n$ do
    1       receive $\mathbf{x}_t$
    2       Calculate the daily return and cumulative return:
            $\mathbf{S}_i = \mathbf{S}_{i-1}(\mathbf{b}_i\mathbf{x}_i)$
    3       Calculate the following variables
            $\mathbf{M}_i = \boldsymbol{\mu}_i\mathbf{x}_i, \mathbf{V}_i = \mathbf{x}_i^\top\boldsymbol{\Sigma}_i\mathbf{x}_i, \overline{\mathbf{x}}_i = \frac{\mathbf{1}^\top\boldsymbol{\Sigma}_i\mathbf{x}_i}{\mathbf{1}^\top\boldsymbol{\Sigma}_i\mathbf{1}}$
    4       Update the portfolio distribution:
            $\boldsymbol{\lambda}_{i+1}$ as above
            $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \boldsymbol{\lambda}_{i+1}\boldsymbol{\Sigma}_i\frac{\mathbf{x}_i - \overline{\mathbf{x}}_i\mathbf{1}}{\mathbf{M}_i}$
            $\boldsymbol{\Sigma}_{i+1} = \left(\boldsymbol{\Sigma}_i^{-1} + 2\boldsymbol{\lambda}_{i+1}\boldsymbol{\phi}\,\mathbf{diag}^2(\mathbf{x}_i)\right)^{-1}$
    5       Normalise $\boldsymbol{\mu}_{i+1}$ and $\boldsymbol{\Sigma}_{i+1}$
            $\mathbf{b}^{(\phi,\varepsilon)} = \underset{\boldsymbol{\mu}\in\boldsymbol{\Delta}_m}{\arg\min}\left\|\boldsymbol{\mu} - \boldsymbol{\mu}_{i+1}\right\|^2$
            $\boldsymbol{\Sigma}_{i+1} = \frac{\boldsymbol{\Sigma}_{i+1}}{m^2\mathbf{Tr}(\boldsymbol{\Sigma}_{i+1})}$
end
```

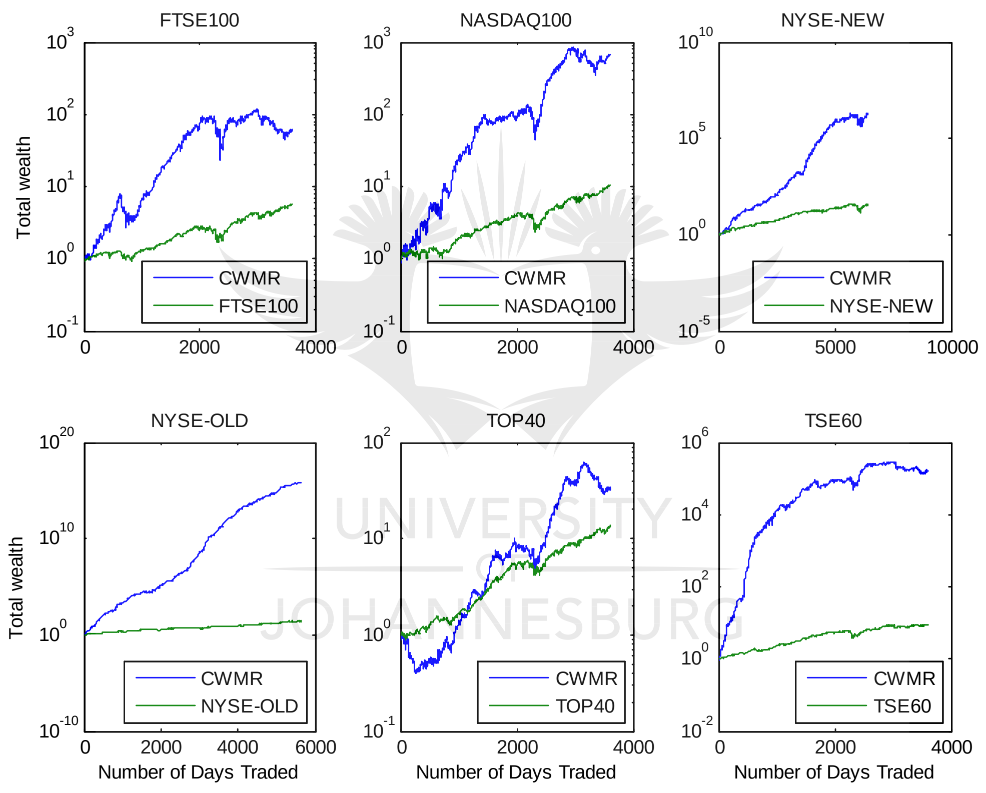
4.6.3 Combining Portfolio of Experts

The CWMR-based sequential PS algorithm requires two important parameters (ϕ, ε) that need to be set in advance by the portfolio manager. In practice, Li et al. (2012) and our own historical simulations show that the CWMR-based PS algorithm is very robust to the parameter ϕ and we will therefore set $\phi = 2$ for the remainder of the thesis. Setting the confidence parameter to a constant simply means that the performance of the CWMR algorithm critically depends on the choice of the sensitivity parameter ε. We therefore form the expert combination as seen in Section 3.4.

4.6.4 Empirical Results

4.6.4.1 Analysis of Cumulative Wealth As usual our first experiment evaluates the compounded wealth achieved by the CWMR-based algorithm over the full sample. Figure 4.6.1 summarises the total wealth achieved by the CWMR algorithm on the six market datasets. Here again, the market index is calculated as equal weighted Buy-and-Hold portfolio on all stock available for that particular market.

Figure 4.6.1: Performance of the CWMR-based algorithm



One striking observation that could be drawn from Figure 4.6.1 is that while the CWMR-based PS algorithm demonstrates very good performance on old data (NYSE(O), NYSE(N)), the performance is very poor using recent and previously untested datasets. For example, the total wealth achieved by the CWMR on the NYSE(O) and the NYSE(N) strategies impressively increased from \$1 to an astronomical 6.63×10^{15} and 1.73×10^{6} respectively. On more recent datasets, the performance is very disappointing, with a growth in compounded wealth that appears very volatile and has underperformed market indices in recent years. Even in the case of the TSE60 where the CWMR-base PS algorithm has generated an impressive compounded wealth of 1.64×10^5 over the long-term, its short term performance has been disappointing at best. Over the past 4 years the model has lost about 30% of wealth value while the market has increased by about 40%. The post 2008 financial crisis performance has been truly disappointing for this algorithm as it continues to experience losses accross all datasets. It is therefore difficult to justify the claim made by Li et al. (2013) that the CWMR is a very robust investment strategy that could be effectively implemented by portfolio managers. It is clear from Figure 4.6.1 that the CWMR-based algorithm results can not be easily generalised to multiple data sets. To confirm our finding we ran the CWMR on many other European stock market data sets. The countries considered included the Nordic countries (Denmark, Finland, Norway, Sweden), Germany, the United Kingdom, Switzerland, France, France, Italy, Portugal and Spain. For all these countries the performance of the CWMR-based algorithm has been disappointing at best, especially in recent years.

Table 4.6.2: Cumulative returns over different trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.33	0.13	0.27	0.20	1.31	0.17	0.58	0.18	0.76	0.15	4.08	0.16
1Yr	0.07	0.21	-0.23	0.17	-0.03	0.11	0.57	0.39	1.64	0.29	0.66	-0.01
2Yrs	-0.09	0.22	-0.20	0.22	-0.08	0.10	0.14	0.26	0.14	-0.04	0.73	0.16
3Yrs	-0.15	0.12	-0.09	0.17	-0.14	0.05	0.02	0.18	0.00	-0.07	1.52	0.23
4Yrs	-0.07	0.16	0.19	0.19	-0.09	0.09	0.19	0.21	0.18	0.03	1.49	0.19
5Yrs	0.03	0.23	0.42	0.23	0.20	0.17	0.56	0.30	0.21	0.05	1.74	0.20
7Yrs	0.02	0.12	0.25	0.16	0.14	0.09	0.33	0.17	0.29	0.08	2.16	0.20
10Yrs	0.22	0.15	0.35	0.22	0.24	0.14	0.37	0.17	0.67	0.07	3.69	0.19
15Yrs	0.21	0.13	0.40	0.21	0.49	0.15	0.49	0.18	0.78	0.07	4.80	0.13

In addition to the final cumulative wealth and the CAGR, we are also interested in examining how the cumulative wealth changes over different sub-samples. Table 4.6.2 shows the CAGR generated by the proposed CWMR algorithm over various sub-periods of trading.

In the case of the NYSE(O) and NYSE(N), we can see that the proposed CWMR strategy consistently surpasses its benchmarks returns over the entire period but this can not be generalised to the rest of our datasets.

4.6.4.2 Performance Risk Analysis Table 4.6.3 shows the various risk measures on the six datasets. As shown in this Table, the CWMR-based strategy has a volatility of returns that is significantly higher than the one achieved by its respective benchmarks. We also notice that both the beta and the drawdowns are much higher with the CWMR-based algorithm when compared to earlier Online learning procedures. To us this is an indication that the CWMR-based strategy generates much worse risk adjusted returns on these new datasets.

Table 4.6.3: Risk Statistics of the CWMR-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.45	0.18	0.34	0.16	0.54	0.17	0.52	0.23	0.47	0.19	0.50	0.13
α	0.03	0.00	0.02	0.00	0.30	0.00	0.10	0.00	0.16	0.00	0.58	0.00
β	1.50	1.00	1.26	1.00	1.51	1.00	1.37	1.00	1.28	1.00	1.45	1.00
γ	2.97	0.00	0.42	0.00	-1.37	0.00	1.38	0.00	2.16	0.00	4.17	0.00
$\beta\uparrow$	1.58	1.00	1.29	1.00	1.41	1.00	1.47	1.00	1.38	1.00	1.61	1.00
$\beta\downarrow$	1.43	1.00	1.20	1.00	1.47	1.00	1.28	1.00	1.08	1.00	1.38	1.00
SR	0.70	0.37	0.68	0.78	1.69	0.59	1.00	0.53	1.28	0.44	3.35	0.64
MDD	-0.75	-0.40	-0.62	-0.30	-0.56	-0.41	-0.68	-0.47	-0.78	-0.64	-0.31	-0.37
PP	0.55	0.56	0.56	0.57	0.57	0.58	0.55	0.56	0.54	0.55	0.57	0.53
PP$\uparrow$	0.77	1.00	0.76	1.00	0.75	1.00	0.78	1.00	0.74	1.00	0.73	1.00
PP$\downarrow$	0.27	0.00	0.28	0.00	0.32	0.00	0.26	0.00	0.30	0.00	0.38	0.00

4.6.4.3 CWMR Brokerage Costs Analysis Figure 4.6.2 shows the performance of the CWMR algorithm net of brokerage commissions against some benchmark strategies for our six datasets.

Figure 4.6.2: Brokerage Cost Analysis for the CWMR PS Strategy

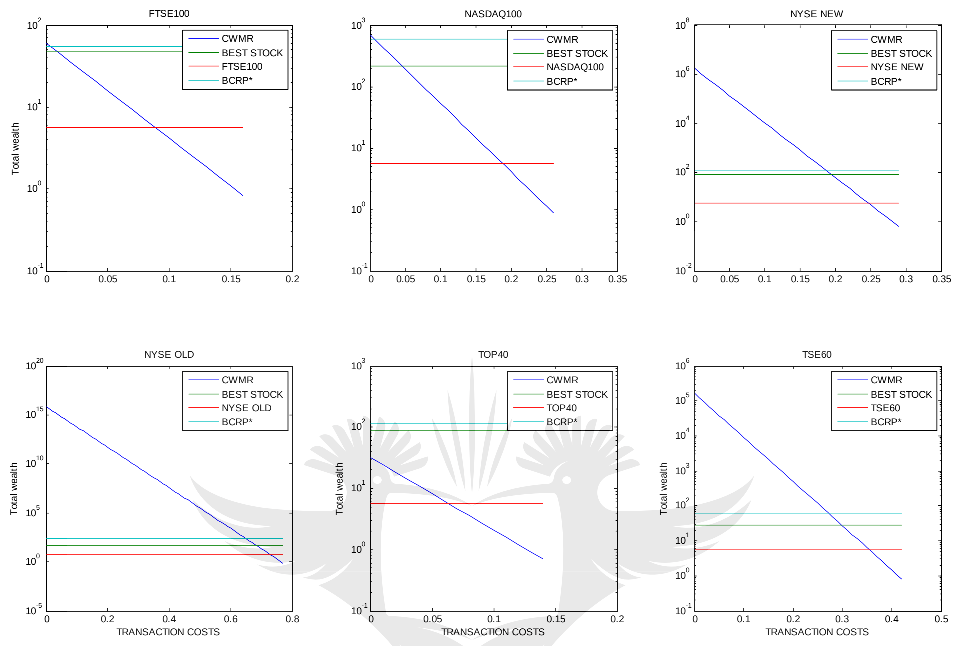


Figure 4.6.2 seems to show that the CWMR investment algorithm can tolerate moderate proportional commission rates and still beat competing benchmarks algorithms on some datasets. Although the CWMR trading algorithm beats the overall stock market indices for all brokerage commissions below 10 basis points, the comparison with the best stock and the best constantly rebalanced portfolio (in hindsight) is very ambiguous. Only in three out of six datasets - the NYSE-OLD, the NYSE-NEW and the TSE60 - does the CWMR performace compare favourably with those of the best stock and the BCRP* for moderate transaction costs. Therefore, the performance of the CWMR trading algorithm appears to be data dependent and any generalisation to different datasets need to be taken with due care. In fact, in the next chapter we proposed a simple heuristic-based modifications of this algorithm that improves the performance of this base algorithm, quite dramatically, in some cases, in all datasets and for all periods considered.

4.7 PASSIVE-AGGRESSIVE MEAN REVERSION (LI ET AL. (2012))

4.7.1 Introduction

The last algorithm surveyed in this chapter is the Passive Aggressive Mean Reversion (PAMR) strategy of Li et al. (2012). PAMR exploits the mean reversion property of stock prices by adapting the Online Passive Aggressive (PA) learning algorithm of Shalev-Shwartz et al. (2003) and Crammer et al. (2006). The Online Passive Aggressive learning was originally proposed for classification tasks. Loosely speaking, the basic idea of Passive Aggressive for classification is to passively keep the previous solution if the loss is zero, while the algorithm aggressively updates the solution whenever the suffering loss is nonzero. Using the theoretical foundations of the Online passive aggressive learning (Crammer et al. 2006), PAMR's key idea is to design a loss function in order to take advantage of the mean reversion property. The loss function works as follows; if the expected return based on last price relative is larger than a threshold, the loss will linearly increase; otherwise, the loss is zero. Given a portfolio vector $\mathbf{b}$ and a price relative vector $\mathbf{x}_t$, the PAMR mathematically defines the following loss function for the t^{th} trading day

$$l_{\varepsilon}^{t}\left(\mathbf{b};\mathbf{x}_t\right)=\begin{cases}0 & \mathbf{b}.\mathbf{x}_t\leq\varepsilon\\ \mathbf{b}.\mathbf{x}_t-\varepsilon & otherwise\end{cases}\tag{4.19}$$

The intuition behind this loss function is quite simple and works as follows; Firstly, if the portfolio daily return is below a certain threshold, the algorithm tries to keep the previous portfolio such that it passively reverts to the mean to avoid unecessary churn. Secondly, if the portfolio daily return is above the threshold, the algorithm will actively rebalance the portfolio to ensure that the expected portfolio daily return is below the threshold in the belief that the stock price relatives will revert in the next trading day. In the formulation above, $0\leq\varepsilon\leq1$ is the mean reversion (also called "sensitivity") parameter. Therefore, according to the PAMR paradigm the next period weight vector is derived via the following optimisation problem

$$\mathbf{b}_{t+1}=\begin{cases}\underset{b\epsilon\Delta_m}{\arg}\ \min\frac{1}{2}\left\|\mathbf{b}-\mathbf{b}_t\right\|^2\\ st \qquad l_{\varepsilon}\left(\mathbf{b};\mathbf{x}_t\right)=0\end{cases}\tag{4.20}$$

4.7.2 Algorithm Design and Closed-Form Solution

The formulation in Equation 4.20 attempts to find an optimal portfolio by minimising the deviation from the last portfolio $\mathbf{b}_t$ under the condition of satisfying the constraint of zero loss. On the one hand, the model passively keeps the last portfolio, that is, $\mathbf{b}_{t+1} = \mathbf{b}_t$ whenever the portfolio's daily return is below the threshold ε. In that case the model anticipates a high likelihood of mean reversal in the next period; therefore, avoiding unecessary portfolio rebalancing and transaction costs. On the other hand, whenever the loss is nonzero, the algorithm aggressively updates the solution by forcing it to strictly satisfy the constraint $l_\varepsilon\left(\mathbf{b}_{t+1}; \mathbf{x}_t\right)$. This simply means that when the portfolio's daily return is above the threshold, ε, the algorithm actively rebalances the portfolio weights. Li et al. (2012) proposed the following closed-form solution for the PAMR algorithm.

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \tau_t\left(\mathbf{x}_t - \overline{\mathbf{x}}_t 1\right) \tag{4.21}$$

where

$$\overline{\mathbf{x}}_t = \frac{\mathbf{x}_t 1}{m}, \text{ and } \tau_t = \frac{l_\epsilon^t}{\left\|\mathbf{x}_t - \overline{\mathbf{x}}_t 1\right\|^2} \tag{4.22}$$

The Passive Aggressive Mean Reversion (PAMR) algorithm of Li et al. (2012) has been proven to be a very robust investment strategy, as it exploits the mean reversion property of financial markets. Table 4.7.1 displays the pseudo code of PAMR algorithm.

Table 4.7.1: PAMR $(\mathbf{x},\varepsilon)$

```
Inputs   $\varepsilon$ : the sensitivity parameter
Output   $\mathbf{b}$ : expert's portfolio weights
for      $t = 1, 2, ..., n$ do
    1    receive $\mathbf{x}_t$
    2    Suffer a loss $\mathbf{l}_{\varepsilon}^{t} = \max\left(\mathbf{0}, \mathbf{b}_t\mathbf{x}_t - \boldsymbol{\varepsilon}\right)$
    3    Set parameter $\tau_t = \frac{\mathbf{l}_{\varepsilon}^{t}}{\|\mathbf{x}_t - \overline{\mathbf{x}}_t\mathbf{1}\|^2}$
    4    Set parameter $\overline{\mathbf{x}} = \frac{\mathbf{x}_t}{\mathbf{m}}$
    5    Update portfolio weights $\mathbf{b}_{t+1} = \mathbf{b}_t - \tau_t\left(\mathbf{x}_t - \overline{\mathbf{x}}_t\mathbf{1}\right)$
end
```

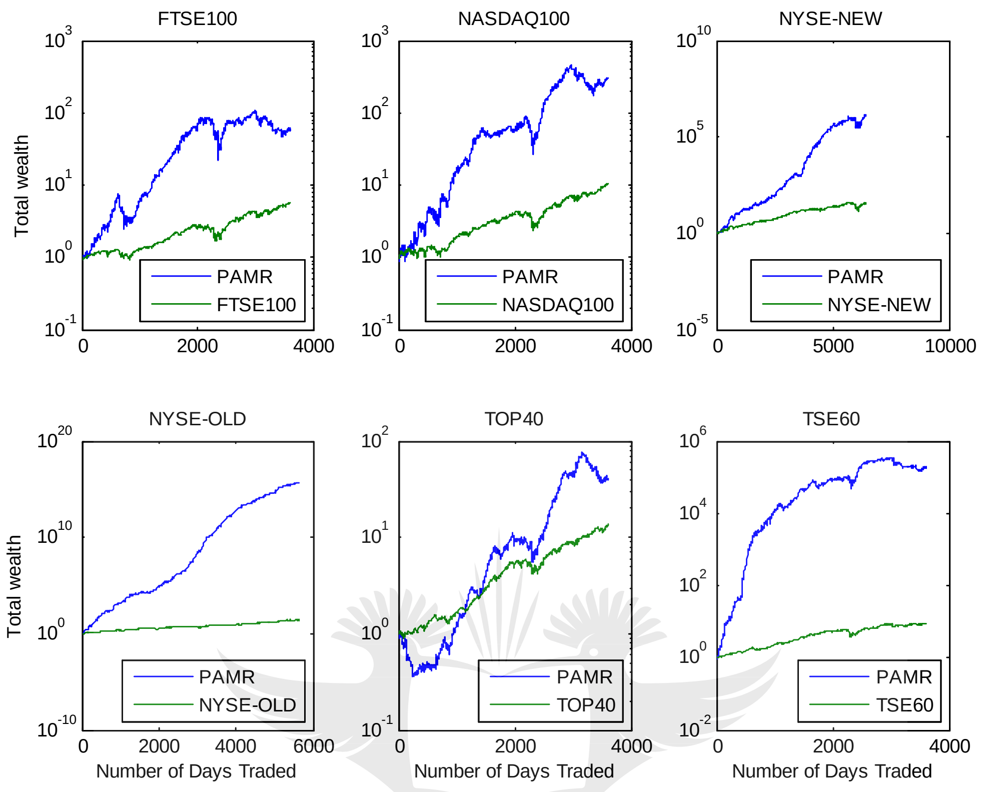
4.7.3 Expert Aggregation

The PAMR-based sequential PS algorithm requires one important parameter ε, the mean reversion parameter, that needs to be set in advance by the portfolio manager. Because the performance of the algorithm depends on an appropriate choice of this parameter and this performance can fluctuate significantly depending on which pairs of parameter one chooses our approach is to uniformally allocate the initial capital amongst all the various discretised ε parameters on the first day and never rebalance afterward. This expert combination strategy is very similar to those implemented so far in this chapter.

4.7.4 Empirical Results

4.7.4.1 Analysis of Cumulative Wealth Figure 4.7.1 shows the total wealth achieved by the PAMR algorithm for the six market datasets. As usual, the market portfolio is calculated as equal weighted Buy-and-Hold portfolio on all stock available for that particular market.

Figure 4.7.1: Cumulative Wealth Growth of the PAMR algorithm



One striking observation from the historical simulations is that the CWMR and PAMR display very similar portfolio growth in wealth despite the differences in their respective algorithms. In fact, a simple pearson correlation coefficient between the returns series generated by the two Online PS strategies exceed 80% on all datasets. Therefore the conclusions drawn earlier for the CWMR are directly applicable to the PAMR-based PS algorithm. As in the CWMR-based algorithm, the PAMR impressive performance on older datasets (NYSE(O), NYSE(N)) has not carried over on recent and previously untested datasets. In fact, on more recent datasets, the performance is very disappointing with a compounded wealth that appears very volatile and has underperformed market indices in recent years. Even in the case of the TSE60 where the PAMR-base PS algorithm has generate and impressive compounded wealth of 1.64×10^5 over the long-term, its short term performance has been disappointing. Over the past 4 years the model has lost about 30% of wealth while the market has increased by about 40%. It is indeed very difficult to see this strategy as very robust investment strategy that could be effectively implemented by portfolio managers. In the next chapter we propose simple and intuitive extensions of this algorithm that seems very robust for all datasets and all sample periods.

Table 4.7.2: CAGR on Selected Trading Periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.33	0.13	0.29	0.20	1.34	0.17	0.49	0.18	0.73	0.15	4.00	0.16
1Yr	0.13	0.21	-0.22	0.17	0.07	0.11	0.43	0.39	1.82	0.29	0.56	-0.01
2Yrs	-0.05	0.22	-0.20	0.22	-0.06	0.10	0.04	0.26	0.23	-0.04	0.69	0.16
3Yrs	-0.11	0.12	-0.06	0.17	-0.15	0.05	-0.03	0.18	0.05	-0.07	1.45	0.23
4Yrs	-0.05	0.16	0.22	0.19	-0.06	0.09	0.15	0.21	0.22	0.03	1.44	0.19
5Yrs	0.05	0.23	0.45	0.23	0.24	0.17	0.47	0.30	0.27	0.05	1.65	0.20
7Yrs	0.03	0.12	0.26	0.16	0.17	0.09	0.27	0.17	0.34	0.08	2.11	0.20
10Yrs	0.23	0.15	0.39	0.22	0.27	0.14	0.32	0.17	0.69	0.07	3.67	0.19
15Yrs	0.22	0.13	0.45	0.21	0.52	0.15	0.42	0.18	0.78	0.07	4.84	0.13

In addition to the final cumulative wealth and the CAGR, we also examine how the cumulative wealth changes over different trading sub-periods. Table 4.7.2 shows the trends of the cumulative wealth by the proposed PAMR algorithm together with the performance of respective indices over the sub-periods considered. In the case of the NYSE(O) and NYSE(N), we can see that the proposed PAMR strategy consistently surpasses its benchmarks returns over the entire period. However, the performance of the PAMR-based algorithm has not kept pace with that generated by simple Buy-and-Hold on recent datasets. In fact the PAMR has been a losing strategy of late on all datasets.

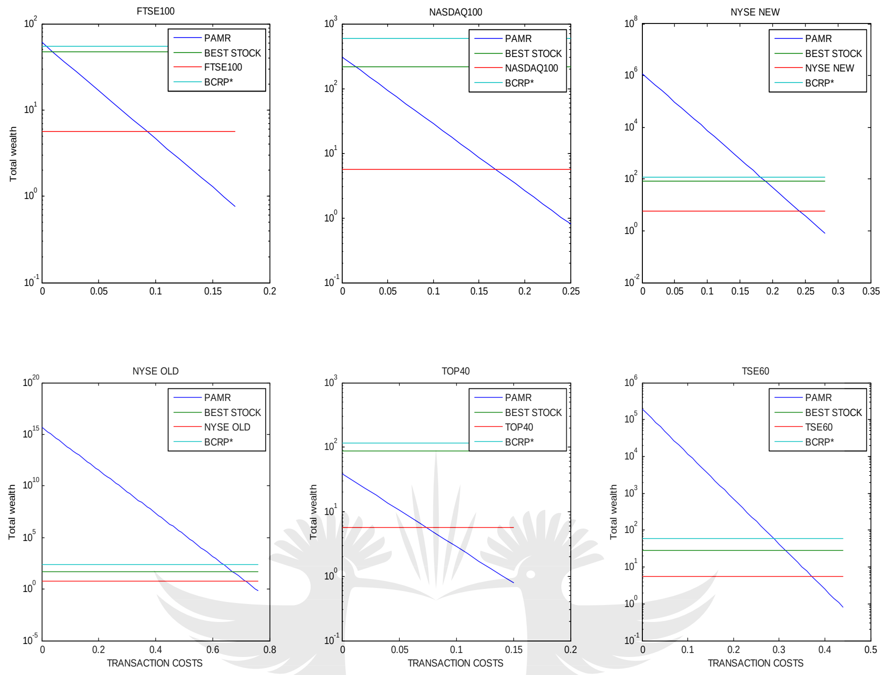
4.7.4.2 Performance Risk Analysis Table 4.7.3 shows a table of risk statistics generated by the PAMR algorithm for the six datasets. The PAMR-based strategy seems to have a volatility of returns that is significantly higher than the one achieved by its respective Buy-and-Hold benchmarks. Although the long-term Sharpe Ratios seems to indicate that the PAMR generates better risk-adjusted returns than their respective benchmarks, we believe this can be a misleading conclusion. In our view, the PAMR-based strategy generates much worst risk adjusted returns especially using our new datasets.

Table 4.7.3: Risk Statistics of the PAMR-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.44	0.18	0.34	0.16	0.55	0.17	0.51	0.23	0.45	0.19	0.50	0.13
α	0.03	0.00	0.03	0.00	0.30	0.00	0.06	0.00	0.15	0.00	0.57	0.00
β	1.51	1.00	1.27	1.00	1.53	1.00	1.39	1.00	1.27	1.00	1.45	1.00
γ	2.73	0.00	0.05	0.00	-1.00	0.00	2.35	0.00	1.89	0.00	4.69	0.00
$\beta\uparrow$	1.58	1.00	1.27	1.00	1.46	1.00	1.52	1.00	1.34	1.00	1.63	1.00
$\beta\downarrow$	1.44	1.00	1.21	1.00	1.47	1.00	1.28	1.00	1.08	1.00	1.37	1.00
SR	0.71	0.37	0.72	0.78	1.70	0.59	0.90	0.53	1.28	0.44	3.31	0.64
MDD	-0.75	-0.40	-0.67	-0.30	-0.60	-0.41	-0.70	-0.47	-0.75	-0.64	-0.31	-0.37
PP	0.55	0.56	0.55	0.57	0.57	0.58	0.55	0.56	0.54	0.55	0.57	0.53
PP$\uparrow$	0.76	1.00	0.76	1.00	0.74	1.00	0.78	1.00	0.74	1.00	0.73	1.00
PP$\downarrow$	0.27	0.00	0.28	0.00	0.33	0.00	0.26	0.00	0.30	0.00	0.38	0.00

4.7.4.3 PAMR Brokerage Costs Analysis Figure 4.7.2 shows the performance of the PAMR-based PS selection algorithm net of brokerage commissions against some benchmark strategies on our six datasets. This performance is very similar to the one presented in the previous section. We can therefore draw very similar conclusions as in the case of the CWMR algorithm..

Figure 4.7.2: Brokerage Cost Analysis for the PAMR Algorithm



4.8 SUMMARY

This chapter has conducted an empirical survey on the most promising online PS problem. After presenting the mathematical foundation of each individual algorithms, we have analyzed each of the algorithms empirically using both existing (old) datasets and some recent data that have never been tested previously . The simulation results have demonstrated that all the surveyed PS algorithms (exept the Anticor) achieve less impressive cumulative wealth growth on all new and previously untested datasets compared to what the results were on old reported datasets. Table 4.8 summarizes the cummulative performance of all the algorithms surveyed and for all the data sets.

Table 4.8: Performance comparison

	FTSE100	NASDAQ	NYSE(O)	NYSE(N)	TOP40	TSE60
Market	5.00E+00	1.07E+01	1.82E+01	1.42E+01	1.37E+01	6.12E+00
Best Stock	4.76E+01	2.15E+02	8.48E+01	5.33E+01	8.78E+01	2.78E+01
UBCRP	5.63E+00	1.05E+01	3.18E+01	2.67E+01	1.36E+01	9.18E+00
BCRP*	5.48E+01	5.92E+02	1.21E+02	2.49E+02	1.15E+02	5.83E+01
Anticor	6.82E+01	2.67E+02	4.17E+07	5.59E+05	4.54E+02	1.51E+03
CWMR	6.02E+01	6.87E+02	6.63E+15	1.73E+06	3.19E+01	1.64E+05
FNN	1.44E+01	1.33E+02	3.70E+12	2.01E+05	1.00E+02	4.81E+03
Kernel	3.59E+01	5.55E+01	8.68E+07	7.34E+02	1.00E+02	2.44E+02
NN	1.59E+01	1.62E+02	4.32E+10	6.38E+04	8.71E+01	4.24E+02
PAMR	6.05E+01	3.07E+02	4.62E+15	1.20E+06	3.92E+01	1.95E+05

For comparison purposes, Table 4.8 also includes the performance of the various market indices as well as the performance of the best stock for the respective markets. The table also includes the performance of the uniform best constantly rebalanced portfolio for each country surveyed as well as the best constantly rebalanced portfolio in hindshight. Although the PAMR and the CWMR strategies generated excellent long term wealth growth overall, this performance is yet to be experienced on more recent data. It is well possible that stock markets are more efficient than history suggests and therefore the need of finding more robust trading strategies is more compelling than never. The remainder of this thesis is dedicated to finding such strategies that can perform exceeding well on all or most data surveyed.

5.0 KALMAN FILTERING AND ONLINE LEARNING ALGORITHMS FOR PORTFOLIO SELECTION PROBLEMS

5.1 INTRODUCTION

The conventional investment wisdom over the last several decades has been to Buy-and-Hold good-quality stocks for the long run with the hope that these securities will rise over time. It has been generally believed that security markets are extremely efficient in reflecting information not only about individual stocks but also about the stock market as a whole. The accepted view was that because information arises stochastically and is incorporated into the prices of securities without any material delay, neither technical analysis nor fundamental analysis would enable an investor to achieve returns greater than those that could be obtained by holding a randomly selected portfolio of individual stocks, at least not without taking additional risk. However, the amount of evidence showing the disadvantage that traditional, long-term, Buy-and-Hold investors currently face is staggering. After recent bear markets and the resulting poor performance delivered by most fund managers, there is renewed search for reliable active investment strategies that can outperform not only the market but also the best stock.

In recent years the growth of theoretically well grounded algorithms for Online PS problems has been significant. These algorithms have demonstrated good finite sample properties with a performance that generally exceeds both the market and the best stock even after accounting for transaction costs. As argued in Chapter 4, algorithms such as the nonparametric Kernel-based PS algorithm (Gyorfi et al. (2006)), the Passive Aggressive-base sequential PS (Li et al. (2012)) and the Anticor algorithm (Borodin et al. (2004)) have demonstrated an impressive wealth growth in historical simulations. However, the empirical simulatons presented in Chapter 4 also suggest that the excellent performance generated by most of these algorithms can not be generalised for all datasets and all time frames.

This chapter presents a new heuristic approach to Online PS that significantly improves the performance of all the algorithms presented in Chapter 4 and for all datasets and time periods. We build on the existing state-of-the-art algorithms for PS but use ideas from signal processing and statistical learning to demonstrate the superiority of our new methodology. Unlike most existing Online learning algorithms that use the raw price relative as the input in the programme trading algorithm, we propose an alternative measure of price relative that is more consistent with portfolio manager's best practice. We use a state-space model via the Kalman Filter algorithm to filter price-cycle oscillations out of the current share prices and compute the Trend Adjusted Price Relative (TAPR). The TAPR helps to de-noise the stock price data in order to account for the possibility of multi-period mean reversion in stock prices as argued by Li et al. (2012). To our knowledge, this is the first time that research has combined ideas from signal processing with Online learning algorithms to select portfolios in an optimal way.

To demonstrate the usefulness of our methodology we evaluate our algorithm against some benchmark Online portfolio allocation techniques using previously untested market datasets. Our algorithm substantially outperforms existing Online stock selection techniques without much additional computational demand or modelling complexity.

5.2 MATHEMATICAL MODEL

The stock market model considered in this chapter is the same as the one presented in Chapter 2. We consider a market of m assets such that a market vector $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^m) \in R_+^m$ is the vector of m numbers representing price relatives for a given trading period. The j^{th} component $\mathbf{x}^j$ of $\mathbf{x}$ expresses the ratio of two consecutive closing prices of asset j, such that $x_{t,i} = \frac{p_{t,i}}{p_{t-1,i}} \geq 0$ and each element $p_{t,i}$ represents the closing price of asset i on period t and $p_{t-1,i}$ is the closing price of stock i on period $t-1$. Thus, an investment in asset i on period t increases by a factor of $\mathrm{x}_{t,i}$. The investor distributes his capital at the beginning of each trading period according to a portfolio vector $\mathbf{b} = (\mathbf{b}^1, \mathbf{b}^2, ..., \mathbf{b}^m)$, where the j^{th} component $\mathbf{b}^j$ of $\mathbf{b}$ denotes the proportion of the investor's capital invested in asset j. Throughout this thesis we assume that the portfolio vector $\mathbf{b}$ has non negative components, which means that the investment strategy is self- financing and that both consumption of capital and short selling is not permitted. Mathematically this simply means that $\sum_{j=1}^{m} b^j = 1$ and $b^j \geq 0$.

Starting with an initial wealth S_0, we showed in the previous chapter that after n trading periods, the investment strategy achieves the wealth

$$S_n = S_0 \prod_{i=1}^{n} \left\langle \mathbf{b}_i \left(\mathbf{x}_1^{i-1}\right), \mathbf{x}_i \right\rangle \ = S_0 \exp \left\{ \sum_{i=1}^{n} \log \left\langle \mathbf{b}_i \left(\mathbf{x}_1^{i-1}\right), \mathbf{x}_i \right\rangle \right\} \tag{5.1}$$

$$S_n = S_0 \exp \left\{ n W_n (B) \right\} \tag{5.2}$$

where $W_n(B)$ denotes the average growth rate and is given by

$$W_n (B) = \frac{1}{n} \sum_{i=1}^{n} \log \left\langle \mathbf{b}_i \left(\mathbf{x}_1^{i-1}\right), \mathbf{x}_i \right\rangle . \tag{5.3}$$

This is essentially a log utility function whose expected value needs to be maximised given a suitable choice of non-negative portfolio vectors $\mathbf{b}_i\left(\mathbf{x}_1^{i-1}\right)$. Therefore maximising $S_n = S_n(B)$ is equivalent to maximising the average growth rate whose expression is given by

$$W_n (B) : \mathbf{b}_i^* \left(\mathbf{x}_1^{i-1}\right) = \arg_b \max E \left\{ \log \left\langle \mathbf{b}_i \left(\mathbf{x}_1^{i-1}\right), \mathbf{x}_i \right\rangle \mid \mathbf{x}_1^{i-1} \right\} \tag{5.4}$$

This is a nonlinear convex optimisation problem for which closed-form solutions are not easily available. The search for an acceptable solution has lead many researchers in the machine-learning community to suggest various deterministic or randomised rules that explicitly determine a sequence of portfolio weights with the aim of maximising the investor's wealth without prior knowledge of the statistical distribution of stock prices. The current research study proposes many such algorithms.

5.3 TREND ADJUSTED PRICE RELATIVES

The generally accepted practice in published state-of-the-art algorithms (see Borodin et al. (2004); Li et al.(2012))) is to use the raw price relative as the main argument in the machine-learning algorithm. Raw price relative for security i is defined as the ratio of two consecutive closing prices $x_{t,i} = \frac{p_{t,i}}{p_{t-1,i}}$ at time t . However, raw price relatives and their logarithms are notoriously volatile time series and there is very little to believe that the mean reversion characteristics will be effective in the very next periods, if at all. While the academic research community has adopted the raw price relative as the primary input argument in machine-learning systems, market practitioners have always applied some smoothing to stock prices before proceeding to any statistical analysis.

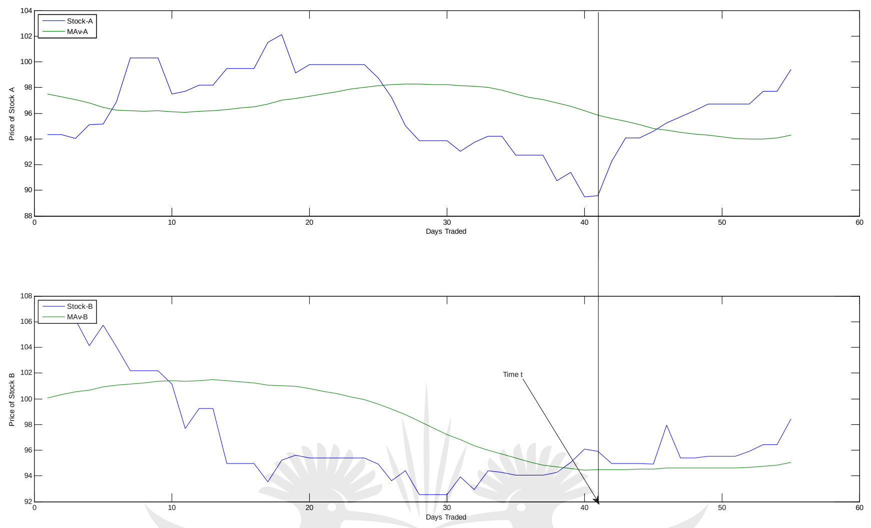
Many stochastic processes, including stock prices, have inherent noise that obscures the true underlying values. To be successful in today's market place, portfolio managers need to see through all the market noise that occurs on a daily basis and be able to identify the trend in stock prices. Classical approaches to this problem have been to apply a moving average, as in Li et al. (2012), or an exponential moving average to the time-series to obtain a smoothed values. We believe that it is potentially risky to use the simple moving averages on time-series because they most likely will change the statistical properties of the time-series under consideration during period of highten volatility. Appropriate smoothing of price data can eliminate some of the market noise and allow the portfolio manager to focus on trading more persistent patterns. In order to reduce the impact of these noises in smoothing stock prices we use the scalar Kalman Filter (KF).

The KF is an optimal filter that provides us with clearer stock price resolution and allow us to isolate the peak excursions when the stock price significantly departs from its "true" unobserved component. When the stock price significantly departs from its trend price, we anticipate that the move is unsustainable and take the view that short-term price will reverse from these peaks. Therefore, critical to our analysis is not how much the price moves from one period to the next as assumed by comparable studies, but instead, how far apart the stock price is from its Kalman trend.

To illustrate why TAPR and raw price relative can lead to two very distinct conclusions, we plot in Figure 5.1 two actively traded stock prices in the Johannesburg Stock Exchange together with their respective trend derived from a simple moving average filter. There are clearly periods where both stocks lie above, below or in opposite sides of their respective moving averages. On the surface, it seems as though the higher the stock price is from its moving average, the more bullish the market is (and the lower it goes, the more bearish). In practice, however, the reverse is generally true. Extremely high readings are a warning that the market may soon reverse to the downside as this tend to reveal that traders are far too optimistic. When this occurs, fresh new buyers are often few and far between. Meanwhile, very low readings signify the reverse; the bears are in the ascendancy and a bottom is near.

To illustrate this idea further we consider the following simple example: At time t, for example, stock A rises by about 3% while stock B falls by 1%. According to the standard mean reversion paradigm (Borodin et al. (2004); Li et al.(2012) are good examples), most Online learning algorithms will likely start transferring wealth away from stock A and into stock B, given the outperformance of stock A over B most. However, we believe this is a premature and potentially misleading interpretation of the dynamics of these stocks, as stock A comes from a very over sold position (A is below its moving average plotted in red in Figure 5.1) and is simply "catching up". Our proposed algorithm acknowledges this fact and will instead recommend transferring more wealth away from stock B and into stock A. As a consequence the distance between a stock price and its moving average (or trend) is the critical input into our proposed algorithm.

Figure 5.1: Impact of smoothing on two stock prices



In this chapter we use the Kalman Filter as the appropriate filtering methodology, given its well-known robustness to noisy data. Using the Kalman Filter helps us to filter out very volatile and cyclical components of stock prices and derive what we refer to as the TAPR. If p_t represents the stock price at time t and p_t^k the filtered or unobserved "true" price (as defined below) at time t, the TAPR is simply expressed as

$$TAPR = \frac{p_t}{p_t^k}$$

The TAPR ratio is used in this thesis to judge how over sold or over bought the stock of a corporation is relative to its own unobserved true price. The farther away the price relative is from its own trend, the more attractive the stock will be for purchase or for sale in our model. Because our algorithm takes a bet only when a given stock price significantly deviates from its trend, this new measure of price relative is therefore expected to yield better mean reversion characteristics compared to traditional counterparts.

5.3.1 Statistical Properties of the TAPR

Let us assume that the stock price process is governed by a state-space representation where the measurement equation is given by:

$$p_t = Mp_t^k + v_t \tag{5.5}$$

where M is known and $v_t \sim \mathbb{N}(0, P)$ with P known. For simplicity we assume that the prices process p_t^k is uncorrelated with the innovation v_t. Equation 5.5 essentially describes the relationship between the observed stock price p_t and the unobserved "true" stock price p_t^k.

Now, Let us assume that there is a transition equation that follows an autoregressive AR(1) process and given by

$$p_t^k = p_0 + \phi p_{t-1}^k + w_t \tag{5.6}$$

where $|\phi| < 1$ is assumed known and $w_t \sim \mathbb{N}(0, Q)$ with Q known.

From Equation 5.5 we obtain the following ratio:

$$\frac{p_t}{p_t^k} = M + \frac{v_t}{p_t^k} \tag{5.7}$$

Equation 5.7 represents our measure of the TAPR centred around the known constant coefficient M. TAPR is dominated by the behaviour of the ratio $\frac{v_t}{p_t^k}$ whose statistical properties could be easily defined .

From the specification of the transition equation it is easy to demonstrate that the mean $E\left(p_t^k\right) = \frac{p_0}{1-\phi}$ and the variance $Var\left(p_t^k\right) = \frac{Q}{1-\phi^2}$ therefore $p_t^k \sim \mathbb{N}\left(\frac{p_0}{1-\phi}, \frac{Q}{1-\phi^2}\right)$. In Appendix A we prove (using the Taylor expansions around $g(.)$) that given two random variables v_t and p_t^k where p_t^k has support $0, \infty)$ the function $G = g\left(v_t, p_t^k\right) = \frac{v_t}{p_t^k}$, has the following approximates for $E(G)$ and $Var(G)$.

$$E\left(\frac{v_t}{p_t^k}\right) = \frac{Ev_t}{Ep_t^k} - \frac{Cov\left(v_t, p_t^k\right)}{E^2 p_t^k} + \frac{Var\left(p_t^k\right) Ev_t}{E^3 p_t^k} \tag{5.8}$$

$$Var\left(\frac{v_t}{p_t^k}\right) \approx \frac{E^2 v_t}{E^2 p_t^k}\left[\frac{Var(v_t)}{E^2 v_t} - 2\frac{Cov\left(v_t, p_t^k\right)}{Ev_t \; Ep_t^k} + \frac{Var\left(p_t^k\right)}{E^2 p_t^k}\right] \tag{5.9}$$

We therefore provide the following expression for the mean and variance of the ratio $\frac{p_t}{p_t^k}$

$$E\left(\frac{p_t}{p_t^k}\right) = M \tag{5.10}$$

With some abuse of notation we can calculate $Var\left(\frac{p_t}{p_t^k}\right)$ as:

$$Var\left(\frac{p_t}{p_t^k}\right) = aVar(v_t) + bVar(p_t^k), \text{ where } a = \frac{E^2 v_t}{E^2 p_t^k E^2 v_t}, b = \frac{E^2 v_t}{E^2 p_t^k E^2 p_t^k} \tag{5.11}$$

The ratio of the observed stock price to its unobserved component (Kalman trend) is such that

$$\frac{p_t}{p_t^k} \sim \mathbb{N}\left(M, aR + b\frac{Q}{1-\phi^2}\right) \tag{5.12}$$

This last expression demonstrates that the statistical properties of the TAPR $\frac{p_t}{p_t^k}$ are well established and that these in a sense assure better mean reversion characteristics. This finding is the main motivation why we would expect historical simulations that use the new price relative measure to outperform the base line models that rely on raw price relative as used in most empirical analyses. In fact, the last equation shows that 95% of time we will have the relationship below

$$M - 2\left(aR + b\frac{Q}{1-\phi^2}\right) \leq TAPR \leq M + 2\left(aR + b\frac{Q}{1-\phi^2}\right) \tag{5.13}$$

and whenever our TAPR is outside those bounds we can expect a reversal in the very next period.

5.3.2 The Scalar Kalman Filter Algorithm

To fully specify our measure of TAPR, we now explain the basic steps needed to derive the Kalman Filter-based price trend, p_t^k.

The Kalman Filter is a recursive algorithm that produces estimates of a time series of unobservable variables (along with parameter estimates for the theoretical model that generates the data) using a related but observable time series of variables. The estimates of the unobservable variables are updated at each time step based on the revelation of new observable data. The Kalman Filter uses the current observation to predict the next period's value of unobservable stock price and then uses the realisation of the next period to update that forecast. The linear Kalman Filter is optimal; i.e. it has the minimum Mean Squared Error estimator if the observed variable and the noise are jointly Gaussian.

Let us assume $p_1, p_2, ..., p_t$ are the observed values of the stock prices for a given firm at time $1, 2, ..., t$. We assume that p_t depends on an unobservable quantity p_t^k, known as the state of nature or the true stock price. The Kalman Filter recursive estimation algorithm works as follows:

At time t_0 the process starts with an initial estimate p_0^{k*} for p_t^k which has a mean of μ_0 and a variance:

$$P = E\left(p_0^k - p_0^{k*}\right)^2 \tag{5.14}$$

At time t_1 and before any measurement is taken (or before any stock price is revealed) the state price is given by:

$$\overline{p}_1^k = \phi p_0^{k*} \tag{5.15}$$

its variance is given by:

$$\overline{P} = E\left(p_1^k - \overline{p}_1^k\right)^2 = E\left[\phi p_0^k + \mu_0 - \phi p_0^{k*}\right]^2 = \phi^2 P + Q, \tag{5.16}$$

and the transition equation is given by:

$$\overline{p}_1 = M\overline{p}_1^k \tag{5.17}$$

Still at time t_1 and after the measurement p_1 becomes available:

$$p_1^{k*} = \overline{p}_1^k + K\left[p_1 - \overline{p}_1\right] = \overline{p}_1^k + K\left[p_1 - M\overline{p}_1^k\right] \tag{5.18}$$

where K is the kalman gain. After p_1 becomes available, the variance of the measurement needs to be updated in the following way:

$$P = \left[p_1^k - p_1^{k*}\right]^2 = \left[p_1 - \overline{p}_1^k - K\left[p_1 - M\overline{p}_1^k\right]\right]^2 \tag{5.19}$$

$$P = \left[p_1^k - \overline{p}_1^k - K\left[Mp_1^k + w_1 - M\overline{p}_1^k\right]\right]^2 = \overline{P}\left(1 - KM\right)^2 + RK^2 \tag{5.20}$$

The value of the Kalman gain (K) that minimises the variance is:

$$K = M\overline{P}\left(\overline{P}M^2 + R\right)^{-1} \tag{5.21}$$

Table 5.1 shows the 5 equations needed for a recursive estimation of the multivariate Kalman Filter.

Table 5.1	KF($\mu_{t-1}, \sum_{t-1}, \mu_t, z_t$)
Inputs	A, B, R all constant parameters
Output	$\mu_t, \sum_t$
1	$\overline{\mu} = A\mu_{t-1} + B\mu_{t-1}$
2	$\overline{\sum} = A\sum_{t-1} A^\intercal + R$
3	$K_t = \overline{\sum}_t C_t^\intercal \left(C_t \overline{\sum}_t C_t^\intercal + Q_t \right)^{-1}$
4	$\mu_t = \overline{\mu}_t + K_t \left(z_t - C_t \overline{\mu}_t \right)$
5	$\sum_t = \left(I - K_t C_t \right) \overline{\sum}_t$

Table 5.2 shows a pseudo code implementation to the Kalman-induced TAPR algorithm

```
Table 5.2: TAPR (Price, $\phi$, $M$, $P$)
$|\phi| < 1$: autoregressive coefficient of the state equation
$M$: coefficient in the measurement equation
Initialise the Kalman Filter parameters $(Q, R, Z)$
Output: $\mathbf{x}_t$
for $t = 1, 2, ...n$ do
    1  $\overline{Z} = \phi \widehat{Z}$
    2  $\overline{P} = \phi^2 P + Q$
    3  $K = M\overline{P}\left[\overline{P}M^2 + R\right]^{-1}$: Kalman gain
    4  $\widehat{Z} = \overline{Z} + K\left[P_t - M\overline{Z}\right]$
    5  $P = \overline{P}\left[1 - KM\right]^2 + RK^2$
    6  $\mathbf{x}_t = \frac{\textbf{Price}}{\widehat{Z}_t}$: Trend Adjusted Price Relative
end
```

5.3.3 TAPR Parameter Settings

This section briefly explains the TAPR parameter settings that will be used throughout this thesis. There are four key parameters in the proposed Kalman Filter-induced TAPR algorithms. The first parameter is the variance, R, of the innovation in the state equation (Equation 5.5). We set $R = 100$ throughout this thesis. The second parameter is the coefficient M in the state equation, which will be set arbitrarily to $M = 0.5$. The third paramter represents the variance of the white noise parameter, Q, in the transition equation. Throughout this thesis we will assume tht $Q = 1000$. The last parameter is the autoregressive coefficient, ϕ, in the transition equation (Equation 5.6). For simplicity purpose we set this parameter as: $\phi = 0.95$. Despite these simple settings, we acknowledge that the best parameters are often datasets dependent. In all our experiments, we decided to simply set these parameters empirically without trying to optimise their values. An alternative and more efficient way to set these parameters is to estimate their values using well known routines like Gibbs Sampling. It is therefore likely that the parameters choice made in this section might be suboptimal and lead to lesser wealth growth than what could be achieve via a more robust parameter estimation.

5.4 ONLINE PORTFOLIO SELECTION ALGORITHMS VIA TREND ADJUSTED PRICE RELATIVES

This section analyses the empirical validity of our proposed mean reversion principle via the TAPR. To do so, we revisit some of the Online PS algorithms surveyed in Chapter 4 and propose a set of new and powerful algorithms that substantially outperform those presented in Chapter 4. We show that our novel approach is in fact general enough and can improve the performance of most existing state-of-the-art PS algorithms without any additional modelling complexity.

5.4.1 Momentum Anticor (MAC) Algorithm via TAPR

In this section, we revisit the Anticor (AC) algorithm of Borodin et al. (2004) and proposes very important and significant extensions. It will be shown that the new proposed algorithms can exploit both mean reversion and momentum features of stock prices.

In its original form, the AC algorithm provides results on historical stock prices that show that an algorithm derived from simple heuristics can significantly outperform those that provide theoretical guarantees. Borodin et al. (2004) show historical simulation results or some real-market datasets that demonstrate that the AC algorithm is indeed very robust in those datasets based solely on the mean reversion principal.

Despite this impressive empirical performance, an extensive body of behavioral finance literature has documented that price reversal is hardly the only feature at play in equity markets. It has been argued that price momentum and reversals tend to coexist in world stock markets in the short term. In a comprehensive investigation, Conrad and Kaul (1998) find both momentum and contrarian profits in the U.S. market, depending on the time horison investigated. Balvers and Wu (2006) also demonstrate that mean reversion and momentum can simultaneously occur on the same set of assets in 18 developed countries. This coexistence of both momentum and reversal means that an exclusive focus on mean reversion is likely to generate suboptimal portfolio allocations. It is therefore possible that in the presence of both price reversal and price continuation, the original AC algorithm will fail to perform optimally. To correct this shortcoming we provide an important modification to the AC algorithm that can deal with both momentum as well as reversal in the following way.

As in Borodin et al. (2004), our proposed algorithm evaluates changes in stock performance by dividing the historical sequence of past returns series into equal-sized periods called windows, each with a length of w days where w is an adjustable parameter. Our algorithm also transfers wealth from one group of experts to another group of experts depending on the most recent performance and the strength and direction of correlation over subsequent windows.

For a window length w, we consider $\mathbf{LX}_1$ and $\mathbf{LX}_2$ as two $w \times n$ matrices over two consecutive time windows, which we compute as follows:

$$\mathbf{LX}_1 = \left(\log\left(X_{t-2w+1}\right), ..., \log\left(X_{t-w}\right)\right)^T \text{ and } \mathbf{LX}_2 = \left(\log\left(X_{t-w+1}\right), ..., \log\left(X_t\right)\right)^T \tag{5.22}$$

The j^{th} column of $\mathbf{LX}_k$ is denoted by $LX_k(j)$ and simply tracks the performance of stock j in window k where $k = 1, 2$. Let $\mu_k(j)$ be the mean of $LX_k(j)$ and $\sigma_k(j)$ be the corresponding standard deviation. The cross-covariance matrix between the columns vectors of LX_k is defined as follows:

$$Mcov(i,j) = \frac{1}{w-1}\left[LX_1(i) - \mu_1(i)\right]^T \left[LX_2(j) - \mu_2(j)\right] \tag{5.23}$$

and the corresponding cross-correlation matrix is given by:

$$MCorr(i,j) = \begin{cases} \frac{MCov(i,j)}{\sigma_1(i)\sigma_2(j)}, & \sigma_1(i), \sigma_2(j) \neq 0 \\ 0 & otherwise \end{cases} \tag{5.24}$$

The reversion to mean strategy of Borodin et al. (2004) states that if

$$\mu_2(i) \geq \mu_2(j) \text{ and } Mcorr(i,j) > 0 \tag{5.25}$$

the proportion of wealth to be moved from stock i to stock j is defined as:

$$claim_{i \to j} = Mcorr(i,j) + \max\left(-MCorr(i,i), 0\right) + \max\left(-MCorr(j,j), 0\right) \tag{5.26}$$

To account for the possibility of short term reversal as well as momentum we expand the benchmark AC algorithm as follows. If

$$\mu_2(i) \geq \mu_2(j) \text{ and } Mcorr(i,j) \leq 0$$

the proportion of wealth to be moved from stock i to stock j is defined as

$$claim_{i \to j} = -Mcorr(i,j) + \max\left(MCorr(i,i), 0\right) + \max\left(MCorr(j,j), 0\right) \tag{5.27}$$

Therefore whenever our algorithm detects that stock i outperformed stock j during the last window but $i's$ performance in the last window is not anti-correlated to $j's$ performance in the second-to-last window it transfers wealth from stock j to stock i (the model is adding into the holding of stock i) and calculate new portfolio weights. The simple logic here is that there will be price continuation in the direction of the outperforming stock.

From both the reversal to the mean and the price continuation conditions we calculate the transfers of stock i to stock j as:

$$transfer_{i\to j} = b_{t-1}\left(i\right)\frac{claim_{i\to j}}{\sum_j claim_{i\to j}} \tag{5.28}$$

Using these transfer values, the portfolio is defined to be:

$$b_t\left(i\right) = b_{t-1}\left(i\right) + \sum_{i\neq j}\left(transfer_{j\to i} - transfer_{i\to j}\right) \tag{5.29}$$

and we call the resulting algorithm the Momentum-Anticor or MAC. In Table 5.3, we show a pseudo code implementation of the proposed KMAC algorithm where the TAPR is derived from the scalar Kalman Filter algorithm.

Table 5.3: KMAC $(\mathrm{Pr}\,ice, w, \phi, M, P)$

```
$w$: window sise
$\phi$: autoregressive coefficient of the state equation
$M$: coefficient in the measurement equation
$\mathbf{b}_0$: initial portfolio weights $\mathbf{b}_0 = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$
Initialise the Kalman Filter parameters $(Q, R, Z, V)$
for $t = 1, 2, ...$
1    Estimate the true price $\mathbf{Z}_t$ using the following procedure
2    $\mathbf{x}_t$=TAPR $(\mathrm{Pr}\,\mathbf{ice}, \phi, M, P)$
3    Return the current portfolio $\mathbf{b}_t$ if $t < 2w$
4    compute $\mathbf{LX}_1 = (\log(X_{t-2w+1}), ..., \log(X_{t-w}))^T$
5    compute $\mathbf{LX}_2 = (\log(X_{t-w+1}), ..., \log(X_t))^T$
6    compute $\boldsymbol{\mu}_1 = average(LX_1)$ and $\boldsymbol{\mu}_2 = average(LX_2)$
7    compute $Mcov(i,j) = \frac{1}{w-1}[LX_1(i) - \mu_1(i)]^T [LX_2(j) - \mu_2(j)]$
8    compute $MCorr(i,j) = \begin{cases} \frac{MCov(i,j)}{\sigma_1(i)\sigma_2(j)}, & \sigma_1(i), \sigma_2(j) \neq 0 \\ 0 & otherwise \end{cases}$
9    Initialise $claim_{i \to j} = 0$
10   if $\mu_2(i) \geq \mu_2(j)$ and $Mcorr(i,j) > 0$
11   $claim_{i \to j} = Mcorr(i,j) + \max(-MCorr(i,i), 0) + \max(-MCorr(j,j), 0)$
12   else if $\mu_2(i) \geq \mu_2(j)$ and $Mcorr(i,j) \leq 0$
13   $claim_{i \to j} = -Mcorr(i,j) + \max(MCorr(i,i), 0) + \max(MCorr(j,j), 0)$
14   $transfer_{i \to j} = b_{t-1}(i) \frac{claim_{i \to j}}{\sum_j claim_{i \to j}}$
15   $b_t(i) = b_{t-1}(i) + \sum_{i \neq j}(transfer_{j \to i} - transfer_{i \to j})$
end
```

The pseudo code presented in Table 5.3 is clearly a generalisation of the AC algorithm. Because it accounts for both price reversals and price momentum, we expect our KMAC to perform significantly better than the original AC algorithm. Further to that, Table 5.3 also allows us to derive a new class of algorithms that could form the basis for valid investment strategies. For example, if the portfolio manager uses only the original AC Algorithm together with the TAPR we refer to this algorithm as the Kalman Anticor (KAC) algorithm. Of course the derived KAC algorithm will not exploit the joint coexistance of both mean reversals and momentum but will still represent an improvement from the benchmark AC algorithm. Similarly, if a portfolio manager uses the raw price relative as the main argument, together with our proposed generalised AC Algorithm, we refer to this version as the Momentum Anticor or MAC. The empirical simulations below will show the results of this new class of algorithms, including the MAC, the KAC and the KMAC together with the originally proposed AC algorithm.

5.4.2 Combining Portfolio of Experts

Our proposed Anticor-based Online PS algorithms require an important fine tuning parameter, namely the window length w. Because it is impossible to know ex ante what window sise will generate better out of sample performance our approach is simply to select different window periods, called experts, and allow them to compete. As in Gyorfi et al. (2004) we form a mixture of all experts using a positive probability distribution q_w on the set of all window lengths w of positive integers. The investment strategy simply weights these experts H^w according to their past performances and the q^w such that after the t^{th} trading period the investor wealth becomes:

$$S_t = \sum_{w} q_w S_t \left(H^w\right) \tag{5.30}$$

where $S_t\left(H^w\right)$ is the capital accumulated after t^{th} trading period using the expert H^w with initial capital $S_0 = 1$. We then form our final portfolio by weighting all expert portfolio using the following:

$$b\left(X_1^{t-1}\right) = \frac{\sum_w q_w S_{t-1}\left(H^w\right) h^w \left(x_1^{t-1}\right)}{\sum_w q_w S_{t-1}\left(H^w\right)} \tag{5.31}$$

5.5 EMPIRICAL RESULTS

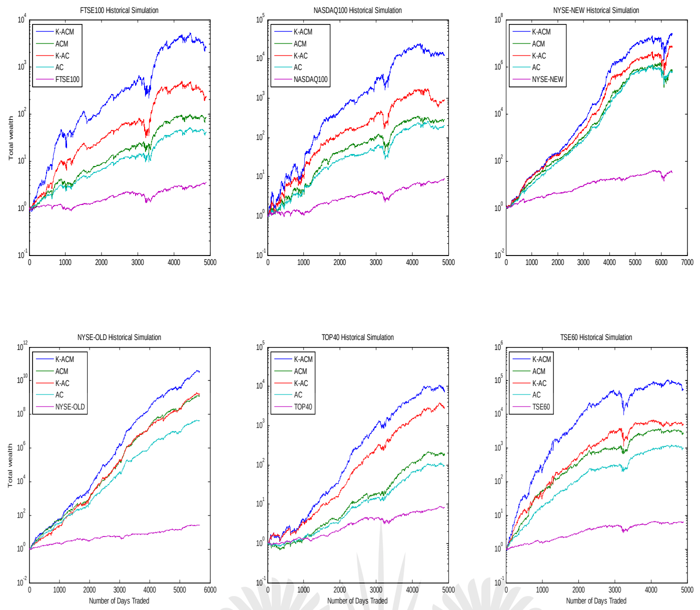
This section presents numerical results obtained by applying all the Anticor-based algorithms, including the MAC, the KAC and the KMAC, using six financial market datasets described in Chapter 3. As usual, the back-testing experiments in this section will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed on a daily basis. For our simulatoin purpose we set the maximum window sise $W = 30$.

5.5.1 Analysis of Cumulative Wealth

The first experiment evaluates the compounded wealth achieved by the AC-based learning-to trade algorithms including a $c = 10$ basis points transaction cost over the entire sample period. Figure 5.2 summarises the total wealth achieved by the new class of algorithms on the six market datasets.

On the NYSE (N) datasets for example, after trading for 22 years, the total wealth achieved by the AC strategy and the MAC strategy impressively increases from \$1 to almost \$560K and \$640K, respectively, which are much higher than the best stock that achieves \$84.82 and the market index that achieves a mere \$31.82. For the same datasets the performance of the KAC and KMAC are even more impressive. In the case of the KAC, for example, 1\$ rises to \$6.5-million after 6431 days of trade while the KMAC rises to an impressive \$23.0-million during the same time frame. Our proposed methodology therefore achieves a growth in wealth that is more than 41 times that achieved by its benchmark algorithm. This dominance of our proposed algorithms relative to their benchmarks is also evident in all other datasets and time frames as shown in Table 5.4. Both KAC and KMAC achieve considerably better results than the market index, the best stock in the market, as well as all their benchmark AC algorithm for all data sets.

Figure 5.2: Performance of Selected AC-based Algorithms



Besides the preceding results, we are also interested in examining how the total wealth achieved by various strategies changes over different trading periods. Table 5.4 shows the total wealth achieved by the KMAC (simply called ALG in Table 5.4). From Table 5.4, we first observe that our proposed KMAC algorithm consistently outperforms not only the benchmark index but also the AC over most trading sub-samples and all datasets. It is also interesting to note that, although the market drops sharply due to the financial downturn in 2008, the proposed KMAC algorithms are still able to achieve excellent results in all datasets, which is especially more impressive in the later part of the South Africa Top40 datasets. All these impressive results demonstrate the usefulness and robustness of the proposed learning-to-trade algorithm.

Table 5.4: CAGR of the KMAC Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.44	0.13	0.57	0.20	0.79	0.17	0.68	0.18	0.69	0.15	1.55	0.16
1Yr	-0.07	0.21	0.12	0.17	-0.01	0.11	0.36	0.39	1.51	0.29	0.58	-0.01
2Yrs	0.04	0.22	0.07	0.22	0.09	0.10	0.10	0.26	-0.25	-0.04	0.73	0.16
3Yrs	0.09	0.12	0.22	0.17	0.08	0.05	0.13	0.18	-0.17	-0.07	0.93	0.23
4Yrs	0.16	0.16	0.36	0.19	0.13	0.09	0.13	0.21	-0.04	0.03	0.74	0.19
5Yrs	0.41	0.23	0.54	0.23	0.35	0.17	0.48	0.30	-0.04	0.05	0.77	0.20
7Yrs	0.23	0.12	0.46	0.16	0.18	0.09	0.40	0.17	0.11	0.08	1.02	0.20
10Yrs	0.30	0.15	0.57	0.22	0.34	0.14	0.43	0.17	0.40	0.07	1.24	0.19
15Yrs	0.34	0.13	0.64	0.21	0.48	0.15	0.52	0.18	0.54	0.07	1.51	0.13

Despite this impressive long-term performance on all datasets we notice that the KMAC-based PS algorithm has been quite subdued in recent times on some datasets. For example the algorithm achieves a 1% gain in 2013 and a 20% gain in 2012 using the TSE60 index in Canada. The same pattern can be seen in the FTSE100 in the UK, where the algorithm achieves an annualised compounded growth rate that was much less than its benchmark algorithm in 2013.

5.5.2 KMAC Performance Risk Analysis

Table 5.5 shows some of the KMAC risk measure on the six datasets. From the results, we see that the KMAC-based strategy has a volatility of returns that is higher than the one achieved by its respective benchmarks. However, the Sharpe Ratio generated by the algorithm is also significantly higher than the respective stock market indices. This is an indication that the KMAC-based strategy generates much better risk adjusted returns.

Table 5.5: Risk Statistics of the KMAC-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.41	0.18	0.32	0.16	0.42	0.17	0.50	0.23	0.44	0.19	0.37	0.13
α	0.13	0.00	0.11	0.00	0.20	0.00	0.10	0.00	0.14	0.00	0.30	0.00
β	1.44	1.00	1.24	1.00	1.48	1.00	1.37	1.00	1.29	1.00	1.35	1.00
γ	2.01	0.00	1.60	0.00	-0.90	0.00	2.69	0.00	2.39	0.00	4.03	0.00
Bl $\uparrow$	1.56	1.00	1.27	1.00	1.49	1.00	1.53	1.00	1.46	1.00	1.42	1.00
Br $\downarrow$	1.39	1.00	1.19	1.00	1.49	1.00	1.28	1.00	1.13	1.00	1.26	1.00
SR	1.28	0.37	1.47	0.78	1.55	0.59	1.14	0.53	1.32	0.44	2.63	0.64
MDD	-0.63	-0.40	-0.47	-0.30	-0.57	-0.41	-0.60	-0.47	-0.90	-0.64	-0.37	-0.37
PP	0.57	0.56	0.57	0.57	0.57	0.58	0.56	0.56	0.56	0.55	0.57	0.53
PP$\uparrow$	0.81	1.00	0.80	1.00	0.78	1.00	0.81	1.00	0.79	1.00	0.79	1.00
PP$\downarrow$	0.26	0.00	0.27	0.00	0.28	0.00	0.26	0.00	0.28	0.00	0.31	0.00

Further, we observe that the maximum drawdown on the six stock datasets are higher when compared with that achieved by their respective benchmarks. For example, we also note that the percentage of time the KMAC-based algorithm generates positive returns (PP) is very similar to the percentage of time the varous indices generate positive returns. However, when the market index was up, the KMAC-based algorithm generated positive returns more than 80% of time on all datasets. These impressive results suggest that the value add of the KMAC-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market. This fact is also confirmed by the Gamma coefficient in Table 5.5 and the asymmetric betas. Excluding the TSE60 in Canada, the KMAC-based PS algorithm demonstrates very impressive market-timing abilities. On the NASDAQ for example, when the market was up by 1%, the KMAC on average gained 1.53%. And when the market was down, the KMAC lost on average 1.28%. Similar asymmetric behaviour can be seen with all other datasets.

This impressive empirical performance clearly demonstrates that mean reversion and momentum can simultaneously occur on the same set of assets in the six markets in studied here. It is therefore evident that in the presence of both price reversal and price continuation, the original AC algorithm fails to perform optimally. To correct this shortcoming we have provided an important modification to the AC algorithm that seems to deal quite effectively with both momentum as well as reversal in comprehensive way. In our view the KMAC-based PS algorithm is indeed a very robust investment strategy.

5.5.3 KMAC Brokerage Costs Analysis

In this thesis we work on the assumption that there are charges on all transactions equal to a fixed percentage of the amount transacted. We adopt the proportional transaction cost model following Blum and Kalai (1999) and Borodin et al. (2004); that is, rebalancing the portfolio on any given day incurs transaction costs for both buy and sell orders.

At the beginning of the t^{th} trading day, the portfolio manager rebalances the portfolio from the previous closing price adjusted portfolio b_{t-1} to a new portfolio b_t. Specifically, we consider a transaction cost rate $c \in (0, 1)$. Thus, with transaction cost rate the total wealth achieved by the strategy becomes:

$$S_T^c = S_0 \prod_{t=1}^{T} \left[(\mathbf{b}_t, \mathbf{x}_t) \left(1 - \frac{c}{2} \sum_{k=1}^{m} \left| \widetilde{b}_t^k - b_t^k \right| \right) \right] \tag{5.32}$$

Transaction costs are therefore taken into account and we assume that a round-trip trading cost per trade of 10 basis points might just be enough to incorporate an estimate of price slippage and other costs as a single market friction coefficient.

Figure 5.3 shows the performance of the KMAC algorithm net of brokerage commissions against some benchmark strategies on our six datasets. In general, the potential outperformance of the KMAC strategy could be heavily affected if the commissions paid in order to execute every transaction is high or if the prices of stocks that are selected for inclusion in the portfolio rise systematically between investment decision and complete trade execution.

Figure 5.3: KMAC Brokerage Cost Analysis

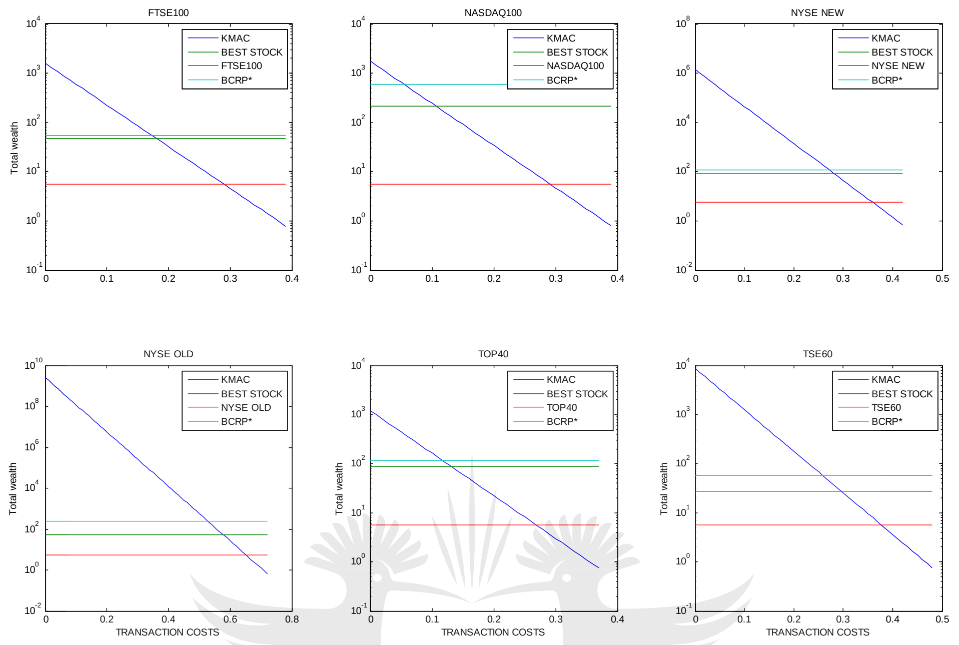


Figure 5.3 clearly demonstrates that our proposed KMAC investment algorithm can tolerate moderate proportional commission rates and still beat competing benchmarks, including the best stock, the Buy-and-Hold equally weighted market index and best constantly rebalanced portfolio in hindsight called BCRP*. The graphs in Figure 5.3 depict the total returns of the KMAC algorithm for varying proportional commission factors $c = 0.1\%, 0.2\%, \ldots$ It does clearly appear that our strategy can withstand reasonable brokerage commissions. For example, with a commission cost of $c = 0.1\%$ or 10 basis points (10 bps), the algorithm still beat the best stock, the market and the BCRP* portfolio in all markets we consider. In fact even with $c < 0.25\%$ our algorithm beats all its respective market indices on all data set.

5.6 PAMR AND CWMR VIA TAPR

This section applies the TAPR concepts developed earlier to some of the PS algorithms surveyed in Chapter 4. We observed in Chapter 4 that the so called state-of-the-art learning to select portfolio strategies in general and the PAMR and CWMR in particular, generally performed better on older datasets than on more recent and previously untested stock prices. This clearly shows that in their current specification these algorithms cannot easily be implemented by a portfolio manager. We also found that although the cumulative wealth achieved by these state-of-the-art approaches is higher over the full sample, their more recent performance is significantly lower than that achieved over the long-term.

Earlier in this chapter, we argued that many stochastic processes, including stock price relatives, have inherent noise that obscures the true underlying values. Therefore, applying raw price relatives, as has become the norm in the Online trading algorithms, is likely to be suboptimal. To be successful in today's market place, portfolio managers need to see through all the market noise that occurs on a daily basis and be able to identify the trend of prices. Classical approaches to this problem have been to apply a moving average, as in Li et al. (2012), or an exponential moving average to asset price time series to obtain a smoothed values.

We made the point that what matters to the investor is not the magnitude of the stock price change relative to its value in the previous period but its distance relative to some expected stock price in the current period. More specifically, when a stock deviates significantly from its expected value (or unobserved true value) as measured by the TAPR ratio, there is a corresponding high likelihood that this over or undervaluation will revert back to some equilibrium. Because our algorithm takes a bet only when a given stock price significantly deviates from its expected (trend) value, our new measure of price relative is therefore expected to yield better mean reversion characteristics and better perfomance where most state-of-the-art PS algorithms fail.

To further demonstrate the validity of our claim, we use two existing state-of-the-art trading strategies, namely the PAMR (Li et al. (2012)) and the CWMR (Li et al. (2013)), that have shown unimpressive performance on more recent datasets. Our strategy in relatively simple and intuitive (see Table 5.6 for a pseudo code). For each algorithm called ALG (PAMR, CWMR, etc...) we start by estimating the "true" unobserved stock price for market data. We then calculate the TAPR and use that as input into the Online PS algorithm. The new derived algorithms are referred to as the KPAMR and KCWMR. We combine the expert in the usual way.

Table 5.6: ALG $(\mathbf{X}, w, params)$

```
Inputs:      $X$: price relative
             $w$: window sise
             $params$: required parameters
Output:      $b$: portfolio weight matrix
Initialise:  $\mathbf{b}_1 = \frac{1}{m}\mathbf{1}$;
             $(Q, R, Z, V)$: kalman Filter parameters
for          $t = 1, 2, ..., n$
    1  Receive $\mathbf{x}_t$
    2  Calculate daily cumulative returns: $\mathbf{S}_t = \mathbf{S}_{t-1} \times (\mathbf{b}_t.\mathbf{x}_t)$
    3  Estimate the unobserved true price $\mathbf{p}_t^k$
    4  Calculate the TAPR: $\mathbf{x}_t = \frac{\mathbf{p}_t}{\mathbf{p}_t^k}$
    5  Update the portfolio weights for the following algorithms
       $\mathbf{b}_{t+1} = \begin{cases} \mathbf{CWMR}(\mathbf{X}, \phi, \varepsilon) \\ \mathbf{PAMR}(\mathbf{X}, \varepsilon) \end{cases}$
end
```

5.7 EXPERIMENTAL RESULTS

In this section, we present the results of our experiments using the six market datasets discussed in Chapter 3.

5.7.1 Parameter choices

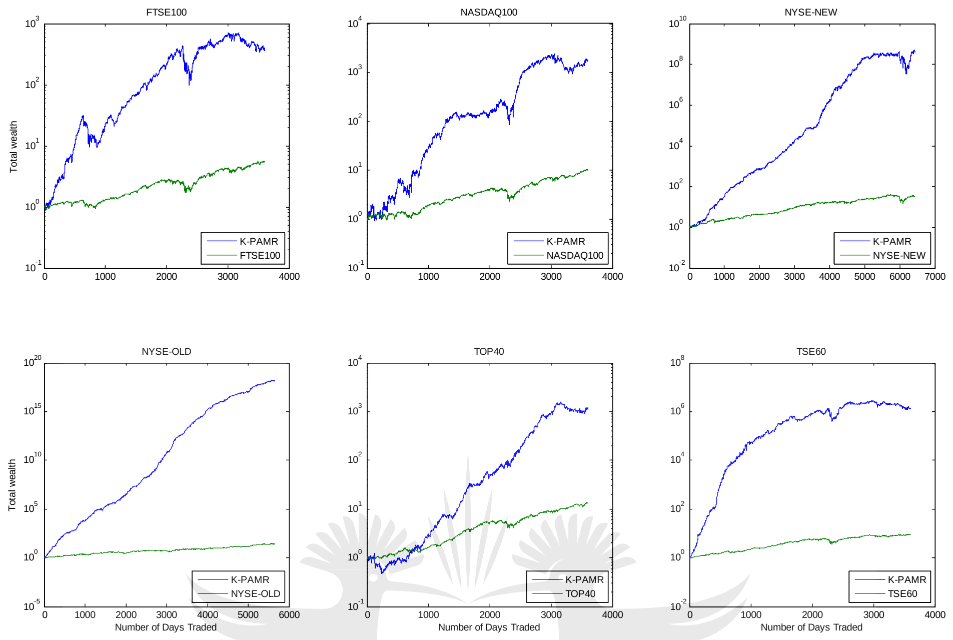
Parameter choices had to be made for all algorithms before performing any historical simulations. Table 5.7 below summarizes the various algorithms used for comparison together with some parameter choices.

Table 5.7: Parameter Settings

1	**Market:** Market strategy is simply the uniform BAH approach
2	**Best-Stock:** Best stock in the market that is a hindsight strategy;
3	**BCRP***. Best Constant Rebalanced Portfolios strategy in hindsight;
4	**Kernel** (W, L, c):. Nonparametric kernel-based moving window strategy with the parameter $W = 5, L = 10, c = 1$ (Gyorfi et al. [2006])
5	**NN**(W, L). Nonparametric nearest-neighbor-based strategy with parameter $W = 5, L = 10, p_l = 0.02 + 0.5\frac{l-1}{L-1}$ (see Gyorfi et al. [2008]).
6	**CORN**(W, ρ): We fix $\rho = 0.3$ and $W = 5$
7	**CWMR**$(\boldsymbol{\phi}, \boldsymbol{\varepsilon})$: We set $\boldsymbol{\phi} = \mathbf{2}$, and $\boldsymbol{\varepsilon} = \mathbf{0.5}$
8	**PAMR**$(\boldsymbol{\varepsilon})$: We set $\boldsymbol{\varepsilon} = \mathbf{0.5}$

Figure 5.3 and Figure 5.4 show the wealth accumulated by the PAMR and CWMR strategies via the TAPR input variable. We call these new algorithms the K-PAMR and the K-CWMR. We see that the TAPR-based alogrithms performs exceptionally well, not only against their benchmark equivalents but also amongst other state-of-the-art algorithms as shown in Chapter 2.

Figure 5.3: K-PAMR Cummulative Wealth



K-CWMR and K-PAMR algorithms peform best on all datasets analyzed within the class of algorithms surveyed. For example, on the NYSE (O) datasets after trading for 22 years, the total wealth achieved by the K-CWMR strategy and the K-PAMR strategies impressively increased from \$1 to almost an astronomical $\$1.27 \times 10^{18}$ and $\$6.62 \times 10^{17}$, respectively. The performance of these two algorithms is 192 and 143 times that achieved by their respective benchmarks, the CWMR and the PAMR using the NYSE(O) datasets. This result clearly shows that our proposed algorithms are competitive with the best base algorithm in all the experimental setups.

Figure 5.4: K-CWMR Cummulative Wealth Growth

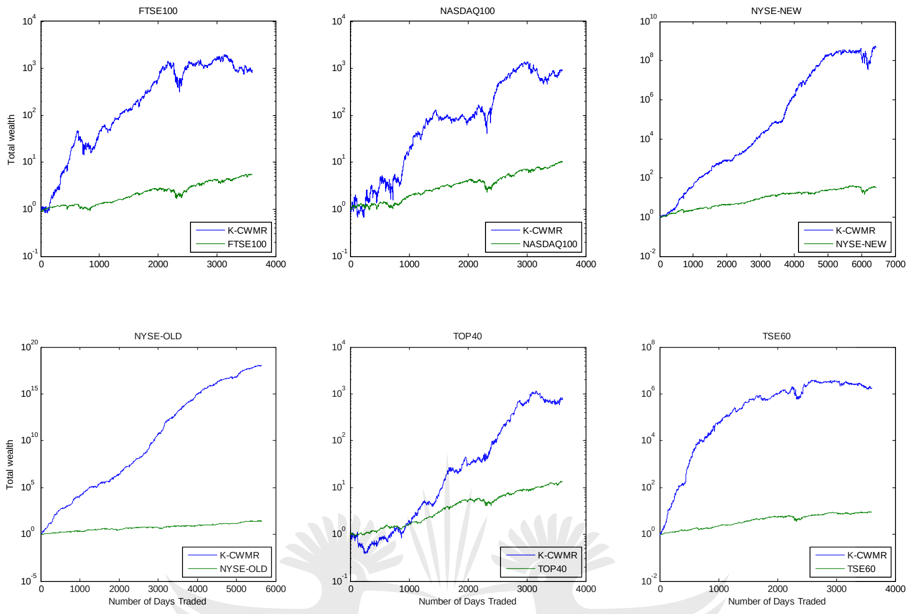


Figure 5.5 and Figure 5.6 show the performance sensitivity of the K-PAMR and K–CWMR-based PS selection algorithms net of brokerage commissions against some benchmark strategies on our six datasets.

Figure 5.5: K-PAMR Brokerage Cost Analysis

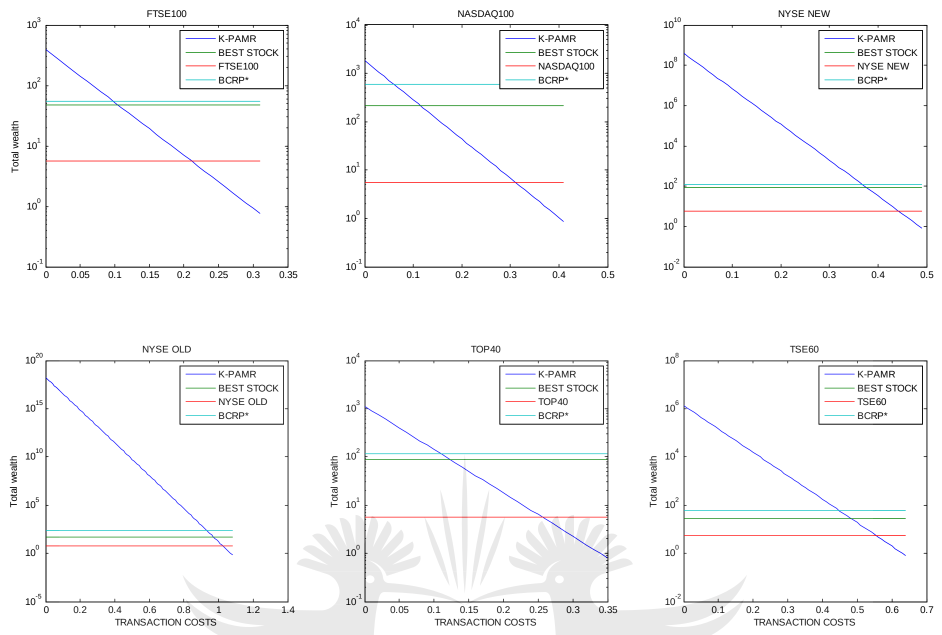
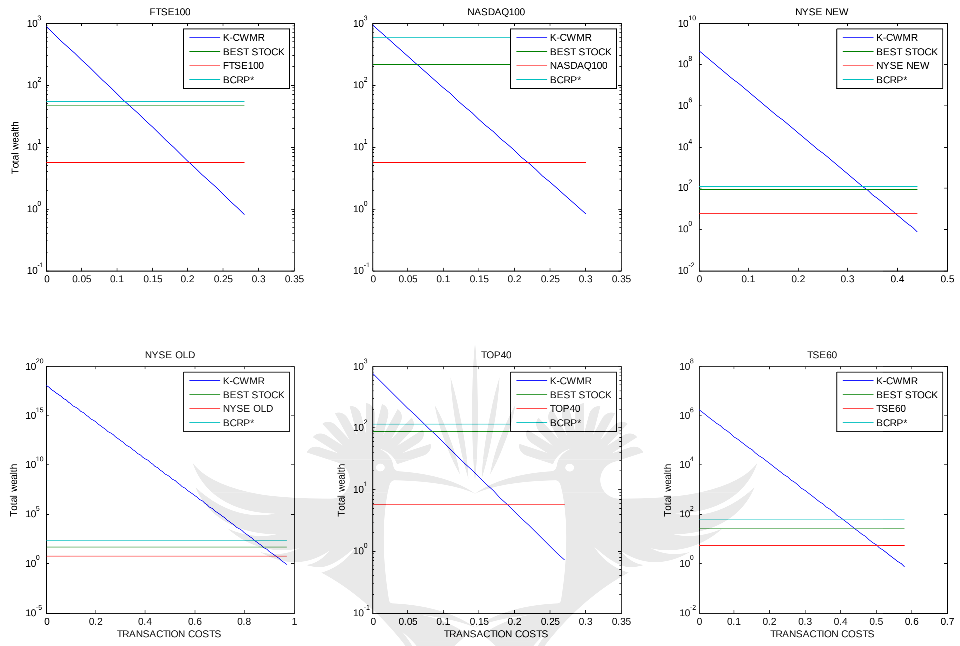


Figure 5.5 and 5.6 clearly demonstrate that our proposed K-PAMR and K-CWMR investment algorithms can tolerate reasonable proportional commission rates and still beat competing benchmarks, including the best stock, the Buy-and-Hold equally weighted market index and best constantly rebalanced portfolio in hindsight called BCRP*. As usual Figure 5.5 and 5.6 depict the total returns of the K-PAMR and K-CWMR algorithms for varying proportional commission factors $c = 0.1\%, 0.2\%, ...$

Figure 5.6: K-CWMR Brokerage Costs Analysis



5.7.2 Performance Comparison

Table 5.8 shows that all the TAPR-based strategies achieve considerably better results than the market index, the best stock in the market, as well as all their state-of-the-art benchmark equivalent strategies. Among all compared algorithms, the proposed K-CWMR, K-PAMR and K-MAC algorithms always achieve the best total wealth on all datasets, and are substantially better than the market index and the best stock in the market.

Table 5.8: Performance Comparison of PS Algorithms.

	FTSE100	NASDAQ	NYSE(N)	NYSE(O)	TOP40	TSE60
BEST STOCK	4.76E+01	2.15E+02	8.48E+01	5.33E+01	8.78E+01	2.78E+01
MARKET	5.00E+00	1.07E+01	1.82E+01	1.42E+01	1.37E+01	6.12E+00
UCRP	5.63E+00	1.05E+01	3.18E+01	2.67E+01	1.36E+01	9.18E+00
BCRP*	5.48E+01	5.92E+02	1.21E+02	2.49E+02	1.15E+02	5.83E+01
AC	6.82E+01	2.67E+02	5.59E+05	4.17E+07	4.54E+02	1.51E+03
MAC	1.81E+02	1.66E+03	6.33E+05	1.30E+09	6.52E+02	4.15E+03
CWMR	6.02E+01	6.87E+02	1.73E+06	6.63E+15	3.19E+01	1.64E+05
PAMR	6.05E+01	3.07E+02	1.20E+06	4.62E+15	3.92E+01	1.95E+05
K-AC	1.35E+02	5.00E+02	4.29E+06	2.58E+09	7.81E+02	1.06E+04
K-MAC	1.58E+03	1.74E+03	1.39E+06	2.52E+09	1.20E+03	8.79E+03
K-CWMR	4.41E+02	4.30E+03	2.86E+08	1.27E+18	1.22E+03	1.02E+06
K-PAMR	3.70E+02	1.21E+03	2.30E+08	6.62E+17	1.13E+03	7.23E+05
CORN	4.14E+01	1.37E+02	3.20E+05	3.61E+14	3.43E+02	2.16E+04
KERNEL	3.59E+01	5.55E+01	7.34E+02	8.68E+07	1.00E+02	2.44E+02
NN	1.59E+01	1.62E+02	6.38E+04	4.32E+10	8.71E+01	4.24E+02

5.8 CONCLUSION

We have presented a new Online approach to PS, which builds on existing state-of-the-art PS algorithms. Our algorithms combine powerful Online PS algorithms with ideas from signal processing and statistical learning to produce portfolios that substantially outperform their benchmark equivalents on the real market datasets. Historical simulations have demonstrated that suitable heuristics can achieve significant growth in portfolio wealth as compared to theoretically well-grounded approaches. De-trending price series and using the ratio of stock prices to their Kalman Filter trend improves the performance of the algorithm quite spectacularly in some cases. This impressive performance using data that are widely available casts some doubts on the market efficiency hypothesis at least on its weakest form. Only in the presence of weakly inefficient markets can these algorithms give the very good performance that we show in the examples. As argued by Gyorfi et al. (2006), this superior performance on datasets available to all investors may partially be explained by the fact that the dependence structures of the markets revealed by the proposed investment strategies are quite complex and, even though all information we use is publicly available, the way this information is exploited remains hidden from most traders.

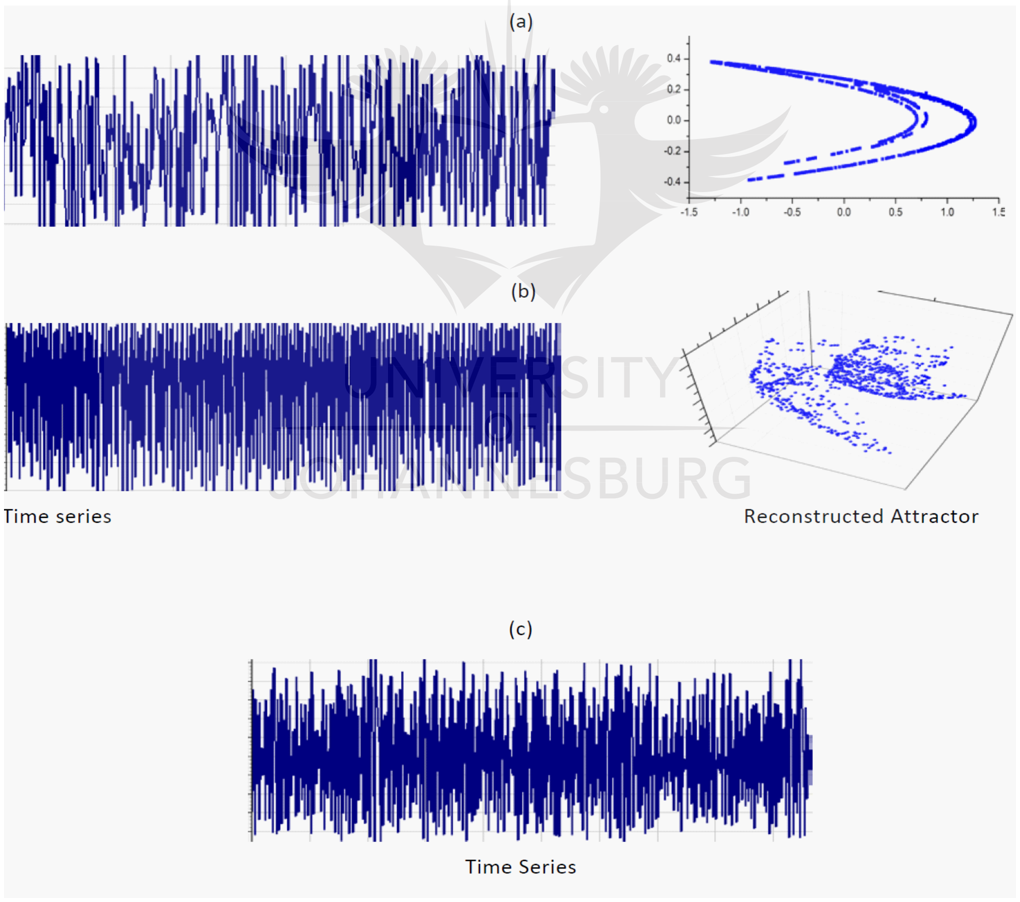
6.0 DELAY COORDINATE EMBEDDING AND SEQUENTIAL PORTFOLIO SELECTION ALGORITHMS

6.1 INTRODUCTION

In Chapter 4 of this thesis we presented a survey of some of the most effective (in terms of overall wealth growth) Online PS algorithms within the machine-learning literature. We showed that the so called state-of-the-art learning to trade PS strategies generally perform exceedingly well, but this performance is generally limited to older datasets. Using more recent and previously untested datasets, we found that not only the cumulative wealth achieved by these state-of-the-art approaches is significantly less impressive but, also, their more recent performance has been considerably poor, with some algorithms significantly underperforming their benchmarks in recent years. In Chapter 5 we introduced a new class of algorithms based on the idea of TAPR. Our algorithms combine powerful Online PS algorithms with ideas from signal processing and statistical learning to produce portfolios that substantially outperform their benchmark equivalents on real-market datasets. This new proposed class of algorithms was shown to achieve significantly better performance on all datasets than its equivalent benchmark algorithms. However, the main limitation shown by all the existing state-of-the-art algorithms as well as the new class proposed in this thesis is their implicit assumption that a representation of stock price time series in low dimension can capture all the complex and recurring patterns.

In truth, economic and financial time-series data may conceal complex recurring patterns. For example, system variables may cycle aperiodically along low-dimensional so-called "strange attractors" that are difficult to detect directly from time-series data. This situation occurs more regularly where the time series may appear mathematically random. To gain more insights into this pheonomenon, let us consider the time-series data shown in Figure 6.1 (panels (a), (b) and (c)). The data in panel (a) and panel (b) are generated by deterministic dynamic models while panel (c) is generated as a white noise. In particular, panel (a) plots one of the three variables from the Ikeda dynamical system while panel (b) plots one of the three variables from the Henon dynamical system. It is easy to see that the data from the Ikeda and Henon dynamical systems are not easily distinguished from the white noise as their underlying deterministic structures are well concealed.

Figure 6.1: Time series data and reconstructed attractors: (a) Ikeda Map, (b) Henon Map, (c) White Noise



"Phase space reconstruction" or Delay coordinate embedding is a method for uncloaking deterministic structures in time series data. Specifically, it reconstructs attractors present in real-world dynamical systems using time series data on a single variable (Broomhead, King, 1985; Schaffer, Kott, 1985; Kott et al, 1988; Williams, 1997). It is now acknowledged that the occurrence of irregular and complicated behaviour that is seemingly induced by the action of external random perturbations can be explained by using the theory of deterministic chaos. Previously, irregularity in stock price time series had been explained by traditional linear stochastic models, which assumed that such signals were projections of a superposition of external random influences on otherwise linear dynamical rules.

This chapter presents a novel non universal, nonparametric learning-to-trade sequential PS algorithm called "Delay Coordinate Embedding algorithm for PS (DCEPS)". As in Gyorfi et al. (2006) and Li et al. (2012), our proposed DCEPS algorithm has its foundations in the similarity-driven non parametric Online learning algorithms. Our new algorithm allows us to construct asymptotically optimal investment strategies in the financial market without prior knowledge of the statistical properties of stock prices. To this end the DCEPS approach to prediction has two main steps; first the DCEPS algorithm attempts to identify the state of the system at some time t, then the algorithm searches the past history of observations for "similar" states in order to predict possible future outcomes. By studying the evolution of the observable outputs following the similar states the DCEPS algorithm infers informations about the future path of the system.

The main novelty here is that our DCEPS algorithm builds on Takens delay coordinate embedding theorem, which allows us to construct a data matrix of overlapping samples and therefore increase the precision of parameter estimates. The underlying intuition is that the dynamics of the entire system are embedded in the history of each stock price series. Unlike existing state-of-the-art similarity-driven PS algorithms (Gyorfi et al. (2006, 2007, 2008), Li et al. (20112)), the DCEPS is derived from a time-delay embedding of multiple time series that allows us to capture nonlinear information found in complex dynamical systems of stock returns. By creating a time-lagged version of the original stock returns series, our methodology allows the portfolio manager to discover hidden patterns normally not detected in a linear space. It turns out that the new DCEPS algorithm is in fact a generalisation of some earlier state-of-the-art Online PS selection algorithms, including the CORN (Li et al. (2012)) and the NN (Gyorfi et al. (2008)) algorithms where pattern identification is performed in lower dimension.

To demonstrate the robustness of this new methodology we evaluate our DCEPS algorithm against benchmark Online portfolio allocation techniques using six stock market datasets of which four were previously untested. In all these datasets our algorithm substantially outperforms existing state-of-the-art Online stock selection techniques without much additional computational demand or modelling complexity.

The rest of the chapter is organised as follows. In Section 6.2 the mathematical model is described, and related results are surveyed briefly. In Section 6.3, we briefly review the theory of state-space reconstruction. Embedding parameters are described in Section 6.4. In section 6.5 we present the mechanics of our delay coordinate embedding for portfolio selection. Section 6.6 deals with expert combination while Section 6.7 presents our empirical results. Section 6.9 concludes this chapter.

6.2 MATHEMATICAL MODEL

The mathematical background in this chapter is very similar to the one built earlier in Chapter 2 of the thesis. We will repeat the arguments here for self containment.

In this chapter, we consider a market of m stock prices such that a market vector $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^m)$ represents the vector of prices for $j = 1, 2, ..., m$. The change in security prices during the t^{th} trading period is represented as a stock market vector $\mathbf{x}_t = (x_t^1, x_t^2, ..., x_t^m) \in \mathbb{R}_m^+$ where $\mathbf{x}_t$ is the vector m of non-negative numbers representing the TAPR for the trading period t. Using the same set up as the one in Chapter 3, we can show that starting with an initial wealth $\mathbf{S}_0$, after n trading periods, the investment strategy $\mathbf{B}$ achieves the wealth:

$$\mathbf{S}_n = \mathbf{S}_0 \prod_{i=1}^{n} \sum_{j=1}^{m} b_t^j \left(\mathbf{x}_1^{t-1}\right) x_t^j = \ \mathbf{S}_0 \exp\left\{ \sum_{i=1}^{n} \log \mathbf{b}_t^\intercal \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t \right\} \tag{6.1}$$

We showed earlier that the fundamental problem in investment is that of maximising $\mathbf{S}_n(\mathbf{B})$, which is equivalent to maximising the following average growth rate

$$\mathbf{W}_n(\mathbf{B}) : \mathbf{b}_i^*\left(\mathbf{x}_1^{i-1}\right) = \arg_{\mathbf{b}} \max \mathbf{E}\left\{ \log \mathbf{b}_t^\intercal \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t \mid \mathbf{x}_1^{i-1} \right\} \tag{6.2}$$

Instead of trying to directly maximise the complex average growth rate, $\mathbf{W}_n(\mathbf{B})$, the practice in the machine-learning community is design algorithms that will achieve asymptotically the same growth rate in wealth without any strong assumptions regarding the statistical properties governing stock prices. This is exactly what the DCEPS will achieve in this chapter.

Several Online PS algorithms have been proposed in the literature including Kelly (1956); Breiman (1961); Cover (1991); Ordentlich and Cover (1996); Helmbold et al. (1996); Borodin and El-Yaniv (1998); Borodin et al. (2000, 2004); Stoltz and Lugosi (2005); Hazan (2006); Györfi et al. (2006); Blum and Mansour (2007); Levina and Shafer (2008); Györfi et al. (2008)). Li et al. (2012) provide a comprehensive survey and an intuitive grouping of several major PS algorithms.

The current chapter builds on the so-called "Pattern-Matching" or similarity-driven PS strategies. Similarity-driven PS strategies aim to optimise the trading strategy by detecting potentially similar information from historical market sequences of price relative. Some of the more popular similarity-driven PS strategies include the Kernel-based PS algorithm (Gyorfi et al. (2006)), the Nonparametric Nearest Neighbor (Gyorfi et al. (2008)) and the Correlation-driven non parametric empirical PS algorithm (CORN) of Li et al. (2012).

6.3 THEORY OF STATE-SPACE RECONSTRUCTION

Traditionally, the behaviour of financial time series has been analyzed using linear stochastic models. There is now considerable evidence that linear stochastic models are not able to account for all the complex behaviour observed in stock prices (see Mantegna and Stanley, (2000) or Johnson et al. (2003)). It is now widely accepted that the complex nature of financial time series can be attributed to the fact that financial markets are nonlinear stochastic, chaotic or a combination of both. For example, Osborne (1959) and Malkiel (1990) characterised financial time series using the theory of Brownian motion while Mandelbrot

(1998) proposed a fractional Brownian motion to characterise and analyze financial time series. In other studies, financial time series have been characterised using nonlinearity (Brock et al.(1991)), chaos and fractals (Hsieh (1991), Lorenz (1993), Peters (1996)), scaling behaviour (Mantegna and Stanley (1995) and (1996)), and self-organised criticality (Bak and Chen (1991); Shlesinger et al (1993)).

6.3.1 Mathematical Foundations of the State-Space Reconstruction

The theory of embedding is a way to move from a temporal time series of measurements to a state space "similar" -in a topological sense-to that of the underlying dynamical system we are interested in analysing. The mathematical basis of continuous dynamical modelling is formed by differential equations of the following type:

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}\left(\mathbf{x},\alpha\right) \tag{6.3}$$

where the real variable t denotes time, $\mathbf{x} =(x_1, x_2, \ldots, x_n)$ represents the state variables of the system, depending on time t and on the initial conditions. α_j are parameters of the system and $\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_n)$ is a nonlinear function of these variables and parameters. In experimental studies, it is not always possible to measure the complete state of a system. When analysing a dynamical system only a few observable quantities are available which, in the absence of noise, are related to the state-space coordinates by:

$$\mathbf{s}\left(t\right) = \mathbf{h}(\mathbf{x}\left(t\right)) \tag{6.4}$$

where $\mathbf{h}$ is an unknown nonlinear function called measurement function.

State-space reconstruction techniques were comprehensively analyzed by Packard et al. (1981) and Takens (1981). Takens(1981) for example, proved that under certain conditions the dynamics on the attractor of the underlying system has a one-to-one correspondence with measurements of a limited number of variables. This argument suggests that by measuring few variables we are able to reconstruct a one-to-one correspondence between the reconstructed state space and the original, which means that it is possible to identify unambiguously the original state space from measurements.

In order to explain the relationship that occurs between the reconstructed and the real state space, let us consider the following dynamical system (see Strozzi et al (2007))

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x})\,;\ \mathbf{x} = (x_1, x_2, \ldots, x_n) \tag{6.5}$$

We can define $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ as follows:

$$\mathbf{y} = (x_1, dx_1/dt, d^2x_1/dt^2, \ldots, d^{n-1}x_1/dt^{n-1})$$

then the equations of motion take the form

$$\frac{dy_j}{dt} = y_{j+1} \tag{6.6}$$

$$\frac{dy_n}{dt} = \mathbf{G}(y_1, y_2, \ldots, y_n) \tag{6.7}$$

for some function **G**. This new coordinate system allows us to proceed from the state space $(x_1, x_2, \ldots, x_n)$ to the space of derivatives $(x_1, dx_1/dt, d^2x_1/dt^2, \ldots, d^{n-1}x_1/dt^{n-1})$ in order to model the dynamics of the time series. Takens (1981) shows that instead of derivatives, $\{s(t), \dot{s}(t), \ddot{s}(t), ...,\}$ one can use delay coordinates, $\{s(t), s(t+\Delta t), s(t+2\Delta t), ...\}$, where Δt is a suitably chosen time delay. In fact, looking at the following approximation of the derivative of $s(t)$:

$$\frac{d\mathbf{s}(t)}{dt} \cong \frac{\mathbf{s}(t+\Delta t) - \mathbf{s}(t)}{\Delta t} \tag{6.8}$$

$$\frac{d^2\mathbf{s}(t)}{dt^2} \cong \frac{\mathbf{s}(t+2\Delta t) - 2\mathbf{s}(t+\Delta t) + \mathbf{s}(t)}{2\Delta t^2} \tag{6.9}$$

it is clear that the new information brought from every new derivative is contained in the series of the delay coordinates. The advantage of using delay coordinates instead of derivatives is that in the case of high dimensions high-order derivatives will tend to amplify considerably the noise in the measurements. Restating in simpler terms, observed measurements are real-valued projections of unknown nonlinear combinations of the underlying state variables of the system and, therefore, completely retain all information of the state variables.

The theoretical justification for our new complex nonlinear PS system is therefore based on the Takens (1981) delay embedding theorem. In fact, Takens (1981) proved that if the dimension of the embedding space is large enough, then the reconstructed phase-space (RPS) is topologically equivalent to the original state space that generated the time series in the first place. Therefore, characterisations and predictions based on the RPS are considered valid and similar to those made if the original state space were available. We will use this characterisation and prediction based on the reconstructed phase-space for multiple time series of stock prices with a varying embedding dimension d and an embedding lag τ. The choice of these two parameters will be dealt with in subsequent sections.

6.3.2 State-Space Reconstruction

Reconstruction of the state space implicitly assumes that the past observed measurements of a time series contain information about the unobserved state variables that can be used to define a state at the present time. This is typically done using delay coordinates. Assuming a predictive reconstruction, the τ-dimensional delay coordinate vector at some time t is defined by

$$\mathbf{x}_t = (x_t, x_{t-\tau}, dx_{t-2\tau}, \ldots, x_{t-(d-1)\tau}) \tag{6.10}$$

A major limitation of the embedding theory is that Takens' theorem has been proven in the case of noise-free systems only. Unfortunately, there is always a certain amount of noise, $\sigma(t)$, in real data such as stock prices. Such noise can appear in both the measurements and the dynamics (see Diks (1999)). Dick (1999) argues that observational noise similar to that studied here-i.e. $\mathbf{s}(t) = \mathbf{h}(\mathbf{x}(t)) + \boldsymbol{\sigma}(t)$-does not affect the evolution of the dynamical system, whereas dynamical noise acts directly on the state of the dynamical system influencing its evolution; for example: $d\mathbf{x}/dt = \mathbf{F}(\mathbf{x}, \alpha) + \boldsymbol{\sigma}(t)$. Because stock prices are inherently noisy and the effects of a relatively small amount of observational noise may put severe restrictions on the characterisation and estimation of the properties of the underlying dynamical system we use the linear filters approach in order to remove the observational noise.

6.4 EMBEDDING PARAMETERS

In real applications, the appropriate choice of the time delay τ and the calculation of an embedding dimension, d, are fundamental for estimating Equation 6.10. Much of research on state-space reconstruction has centred on the problems of choosing the time-delay and the embedding-dimension which constitutes the parameters of the reconstruction for delay coordinates.

6.4.1 Chosing the Optimal Time Delay

If the time delay chosen is too small, there is almost no difference between the elements of the delay vectors, since all points are accumulated around the bisectrix of the embedding space: this is called redundancy (Casdagli et al., 1991). However, when τ is very large, the different co-ordinates may be almost uncorrelated. In this case the reconstructed trajectory may become very complicated, even if the underlying "true" trajectory is simple: this is called "irrelevance". Unfortunately no rigorous way exists of determining the optimal value of τ.

Two common methods for choosing such a τ are the autocorrelation function and average mutual information. The basis of each of these approaches is heuristic and it is not guaranteed for the values obtained using the different approaches to converge.

6.4.1.1 Autocorrelation Function The autocorrelation function measures the expectation of observing the $x_n + \tau$ at a time τ later when x_n is observed. It is a second-order moment function and is given by

$$C_\tau = \frac{\sum_{n=1}^{N} (x_n - \overline{x})(x_{n+\tau} - \overline{x})}{\sigma^2} \tag{6.11}$$

where

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N}(x_i) \text{ and } \sigma^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_i - \overline{x})^2 \tag{6.12}$$

are the mean and variance of the data signal respectively. The autocorrelation function of deterministic systems decays exponentially with increasing lag. Using a τ at which C_τ attains its first zero makes the coordinates linearly uncorrelated and, hence, a good approximation for the optimal τ. Some authors note that in some cases such a criterion completely removes any connection between coordinates making proper reconstruction impossible (Kantz and Schreiber, (1997)).

6.4.1.2 Mutual Information The average mutual information uses ideas from Information Theory to define an optimal time delay τ. The average mutual information is the amount of information (in bits) learned by measurements of x_n through measurements of $x_{n+\tau}$ and is given by:

$$I_\tau = \sum_{x_n, x_{n+\tau}} p\left(x_n, x_{n+\tau}\right) \log_2 \frac{p\left(x_n, x_{n+\tau}\right)}{p\left(x_n\right) p\left(x_{n+\tau}\right)} \tag{6.13}$$

where $p\left(x\right)$ and $p\left(x, y\right)$ are the probability and joint probability functions respectively.

Fraser and Swinney (1986) suggest that the first τ where the first minimum of I_τ occurs ensures attractor unfolding in a time delay embedding. I_τ is the nonlinear equivalent of C_τ that can be used to determine the value τ that makes coordinates independent enough in a time delay reconstruction but still correlated to each other.

6.4.2 Choice of the Optimal Embedding Dimension

Moreover, similar problems are encountered for the embedding dimension. Working in a dimension larger than the minimum required by the data will lead to excessive requirements in terms of the number of data points and computation times necessary when investigating different questions such as, for example, invariants calculation, prediction, etc. Furthermore, noise by definition has an infinite embedding dimension, so it will tend to occupy the additional dimensions of the embedding space where no real dynamics is operating and, hence, it will increase the error in the subsequent calculations. On the other hand, by selecting an embedding dimension lower than required, we would not be able to unfold the underlying dynamics; i.e. the calculations would be wrong since we do not have an embedding.

6.4.2.1 Singular Value Decomposition (SVD) SVD attempts to find consistency in the results by trying a range of values of the embedding dimension d. One constructs a trajectory matrix from the data using a time delay $\tau = 1$. By decomposing the trajectory matrix into orthogonal coordinate space, one hopes that the corresponding distribution of singular values persists for all embedding dimensions greater than some minimum value. Thus, a rational basis for selecting the embedding dimension is established. However, Mees et al. (1987) have noted that the use of SVD in dimension estimation is limited because the number of singular values may depend on the details of the embedding and quality of the data as much as they do on the dynamics of the system.

6.4.2.2 False Nearest Neighbours Another method for determining the minimum embedding dimension is the false nearest neighbour (FNN) method of Kennel et al. (1992). The idea behind FNN is that if two points are neighbours in a reconstructed space of dimension d and fail to remain neighbours in dimension $d+1$, then they are false. The necessary minimum embedding dimension occurs at that dimension when all true nearest neighbours are found; that is, they do not significantly separate in moving to a higher dimension. This assures that embedded points have state space neighbours that are a result of the dynamics and not an artefact of being projected in low-dimensional space.

6.5 DELAY COORDINATE EMBEDDING FOR PORTFOLIO SELECTION (DCEPS)

This section provide a detailed analysis of the basic steps needed to implement our DCEPS algorithm.

The mechanics of the DCEPS-based nonparametric sequential investment strategy consists of four basic steps. In the first step we perform a multivariate phase-space reconstruction of the stock price returns in our universe. At each time step t the algorithm identifies a list of similar historical price relative sequences whose measure of similarity with the most recent market window are within a given pearson correlation threshold. In the third step, the DCEPS-based PS algorithm constructs a vector of portfolio weights that maximises the returns following observation of similar market sequences. Finally, the algorithm combines the various portfolio "experts" according to the idea of the constantly rebalanced portfolio.

6.5.1 Phase-Space Reconstruction of Multiple Trend-Adjusted Price Relatives

The Reconstructed Phase-Space (RPS) is a time-delay embedding of a given time series that allows us to capture nonlinear information found in complex dynamical systems of stock returns. Our technique creates a time-lagged version of a the original stock returns in order to discover hidden patterns normally not detected in a linear space. This novel approach provides the basis for our data-mining-based stock-selection process.

To illustrate the concept of reconstructed phase-space for Online PS, we generalise the specific problem of time series prediction when only a scalar time series is available. Very often, a high dimensional physical system is only observable through a single scalar variable. The method of time-delay embedding is widely applied to estimate the evolution of the underlying vector field. From a scalar time series of TAPR $\{\mathbf{x}_t\}_{t=1}^{n}$ of n observations, we reconstruct a vector time series with evolution topologically equivalent to the original system via the following transformation

$$\mathbf{x}_t \longrightarrow \left(x_t, x_{t-\tau}, x_{t-2\tau}, ..., x_{t-(d-1)\tau}\right) \tag{6.14}$$

The embedding window is thus:

$$\tau_d = (d-1)\,\tau \tag{6.15}$$

In this way we obtain d observations from the vector $\mathbf{x}_t$, skipping every τ^{th} values . Vectors of $\mathbf{x}_t$ are filled with repeated observations from x_t in the sense that two vectors $\mathbf{x}_t$ and $\mathbf{x}_{t+\tau}$ will have almost the same elements.

Our approach uses a generalisation of this scalar RPS for a multivariate time series of stock trend adjusted price relatives (TAPR) and defines the multivariate phase-space reconstructed time series as

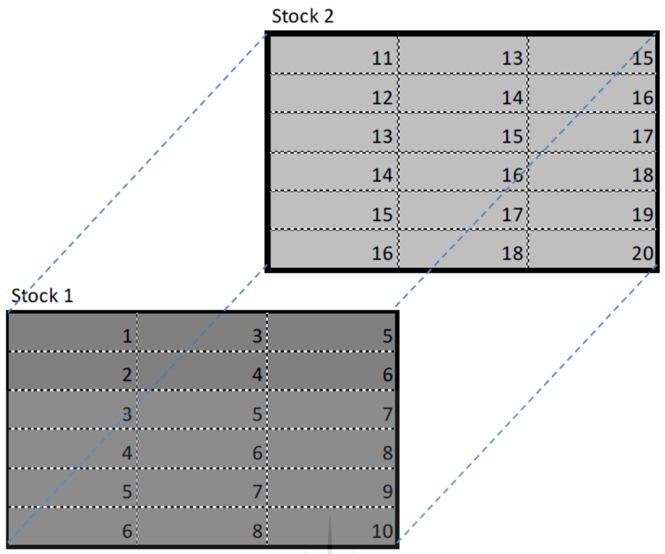
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ . \\ . \\ . \\ \mathbf{x}_m \end{bmatrix} = \begin{bmatrix} x_{1,t}, x_{1,t-\tau}, x_{1,t-2\tau}, ..., x_{1,t-(d-1)\tau} \\ x_{2,t}, x_{2,t-\tau}, x_{2,t-2\tau}, ..., x_{2,t-(d-1)\tau} \\ . \\ . \\ . \\ x_{m,t}, x_{m,t-\tau}, x_{m,t-2\tau}, ..., x_{m,t-(d-1)\tau} \end{bmatrix} \tag{6.16}$$

As an example, consider two stocks s_1 and s_2, whose ordered squences of price relatives range from 1 to 10 and from 11 to 20. Embedding two ordered sequences of numbers (i.e.,$x_{s_1}(t) = \{1, 2, 3, ..., 10\}$ and $x_{s_2}(t) = \{11, 12, 13, ..., 20\}$) for time delay $\tau = 2$ and $d = 3$ produces two 6×3 matrices-RC_{s_1} and RC_{s_2}:

$$RC_{s_1} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \\ 5 & 7 & 9 \\ 6 & 8 & 10 \end{pmatrix} \text{ and } RC_{s_2} = \begin{pmatrix} 11 & 13 & 15 \\ 12 & 14 & 16 \\ 13 & 15 & 17 \\ 14 & 16 & 18 \\ 15 & 17 & 19 \\ 16 & 18 & 20 \end{pmatrix}$$

The 3D figure below (Figure 6.2) shows an example of an RPS from a 2-time series representing historica stock prices. Each original time series is time-delay embedded with an embedding dimension of 3 and an embedding lag of 2 to create the RPS. Generalising this set up to multiple time series of stock prices is trivial.

Figure 6.2: Cubic Representation of the RPS



In order to unfold the dynamics from s_1 and s_2 effectively, we need to choose very carefully the value of parameters τ and d. As argued earlier, a number of procedures exist, such as the Average Mutual Information criterion and the autocorrelation function for the time delay and the False Nearest Neighbours criterion for the embedding dimension. Despite the extensive literature regarding the optimal choice of the embedding dimension and embedding lag when considering only a scalar time series the multivariate case presents significantly more challenges given the possibilty of multiple paramter values for τ and d. One simple way to avoid the complexity associated with multiple-parameter estimates is to reconstruct the multivariate time series with the same embedding lag and embedding dimension but allow the model to circle through a range of τ and d. In fact, our historical simulations shows that the model performance is not very sensitive to the parameter τ. For this reason we decided to set $\tau = 1$ while $1 \leq d \leq 10$.

6.5.2 Pattern Matching

This section deals with the pattern search step where we use a multivariate version of the RPS from stock price time series data as the basis for our data-mining-based stock selection process. At any given trading instance t, we consider a window length k, and an embedding dimension d with $\tau = 1$. We call $x\left\{d\right\}_{t-k}^{t-1}$ the most recent window of price relative for a given embedding dimension d and a window length k $(1 \leq k \leq 10)$. For simplicity we will write $x\{d\}_{t-k}^{t-1} = x_{t-k}^{t-1}$ for the rest of this chapter. It is important to notice that x_{t-k}^{t-1} has a cubic dimension $k \times d \times m$ and if we set $d = 1$ we could always resise the vector x_{t-k}^{t-1} as a $k \times m$ matrix. In this case the DCEPS algorithm is reduced to the CORN algorithm of Li et al. (2012) or the NN algorithm of Gyorfi et al (2008) with minor modifications. It is therefore reasonable to consider the DCEPS-based PS algorithm as a generalisation of most existing similarity-driven PS algorithms, including those proposed by Gyorfi et al. (2008) and Li et al. (2013).

Now let us consider any other discretised return vectors, x_{i-k}^{i-1}, in the whole history of the market sequence whose correlation with the most recent window return vectors, x_{t-k}^{t-1}, is larger than or equal to ρ. Such time instant

$$N_i = \left\{k < i < t; correl\left(x_{i-k}^{i-1}, x_{t-k}^{t-1}\right) \geq \rho \mid 1 \leq d \leq 10, \ -1 \leq \rho \leq 1\right\}$$

is called matching time. If $N_i == \varnothing$,we simply set $b = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$.

In order to search for matching patterns with the most recent window, we use a multi-dimentional version of the pearson correlation measure as a measure of similarity. In order to identify complex market conditions including mean reversion and momentum, the most common measure of similarity has been the use of the Euclidian distance metric. However, given that mean reversion and trend-following price relatives have essentially opposite effects, it is therefore likely that using a directionless measure like the Euclidian distance metric, could result in poor predictive signals (see Li et al. (2012)). In our view the Euclidean distance that forms the basis of most similarity-driven algorithms for PS cannot optimally exploit the full information content of complex market patterns as it concentrates only on the strength and not the direction information of the market movements. Li et al. (2012) illustrate this point clearly and propose the use of the standard pearson correlation as a measure of similarity instead of the Euclidian distance. In this chapter we propose a correlation measure that generalises standard pearson correlation by measuring both the direction and strength of instances of similarities in higher dimensions. Given two vectors $\mathbf{X}$ and $\mathbf{Y}$ of dimension $k \times m$, our correlation coefficient is given by the following equation

$$\rho = \mathbf{correl}\left(\mathbf{X},\mathbf{Y}\right) = \frac{\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)\left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)}{\sqrt{\left(\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)^2\right)\left(\sum_i \sum_j \left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)^2\right)}}$$

where $\overline{\mathbf{X}}$ and $\overline{\mathbf{Y}}$ represent the average of $\mathbf{X}$ and $\mathbf{Y}$ of dimension $k \times m$. To obtain a 2D vector from a 3D structure we simply calculate the average of x_{t-k}^{t-1} along the d dimension. Different values of ρ will likely generate different paths in cumulative portfolio wealth. The search for highly correlated matching historical patterns (higher values of ρ) will generate a smaller number of matching instances, making the optimisation potentially sensitive to fewer inputs and, therefore, unreliable. A threshold value of ρ closer to -1 will generate too many matching instances with the risk that many of these might be spurious. In this chapter we decided to set the threshold value to any positive correlation between the most recent discretised window and any window of the same length in the entire history of TAPR ($\rho = 0.5$).

After locating these historical matches the DCEPS algorithm constructs a fixed portfolio vector to optimise the returns for the trading periods following each matching feature.

$$h\left(x_1^{t-1}\right) = \underset{b \in \Delta_d}{\arg\max} \prod_{k<i<t, d>0, \tau>0} \langle b, X_{N_i} \rangle \tag{6.17}$$

6.5.3 DCEPS Portfolio Optimisation

To optimise the return for the trading periods following each matching sequence Gyorfi et al. (2008) and Li et al. (2012) propose a strategy to learn the optimal portfolio by following the idea of the BCRP (Cover 1991). The idea is to invest in a portfolio that would have maximised (in the BCRP sense) the portfolio returns following instances of similar matches. However, our own simulations show that portfolios generated along these lines display a very high degree of concentration and in most case the total available capital ends up being invested in one stock only. This, of course, renders these strategies unrealistic and unactractive for practical applications. This chapter proposes an alternative optimisation procedure that can generate both robust as well as diversified portfolios with empirical performance that easily surpasses that of existing state-of-the-art algorithms in most datasets.

To find the portfolio weights that maximise the returns for the trading periods following each match we use the intuition provided by the PAMR of Li et al. (2012). The PAMR algorithm has been proven to be a very robust investment strategy, as it exploits the mean reversion property of financial markets. Using the theoretical foundations of the Online passive aggressive learning (Crammer et al. 2006). As shown earlier, PAMR's key idea is to formulate a loss function that can effectively exploit the short-term mean reversion property of stock prices. Given a portfolio vector $\mathbf{b}$ and a price relative vector $\mathbf{x}_t$, the PAMR defines the following loss function for the t^{th} trading day

$$l_{\varepsilon}\left(\mathbf{b};\mathbf{x}_t\right) = \begin{cases} 0 & \mathbf{b}.\mathbf{x}_t \leq \varepsilon \\ \mathbf{b}.\mathbf{x}_t - \varepsilon & otherwise \end{cases} \tag{6.18}$$

Where $0 \leq \varepsilon \leq 1$ is the mean reversion parameter. Therefore according to the Passive Aggressive paradigm the optimisation problem can be formulated as

$$\mathbf{b}_{t+1} = \underset{\mathbf{b}\epsilon\Delta_m}{\arg}\ \min \frac{1}{2}\left\|\mathbf{b}-\mathbf{b}_t\right\|^2 \tag{6.19}$$

$$st \qquad l_{\varepsilon}\left(\mathbf{b};\mathbf{x}_t\right) = 0$$

The formulation set out above attempts to find an optimal portfolio by minimising the deviation from the last portfolio $\mathbf{b}_t$ under the condition of satisfying the constraint of zero loss. This loss is zero when the portfolio return is less than the reversion threshold ε, and otherwise the loss grows linearly with respect to the daily return.

The PAMR algorithm has a very simple and intuitive interpretation; On the one hand, the model specification above passively keeps the last portfolio; that is, $\mathbf{b}_{t+1} = \mathbf{b}_t$, whenever the portfolio daily return is below the threshold ε. In that case the model anticipates a high likelihood of mean reversal in the next period, therefore avoiding unecessary portfolio rebalancing and transaction costs. On the other hand, whenever the loss is nonzero, the algorithm aggressively updates the solution by forcing it to strictly satisfy the constraint $l_{\varepsilon}\left(\mathbf{b}_{t+1};\mathbf{x}_t\right)$. This simply mean that when the portfolio daily return is above the threshold, ε, the algorithm actively rebalances the portfolio weights. Li et al. 2012 propose the following solution for the PAMR algorithm:

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \tau_t \left(\mathbf{x}_t - \overline{\mathbf{x}}_t \mathbf{1}\right) \tag{6.20}$$

where

$$\overline{\mathbf{x}}_t = \frac{\mathbf{x}_t 1}{m}, \text{ and } \tau_t = \frac{l_\epsilon^t}{\left\|\mathbf{x}_t - \overline{\mathbf{x}}_t 1\right\|^2} \tag{6.21}$$

Our use of the PAMR-based optimization paradigm is therefore quite straightforward. After identifying similar sequences from the most recent window of TAPR we form a portfolio that has the highest likelihood of mean reverting in the very next period.

Table 6.1 displays the pseudo code of our implementation of the DCEPS algorithm.

Table 6.1: Pseudo Code of the **DECPS**$(\mathbf{x}, d, k)$

```
Inputs  $d$ : the embedding dimension
        $k$ : the window length of the recent past
        $\varepsilon = 0.1$ : the sensitivity parameter
        $\rho = 0.5$ : correlation coefficient threshold
        $\mathbf{x}$ : multivariate RPS of TAPR
Output  $\mathbf{b}_t$ : expert's portfolio weights for the $t^{th}$ trading day
for     $t = 1, 2, ..., n$ do
    1   receive $\mathbf{x}_t$, TAPT
    2   if $t \leq w + 1$ then
    3       $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
    4   end
    5   for $i = w + 1 \; to \; t - 1$ do
    6       if correl $\left(\mathbf{x}_{i-k}^{i-1}, \mathbf{x}_{t-k}^{t-1}\right) \geq \rho$ then
    7           $N_i = \underset{w+1 \leq i \leq t-1}{corr} \left\{\mathbf{x}_{i-k}^{i-1}, \mathbf{x}_{t-k}^{t-1}\right\} \geq \rho$
    8       end $if$
    9   end $for$
    10  if $N_i == \varnothing$ then
    11      $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
    12  end $if$
    13  else
    14  Suffer a loss $l_\varepsilon^t = \max\left(0, \mathbf{b}_t \mathbf{x}_t - \varepsilon\right)$
    15  Set parameter $\tau_t = \frac{l_\varepsilon^t}{\|\mathbf{x}_t - \overline{\mathbf{x}}_t 1\|^2}$
    16  Update portfolio weights $\mathbf{b}_{t+1} = \mathbf{b}_t - \tau_t \left(\mathbf{x}_t - \overline{\mathbf{x}}_t 1\right), \overline{\mathbf{x}} = \frac{\mathbf{x}_t}{m}$
end
```

Our nonparametric DCEPS algorithm, therefore, uses ideas from nonlinear time-series prediction approach with a time-delay embedding technique to help us make accurate predictions of future stock returns. Because we are able to discover hidden structures in the reconstructed phase-space of the stock price time series, our prediction on future stock price movements is expected to generate superior growth in portfolio wealth when compared to existing PS algorithms. Similar to the Nearest Neighbour (NN) algorithm of Gyorfi et al. (2008) our algorithm is a pattern-driven learning-to-trade strategy that optimises the trading strategy by mining potentially similar patterns from historical market sequences in a higher dimensional space.

6.5.4 Discussion on Parameter Settings

The discussion so far in this chapter has focussed on explaining the various steps that should be taken by the portfolio manager in the design of the DCEPS. We showed that the DCEPS follows many steps, each of which involves different parameters or set of parameter. The DCEPS algorithm has four parameters that need to be fine tuned by the portfolio manager. Two of these parameters, the sensitivity parameter ε and the correlation coefficient are set to constant values as shown in Table 6.1. Of course, there are many ways to optimise these parameters and our discussion here should be indicative and not conclusive of what these parameters should be in practice. The second set of parameters includes the embedding dimension d and the window length k which are floating parameters which are set in the next section.

6.5.5 Mixture of Experts

Our proposed DCEPS algorithm has three main parameters that need to be fine tuned by the portfolio manager. These are the window length k and the embedding dimension d. We therefore mix the experts the usual way.

6.6 EMPIRICAL RESULTS

This section presents numerical results obtained by applying the DCEPS-based algorithm to six financial market datasets described in Chapter 2. As usual, the back-testing experiments in this section will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed on a daily basis.

6.6.1 Analysis of Cumulative Wealth

The first experiment evaluates the compounded wealth achieved by the DCEPS-based learning to trade algorithm including a 10-basis-points transaction cost over the entire sample period. Figure 6.3 shows the total cumulative wealth achieved by the DCEPS algorithm on the six market datasets. The market index is calculated as an equal-weighted Buy-and-Hold portfolio on all stock available for that particular market. Figure 6.3 demonstrates that the DCEPS-based PS algorithm achieves very good performance on old data (NYSE(O), NYSE(N)) as well as on more recent and previously untested datasets. For example, on the South African TOP40 index data set, the total wealth achieved by the DCEPS strategy impressively increased from \$1 to almost \$454 over the trading period (from January 2000 to October 2013). This wealth growth is much higher than the \$13.6 achieved by the market index over the same 13 years periods. During that period, the best stock generated \$87.8 and the best constantly rebalanced portfolio in hindsight only achieved a growth in wealth of \$115. We also notice that all other datasets display similar impressive growth in portfolio wealth. Figure 6.3 also shows that, from a start of \$1, the algorithm achieved a cumulative wealth of $\$4.17 \times 10^7$ in the NYSE(O), \$682 using the FTSE100 and \$2670 using the NASDAQ.

Figure 6.3: DCEPS Performance Comparison

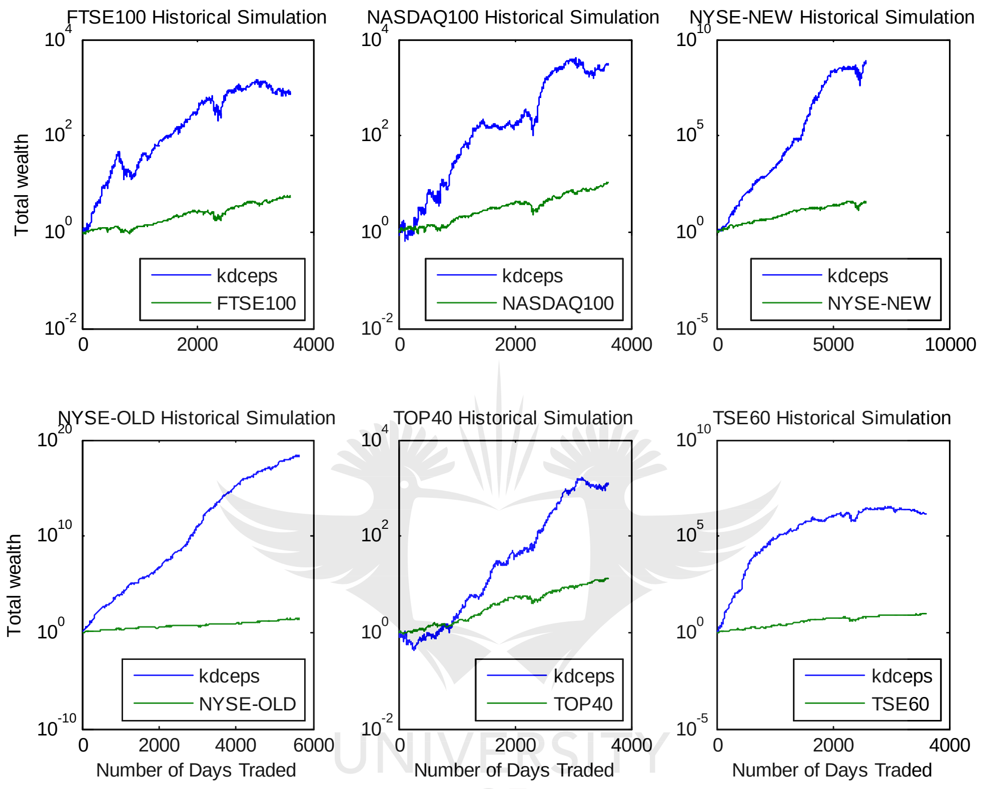


Table 6.2 presents some sub-periods' cumulative performance of the DCEPS algorithm against respective benchmarks for the data set considered. From these results it is easy to see that our proposed algorithm significantly outperforms the respective market indices on all datasets and for most sub periods. Over the full sample, the DCEPS algorithm displays a very impressive CAGR of around 64% and 168% using the TOP40 and TE60 datasets respectively. This compares very farourably with a CAGR of 20% and 17% for both market indices. Similar CAGR can be observed from all other datasets including the NASDAQ, the NYSE (O) and the NYSE (N).

Table 6.2: Cumulative returns over different trading periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.57	0.13	0.64	0.20	1.68	0.17	0.76	0.18	1.18	0.15	5.55	0.16
1Yr	0.05	0.21	0.01	0.17	-0.31	0.11	0.53	0.39	8.22	0.29	1.10	-0.01
2Yrs	-0.16	0.22	-0.12	0.22	-0.15	0.10	0.02	0.26	0.24	-0.04	1.17	0.16
3Yrs	-0.09	0.12	0.08	0.17	-0.19	0.05	0.03	0.18	0.15	-0.07	1.84	0.23
4Yrs	-0.03	0.16	0.44	0.19	-0.15	0.09	0.16	0.21	0.13	0.03	1.53	0.19
5Yrs	0.20	0.23	0.79	0.23	0.22	0.17	0.71	0.30	0.15	0.05	1.73	0.20
7Yrs	0.21	0.12	0.70	0.16	0.14	0.09	0.50	0.17	0.33	0.08	2.39	0.20
10Yrs	0.35	0.15	0.81	0.22	0.30	0.14	0.47	0.17	0.88	0.07	3.80	0.19
15Yrs	0.31	0.13	0.82	0.21	0.61	0.15	0.67	0.18	1.08	0.07	5.26	0.13

It is however noticeable that recent performance of the DCEPS algorithm has been less impressive as evidenced by the one-year, 2-year, 3-year and 4-year CAGR. The performance of the DCEPS-based PS algorithm has been pedestrian in recent times with a 1% gain in 2013 and a 20% gain in 2012 using the TSE60 index in Canada. The same pattern can be seen in the FTSE100 in the UK where the algorithm achieves an annualised compounded growth rate that was much less than its benchmark algorithm in 2013. There are many reasons that could be attributable to this recent poor performance. One of the reason could be that some market participants are increasingly employing mean reversion strategies resulting in a slow erosion of the DCEPS potential profit. Another reason could be due to the 2008 financial crisis and the ensuing price dislocation brought about by the significant sell off in asset prices. Such events could take years or decades to be completely absorbed in the financial market place.

6.6.2 Performance Risk Analysis

Table 6.3 shows some of the DCEPS risk measures on the six datasets considered. From the results, we see that the DCEPS-based strategy has a volatility of returns that is higher than the one achieved by its respective benchmarks. However, the Sharpe Ratios generated by the DCEPS-based algorithms are also significantly higher than the respective stock market indices. This is an indication that the DCEPS-based strategy generates much better risk adjusted-returns.

Table 6.3: Risk Statistics of the DCEPS-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.54	0.18	0.43	0.16	0.61	0.17	0.69	0.23	0.54	0.19	0.54	0.13
α	0.13	0.00	0.11	0.00	0.39	0.00	0.17	0.00	0.25	0.00	0.69	0.00
β	1.57	1.00	1.31	1.00	1.62	1.00	1.43	1.00	1.38	1.00	1.57	1.00
γ	2.00	0.00	2.35	0.00	-2.98	0.00	1.78	0.00	2.56	0.00	3.11	0.00
$\beta\uparrow$	1.63	1.00	1.45	1.00	1.55	1.00	1.53	1.00	1.54	1.00	1.65	1.00
$\beta\downarrow$	1.53	1.00	1.20	1.00	1.69	1.00	1.28	1.00	1.16	1.00	1.49	1.00
SR	0.98	0.37	1.20	0.78	1.79	0.59	1.05	0.53	1.57	0.44	3.60	0.64
MDD	-0.74	-0.40	-0.59	-0.30	-0.71	-0.41	-0.73	-0.47	-0.92	-0.64	-0.35	-0.37
PP	0.56	0.56	0.57	0.57	0.57	0.58	0.55	0.56	0.59	0.55	0.64	0.53
PP$\uparrow$	0.73	1.00	0.74	1.00	0.73	1.00	0.74	1.00	0.75	1.00	0.78	1.00
PP$\downarrow$	0.33	0.00	0.36	0.00	0.37	0.00	0.32	0.00	0.39	0.00	0.49	0.00

Further, we observe that maximum drawdown on the six stock datasets is much higher than those achieved by their respective benchmarks. The maximum drawdown is -74% for FTSE100, -59% for the TOP40, -71% for the TSE60 and -73% for the NASDAQ100. We also note that there is very little difference between the percentage of time the DCEPS-based algorithm and the respective market indices all generated positive returns (PP) returns. However, when the market index was up, the correspondong DCEPS-based algorithm generated positive returns more than 70% of time on all datasets. And when the market was down the DCEPS-based PS algorithm generated positive returns more than 30% of the time for all datasets. This suggests that the value add of the DCEPS-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market.

6.6.3 DCEPS Brokerage Costs Analysis

Because our portfolio is likely to handle a large number of daily transactions and that transactions are not without costs, we should therefore be concerned about the amount of commissions this portfolio will incur in actual market trading. The model of transaction we use is very similar to the one introduced in Chapter 3.

The graphs in Figure 6.4 depict the total returns of the DCEPS algorithm for varying proportional commission factor $c = 0.1\%, 0.2\%, ...$ .Figure 6.5 shows that the DCEPS investment algorithm can tolerate relatively small proportional commission rates and still beat competing benchmarks including the best stock, the equally weighted market index and best constantly rebalanced portfolio in hindsight called BCRP*

Figure 6.4: DCEPS Transaction Cost Analysis for Selected Markets

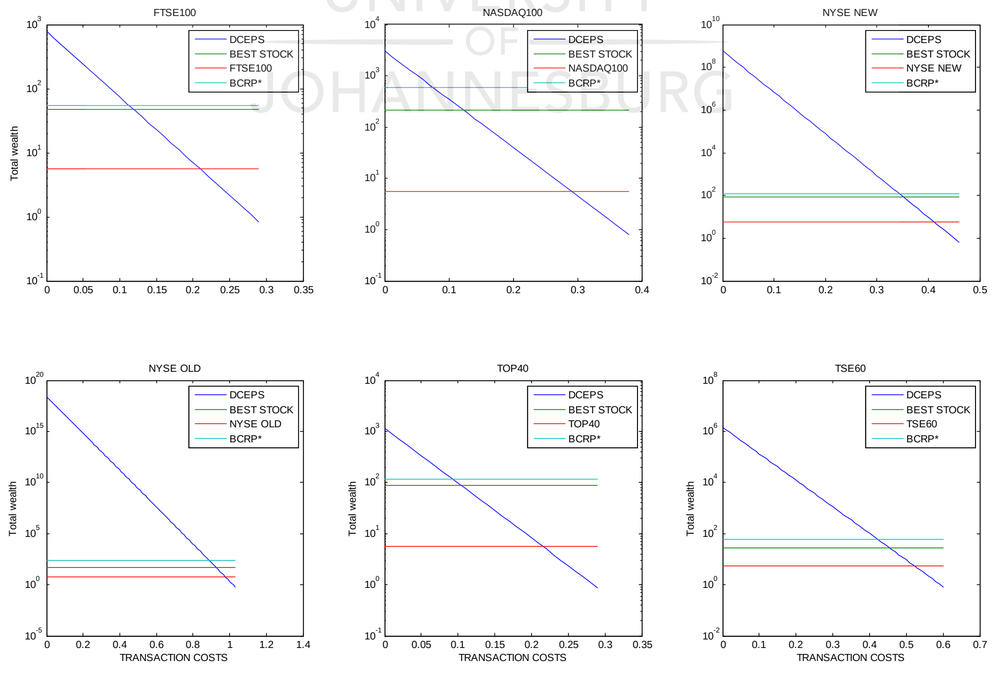


Figure 6.4 also shows the break-even commissions against the best stock, the market and the best constantly rebalanced portfolio in hindsight depends on the market considered. However, for all markets considered, our proposed DCEPS PS scheme performs very well even after adjusting for brokerage commissions. For example, with a commission cost of $c = 0.1\%$ or 10 basis points, the algorithm still beat the best stock, the market and the BCRP* portfolio in all markets we considered. In the case of TSE60 in Canada, our proposed algorithm still outperforms the best stock, the BCRP* and the overal uniform buy and hold at a commission cost of $c = 0.4\%$. In fact even with $c < 0.25\%$ our algorithm beats all its respective market indices.

In summary, the simulation results have demonstrated that the DCEPS-based PS algorithm achieves very impressive cumulative wealth growth for all datasets. These encouraging results show that the DCEPS-based PS algorithm is capable of achieving an excellent trade-off between return and risk. In our view the DCEPS-based PS algorithm is a very robust investment strategy.

6.7 CONCLUSION

In this chapter, we have presented a new Online approach to PS algorithm called DCEPS. Building on Takens(1981) delay embedding theorem, DCEPS combines powerful Online PS algorithms with ideas from signal processing and statistical learning to produce portfolios that substantially outperform their benchmarks equivalent on real-market datasets. This impressive performance generated by the DCEPS-based PS algorithm using data that are widely available again casts some doubt on the market efficiency hypothesis, at least on its weak form. However, we caution against extrapollating the performance of the DCEPS strategy given its recent poor performance. Despite outperforming most existing state-of-the-art PS algorithms on all data sets we believe that the recent algorithm poor performance is a concern to us.

7.0 MULTIVARIATE SINGULAR SPECTRUM ANALYSIS AND ONLINE PORTFOLIO SELECTION

7.1 INTRODUCTION

In 1983 Meese and Rogoff published a thought-provoking paper where they claimed that a simple random walk model could outperform both linear stochastic time series and structural econometric models in predicting the exchange rates. The main investment argument in their thesis (see Meese and Rogoff (1983)) was that securities markets are extremely efficient in reflecting market information and, therefore, no statistical model can enable an investor to achieve returns greater than those that could be obtained by simply buying and holding a randomly selected portfolio of individual asset classes, at least not without taking higher risk. This is the so called "Efficient Market Hypothesis (EMH)". In spite of the popularity of this EMH in academic circles, a considerable number of publications has now demonstrated that asset prices are to some extent predictable at least at the portfolio level.

One of the most popular tool used by investment managers in order to predict short term movements in stock prices is time series analysis. Although there are some notable exceptions, time-series models used for forecasting are often based on the restrictive assumptions of normality, stationarity and linearity. However, quite often financial and economic time-series data are non-Gaussian and may be generated by processes that are nonstationary and/or nonlinear; hence, methods that do not depend on these assumptions are likely to be useful for modelling and forecasting financial time series data. It is widely admitted in the econometric literature that nonlinearity is an intrinsic and fundamental feature of foreign exchanges rates and stock returns (see for instance Hsieh (1991), Ammermann and Patterson (2003), Beine et al. (2008)).

Moreover, most existing models are also essentially parametric and require the specification and estimation of models that are usually linear as the basis for analysis and forecasting.

More precisely, recent years have witnessed the emergence of a new strand of research in asset price forcasting driven by advances in machine-learning theory. Both theoretically well grounded algorithms, as well as algorithms based on simple heuristics, have been shown to exist in the field of Online learning for PS problems. Algorithms such as Universal Portfolio (Cover (1991)), Exponential Gradient (Hembold et al. (1998)) and Online Newton Step (Agarwal et al. (2006)) have not only demonstrated that Online portfolio rebalancing strategies can generate considerable wealth, but also that the wealth achieved by these universal algorithms is guaranteed a certain minimum given sufficient trading time. More recently, algorithms by Gyorfi et al. (2004,205,206) and Li et al. (2012a, 2012b, 2013) have shown very impressive performance with a growth in portfolio wealth that is significantly higher than what could be achieved by investing in the broader market index. What is even more impressive and could represent a major challenge for the EMH is that this growth in portfolio wealth was achieved using only publically available information in the form of stock price relative.

Despite the impressive performance generated by these new machine-learning algorithms, important unresolved issues remain. The main characteristics of asset prices is that they are made up of several components, including a slow component (trend) that influences asset price long-term behaviour, a periodical or oscillatory and a random component (noise) that influence its short-term dynamics. This complex structure of asset prices makes it very difficult for existing state-of-the-art portfolio allocation techniques to fully exploit the dependency structure as well as the nonlinearities found in stock prices. Our view is that successful nonlinear time-series modelling would improve forecasts and produce a richer notion of stock price cycle dynamics than linear time-series models allow. Therefore, we need a reliable statistical method to summarise and understand nonlinearities suitable for time series of stock prices. We therefore depart from stock prices statistical assumptions and methods used by most existing state-of-the-art PS algorithms by constructing models based on Singular Spectrum Analysis (SSA), which does not embody the assumptions of normality, stationarity and linearity and is, therefore, a good candidate for modelling and forecasting stock market returns data.

This chapter proposes a non-universal, nonparametric similarity-driven empirical PS algorithm based on the idea of multivariate singular spectrum analysis (MSSA).

Singular Spectrum Analysis (SSA) and its multivariate extension, MSSA, are nonlinear statistical model that have gained successful application in the various sciences such as meteorological, biomechanical, hydrological, physical sciences, engineering economics and finance (see Khan and Poskitt (2010)). We term our new algorithm "Multivariate Singular Spectrum Analysis for PS" or MSSAPS. MSSAPS not only exploits effective higher-dimentional statistical correlations between market windows via the delay-embedding feature, but also benefits from the exploration of powerful nonparametric machine-learning techniques. One of the striking feature of MSSAPS is that we make no assumtpion about the statistical properties of stock prices and the model itself requires very few paramters to finetune.

Our proposed MSSAPS algorithm makes an important contribution to the growing literature on Online PS strategies; the MSSAPS algorithm builds on Takens (1981) delay-coordinate-embedding theorem, which allows us to decompose stock prices time series into additive primary components, including trend, cycle and oscillatory components. Unlike existing state-of-the-art similarity-driven PS algorithms, the MSSAPS is derived from a time-delay embedding of multiple time series that allows us to capture potential nonlinear information found in complex dynamical systems of stock returns. Our technique creates a time-lagged version of a the original stock returns in order to discover hidden patterns normally not detected in a linear space. We find that the statistical evidence of out-of-sample predictability of stock returns is stronger in our model specification as compared to most state-of-the-art algorithms in the literature.

To demonstrate the robustness of our methodology we evaluate our MSSAPS algorithm against benchmark Online portfolio allocation techniques using six stock market datasets of which four were previously untested. On all these datasets and for all sub periods considered our algorithm substantially outperforms the existing state-of-the-art Online stock selection techniques.

The rest of the chapter is organised as follows. In Section 7.2, the mathematical model is described, and some related results are surveyed briefly. In Section 7.3, we present the spectral decomposition of time series data and their time-domain reconstruction. Multivariate singular spectrum analyis is described in Section 7.4.and its application to portfolio selection problems is presented in Section 7.5. Section 7.6 briefly describes our parameter choice and expert combination is presented in Section 7.7. Section 7.8 presents our empirical results adjust for brokerage and commissions. Some conclusions are drawn in Section 7.10.

7.1.1 Scope of Our Study

In Chapter 4 of this thesis, we surveyed several Online PS algorithms that have been proposed in the literature including Kelly (1956), Breiman (1961), Cover (1991), Ordentlich and Cover (1996), Helmbold et al. (1996), Borodin et al. (2000, 2004), Györfi et al. (2006,2008) and Li et al. (2012,2013). Building on the work of Li et al. (2012) we proposed a particularly interesting class of machine-learning algorithms for PS problems, the so called "Pattern-Matching" algorithms. We argued that similarity-driven (alos referred to as pattern-matching) empirical PS strategies aim to optimise the trading strategy by detecting potentially similar information from historical market sequences of price relative. A major advantage of these algorithms lies in their nonlinear structure because they possess the ability to identify both momentum and reversal pattern in historical stock prices.

However, a fundamental shortcoming of existing state-of-the-art similarity-driven PS algorithms in general is their failure to explicitely recognise that stock prices are made up of several components including a slow component (trend), a periodical or oscillatory and a random component (noise). The slow component mainly influences the long-run behaviour of stock prices while the oscillatory and random parts are those components that influence the short-run dynamics of the stock price time series. Earnings-growth policy, for example, focuses on the forces that influence long-run behaviour of stock prices whereas market cycle theory focuses on the forces that influence short-run dynamics. This complex structure of asset prices makes it very difficult for all existing state-of-the-art Online learning PS algorithms to fully exploit the dependency structure and the non-linear features found in stock prices. Successful application of non-linear time series modelling is more likely to improve forecasts and produce a richer notion of stock price cycle dynamics as compared to what linear time series models would allow. Therefore, there appears to be a need for a reliable statistical method that will understand and summarise the basic components found in time series of stock prices, hence the justification for the current research.

7.2 SPECTRAL DECOMPOSITION AND TIME-DOMAIN RECONSTRUCTION

Applying statistical-learning techniques to PS algorithms is now widely accepted by the academic community as well as by market practitioners. Tools like Fourier transforms, wavelet analysis or maximum entropy methods have been applied by traders with various degrees of success to model and predict the behaviour of stock prices. The classic methods of analysis such as Fourier analysis, regression analysis, or wavelet analysis tend to decompose the initial function into a series using a fixed system of basic functions such as sines and cosines, which produce strong periodicity property. These classic methods have been less successful in modelling and forecasting stock prices as they fail to explicitely account for nonlinear behaviour of stock price dynamical systems.

In recent years a novel non parametric technique known as Singular Spectrum Analysis (SSA) has been developed in the field of time-series analysis. SSA incorporates elements of classical time-series analysis, multivariate statistics, multivariate geometry, dynamical systems and signal processing (Golyandina et al, 2001) with the goal of looking for nonlinear, non-stationary, and intermittent or transient behaviour in an observed time series. This method has gained successful application in scientific fields such as meteorology, biomechanics, hydrology, physical sciences, economics, finance and engineering (see Khan and Poskitt (2010)).

7.2.1 Literature Review of Singular Spectrum Analysis (SSA)

SSA was independently developped by Broomhead and King (1986) and Fraedrich (1986). Broomhead and King (1986) applied SSA to the problems of dynamical systems theory and laid the mathematical basis used for singular spectrum analysis by combining singular value decomposition (SVD) and embedding theorems. Fraedrich (1986) used observed weather and climate variables to provide information for descriptions of the properties of the attractors of these dynamical systems and to obtain an estimate of the smallest number of variables necessary to explain the system dynamics. Ormerod and Campbell (1997) investigated the applicability of SSA to economic time series. A thorough description of the theoretical and practical foundations of the SSA method, with many examples, can be found in Golyandina et al. (2001). For a comparison between SSA and other techniques for forecasting time series, see for example Hassani (2007) and Hassani et al. (2009, 2010a). It has been shown that the results obtained by the SSA method are more accurate than those obtained by ARIMA and GARCH models (Hassani et al., 2009). For a wide variety of applications across different types of economics and financial time series see Hassani and Thomakos (2010). The SSA technique has also been used for filtering financial data and stock market data in Hassani et al. (2010b). Despite this growing popularity of SSA and its multivariate extension, there has been surprisingly little research on the application of these advanced mathematical statistics tools in Online learning for PS. The present chapter fills that void.

The basic idea behind SSA as applied in this chapter is that it is a tool that embeds single or multiple time series into a higher dimensional matrix, which is then decomposed into a set of base functions or constituent components. The application of SSA consists of four basic steps.

7.2.2 Embedding

The first step in the SSA is the embedding of the time series. This step consists of organising the scalar time series entries in a trajectory matrix.

Let $\mathbf{x} = (x_1, x_2, ..., x_N)$ be a time dependent signal of stock prices, where N is the number of samples of the data. From this one-dimentional time series, we first create a multidimensional space by using the process of embedding (Broomhead and King, (1986)). This is easily accomplished by decomposing the time series in a sequence of lagged copies of itself. Let L be the length for these lagged vectors such that $1 < L < N$, L is called the embedding dimension (Elsner and Tsonis, (1996)) and the number of lagged vectors will depend on the embedding dimension as $K = N - L + 1$. The matrix that is built from the organisation of the lagged vectors is called the trajectory matrix. The resulting trajectory matrix $\mathbf{X}$ is topologically equivalent to the original system:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdot\ \cdot\ \cdot & x_K \\ x_2 & x_3 & \cdot\ \cdot\ \cdot & x_{K+1} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_L & x_{L+1} & \cdot\ \cdot\ \cdot & x_N \end{bmatrix} \tag{7.1}$$

The main characteristics of this matrix are that the anti-diagonals of the matrix present the same values and are symmetrical around the main diagonal; that is, $\mathbf{X}_{ij}= \mathbf{x}_{i+j+1}$, where $1 \leq i \leq K$ and $1 \leq j \leq L$. The behaviour of this trajectory matrix is therefore that of the so-called Hankel matrix.

7.2.3 Singular Value Decomposition

One of the major implications of the Takens (1981) delay-embedding theorem is that geometrical invariants such as the eigenvalues of fixed and periodic points, generalised dimensions, Lyapunov exponents, and other topological features of the original state space are preserved in the reconstructed space. Thus, one can study the dynamical behaviour in the reconstructed state space instead of the true state space.

The second step is the decomposition of the Hankel matrix (Equation 7.6) in its singular spectrum by using Singular Value Decomposition (SVD). SVD is a decomposition of the form:

$$\mathbf{X} = \sum_{i=1}^{r} \sqrt{\lambda_i} u_i v_i. \tag{7.2}$$

where λ_i is the eigen values of $\mathbf{X}\mathbf{X}^\top$, r is the rank of $\mathbf{X}$ and u_i and v_i are the i^{th} eigenvector of $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$. In general, $\sigma_i = \sqrt{\lambda_i}$p i is called the singular value of the matrix $\mathbf{X}$.

Expression 7.7 can be converted to matrix notation as:

$$\mathbf{X} = \mathbf{U\Sigma V}^{\top} \tag{7.3}$$

where Σ is the diagonal matrix containing all the singular (or eigen) values in descending order and **U** and **V** are the matrices containing the set of orthonormal vectors (eigen vectors) u_i and v_i respectively. Given that the eigenvectors of **X** arise from the autocorrelation matrix $\mathbf{XX}^{\top}$, the components that present the most coherence in the data will be weighted by singular values with higher values. This way, the decomposition of the trajectory matrix in its singular spectrum is very useful for identifying trends in the data. Because the signal in the time series is correlated with its own lagged copies, it will be represented by the largest eigen values and, therefore, eigen values with less weight could be identified as noise.

7.2.4 Rank Reduction and Grouping

The third step is the application of a rank reduction to the trajectory matrix by recovering fewer amounts of singular values from the decomposition. In analyzing the dynamical components of the time series, different singular values can be grouped to recover physical behaviours identified in the decomposition. For noise reduction, the rank that represents most of the signal has to be identified before the rank reduction step. In general, the process consists of recovering a small subset of singular values compared to the full rank of the trajectory matrix. Let k be the desired rank for the trajectory matrix. This can be obtained by doing:

$$\mathbf{X}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^{\top} \tag{7.4}$$

where $\mathbf{X}_k$ is the recovered rank-reduced trajectory matrix. The recovered matrix is $rank = k$ and presents the lowest possible Frobenius norm.

7.2.5 Diagonal averaging

Finally, the time series is recovered from the rank-reduced trajectory matrix. The process of recovering the time signal $\widetilde{\mathbf{X}}_k$ is easily achieved by averaging in the anti-diagonals of $\mathbf{X}_k$.

Golyandina et al. (2001) introduced an operator that is helpful in describing the diagonal averaging of the recovered matrix. The operator works as follows: let $i + j - 1 = n.$ and $N = L + K - 1$, then the element n of $\mathbf{x}_k$ is

$$\widetilde{\mathbf{X}}_k(n) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \mathbf{M}_k(l, n-l-1) & for \quad 1 \leq n \leq L \\ \frac{1}{L}\sum_{i=1}^{L} \mathbf{M}_k(l, n-l-1) & for \quad L+1 \leq n \leq K \\ \frac{1}{K+L-1}\sum_{i=n-K+1}^{L} \mathbf{M}_k(l, n-l-1) & for \quad K+1 \leq n \leq N \end{cases} \tag{7.5}$$

This operation retrieves the component of the initial time series that was recovered after the rank reduction of the trajectory matrix. It is especially this ability of SSA to distinguish noise components from those of trend signals that is of great interest to us, as that can be applied in the filtering of data, which is desirable for a great number of reasons such as data presentation, modelling and so forth. As such, the purpose of SSA is not inherently to identify or build any particular model of the time series investigated. It is rather to provide information on the deterministic and stochastic parts of behaviour in the data, even when the time series is short and noisy (Ormerod and Campbell, 1997) All these properties are very desirable in the analysis of time series data obtained specifically from process plants, therefore justifying the application of SSA in this research. Table 7.1 presents the pseudo code for the SSA implementation.

Table 7 1: SSA (y_T, L)

```
Inputs   $y_T = (y_1, ..., y_T)$ : single time series of stock prices
         $L$ : the window length
Output   $RCs$ : Reconstructed Components of $y_T$
         Initialise the parameter $K = T - L + 1$
  Step 1 Compute the Trajectory Matrix
         Transfer $y_T$ into a multi-dimentional matrix $X_1, ..., X_K$
         where $X_i = (y_i, ..., y_{i-L+1})^\top$ and $X = [X_1, ..., X_K]$
  Step 2 Compute the Matrix $C = XX^\top$
  Step 3 Perform a Singular Value Decomposition of $C = XX^\top$
         and find $P$ and $\Lambda$ such that $XX^\top = P\Lambda P^\top$
         where $\Lambda = diag(\lambda_1, ..., \lambda_L)$ with $\lambda_1 \geq \lambda_2 \geq, ..., \geq \lambda_L$
         and $P = (P_1, ..., P_L)$ is the corresponding eigen vectors
  Step 4 Select a group $l \ (1 \leq l \leq L)$ of eigen vectors
         If $I = \{i_1, i_{2,}, ..., i_l\}$ then $X_I = \{X_{i1}, ..., X_{iL}\}$
  Step 5 Reconstruction of the one-dimentional time series
         Compute the matrix $\widetilde{\mathbf{X}} = \left\| \widetilde{X}_{i,j} \right\| = \sum_{k=1}^{l} P_{ik} P_{ik}^\top X$
         Average over the diagonals of $\widetilde{\mathbf{X}}$
end
```

7.3 MULTI-CHANNEL SINGULAR SPECTRUM ANALYSIS

Our main motivation for using multiple SSA (MSSA) comes from the fact that it is a nonparametric technique that works with arbitrary statistical processes, whether linear or nonlinear, stationary or non-stationary, Gaussian or non-Gaussian. Given that the dynamics of stock prices has usually gone through structural changes during the time period under consideration, we need to make certain that the method of prediction is not sensitive to the dynamical variations. Moreover, contrary to the traditional methods of time series forecasting like autoregressive or structural models that assume normality and stationarity of the series, MSSA method is non-parametric and makes no prior assumptions about the data.

MSSA combines two useful approaches to statistical analysis: (1) it determines major directions in the system's phase space that are spanned by the multivariate time series; and (2) it extracts major spectral components by using data-adaptive filters. In particular, MSSA can separate distinct spectral components in a multivariate data set of limited length and in the presence of relatively high noise levels. Ghil et al. (2002) provide an overview and a comprehensive set of references to both the theory and applications of MSSA.

7.3.1 Mathematical Formulation of the MSSA

Let $\mathbf{x} = \{x_d(n) : d = 1, 2, ..., D, n = 1, 2, ..., N\}$ be a multivariate time series of stock prices with D channels of length N. We assume that each stock price has been normalised so that the starting value is 100. Following the original embedding ideas of Takens (1981), the starting point of MSSA is to embed each stock price into an M-dimensional phase space by using lagged copies $\mathbf{X}_d(n) = [\mathbf{x}_d(n), ..., \mathbf{x}_d(n + M - 1)]$, $n = 1, 2, ..., N - M + 1$. From this we form the full augmented trajectory matrix $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_D)$ which has $D \times M$ columns of length $N - M + 1$ The MSSA algorithm then computes the covariance matrix $C = \mathbf{X}^\top \mathbf{X}/N$ of X and its eigen decomposition. Next, we diagonalise the appropriately symmetrised covariance matrix $\mathbf{\Lambda} = \mathbf{E}^\top \mathbf{C} \mathbf{E}$ to yield a diagonal matrix $\mathbf{\Lambda}$ that contains the real eigenvalues λ_k of C and a matrix E whose columns are the associated eigenvectors e_k. The eigen vectors form a new orthogonal basis in the embedding space of $\mathbf{X}$, and the corresponding eigen values give the variance in the direction of e_k. The spectral decomposition determines the directions of greatest variance successively, from largest to smallest, subject to the condition that each new direction be orthogonal to all the preceding ones.

Projecting the time series $\mathbf{X}$ onto the eigenvectors, $\mathbf{A} = \mathbf{X}\mathbf{E}$ yields the corresponding principal components (PCs) as the columns of $\mathbf{A}$. The PCs have the same length $N - M + 1$ as $\mathbf{X}$ and are uncorrelated at zero lag; they can be considered as filtered components of the time series $\mathbf{x}$, with data-adaptive filters that are given by the eigenvectors. This filtering property becomes more obvious when one looks at the following equation

$$a_k(n) = \sum_{d=1}^{D} \sum_{m=1}^{M} x_d(n+m-1)\, e_{dk}(m) \tag{7.6}$$

with $k = 1, 2, ..., D \times M$, $n = 1, 2, ...N - M + 1$ and a_k are the columns of matrix $\mathbf{A}$.

Our MSSA, however, goes beyond a pure identification and quantification of coupled behavior. We also allow to reconstruct the dynamical behaviour that the coupled subsystems share and that is associated with a specific eigenvalue-eigenvector pairs. The PCs, which are already filtered versions of the original time series, are not appropriate for this purpose, since they combine the properties of all the channels in the data set. Moreover, the PCs have a reduced length $N - M + 1$ and do not allow a unique localisation in time. A way to reconstruct the individual components of the system's behavior that is optimal in the least-squares sense is given by the transformation

$$r_{dk}(n) = \frac{1}{M_n} \sum_{m=L_n}^{U_n} a_k(n-m+1)\, e_{dk}(m) \tag{7.7}$$

The r_{dk} are referred to as reconstructed components (RCs) and represent that part of channel x_d that corresponds to the eigenelement pair (λ_k, e_k). The values of the normalisation factor M_n and the summation bounds L_n and U_n for the central part of the time series, $M \leq n \leq N - M + 1$, are simply $(M_n, L_n, U_n) = (M, 1, M)$. In particular, the time series can be completely reconstructed by the sum of all its RCs:

$$x_d(n) = \sum_{k=1}^{D \times M} r_{dk}(n) \tag{7.8}$$

The major advantage that MSSA holds is, therefore, the decomposition of the time series into the various components that constitute the basis of the time series. These components can be investigated in turn to identify major trends in the data, remove components that can be classified as pure noise and extract oscillatory components present in the data. Because MSSA takes cross-correlations into account, the individual RCs of the different time series are connected, as they represent the same spectral part.

One of the major disadvantage of MSSA is its computational complexity. In general, using MSSA helps one to turn a small data set into a huge amount of data. For example, performing MSSA on a 100×100 matrix $\mathbf{X}$ with lag $\mathbf{L} = 10$ results in a 1000×1000 matrix of eigenvector, 1000 eigenvalues, a 100×1000 matrix of principal components, and a 100×100000 matrix of reconstructed components. These high computational requirements are, of course, unsuitable for Online PS algorithms where the sise of the problem could be multiples of that shown here. To reduce this enormous amount of output, and to make larger problems computationally feasible, we propose a variant of the traditional MSSA algorithm. In our proposed MSSA strategy we sequentially decompose each individual stock price using the singular spectrum analysis discussed earlier, with a fixed lag for all our securities. This step allows us to separate the long-term trend of each stock price from the short-term dynamics. Of course the implicit assumption of our multiple singular spectrum analysis is that the stock pricres cross-correlations be set to zero. Because our focus is on short term portfolio selecton algorithms, we believe this simplification will not materially affect the final results of our analysis will make very large problems in Online PS computationally feasible. We further derive an equivalent Trend Adjusted Price Relative (TAPR) that has the major advantage of putting more emphasis on those components that matter the most for our stock price forecasts. Our new setting clearly represents a major departure from existing state-of-the-art machine-learning algorithms whose main input to the algorithms is the raw price data including its redundant long-term trend. Table 7.2 shows a pseudo code for our multi-channel singular spectrum analysis.

Table 7.2: MSSA $(\mathbf{Y}_T, L)$ pseudo code

```
Inputs    $\mathbf{y}_T = \left(\mathbf{y}_T^1, ..., \mathbf{y}_T^D\right)$ : multiple time series of stock prices
          $L$ : the window length
Output    $\mathbf{RC} \in R^{T\times D}$ : Reconstructed Components of $\mathbf{y}_T$
          Initialise the parameter $K \Longleftarrow T - L + 1$
for       $s = 1, 2, ...\ to\ D$
      3   for $t = 1\ to\ T$
      4       $\mathbf{RC}_t^s \Longleftarrow SSA\left(y_t^s, L\right)$
      5       $\mathbf{RC}\left(t, s\right) \Longleftarrow$ last value of $RC_t^s$
      6   end
end
```

7.4 MULTIVARIATE SINGULAR SPECTRUM FOR PS (MSSAPS)

This section details the application of the multivariate singular specturm analyis to PS (MSSAPS).

The MSSAPS algorithm follows the general construct of all Online learning for PS algorithms. To derive the portfolio weight vector at time t the portfolio manager makes a prediction about the future evolution of the market sequence and the appropriate portfolio weight vect $\mathbf{b}_t$. As usual, this prediction is based only on past information, including past stock price sequences. After this initial prediction the market reveals the vector of price relative vector $\mathbf{x}_t \in \mathbf{R}^d$ from which the portfolio incurs a period return of $\mathbf{b}_t^T\mathbf{x}_t$ which could be a profit $(\mathbf{b}_t^T\mathbf{x}_t \geq 0)$ when or a loss $(\mathbf{b}_t^T\mathbf{x}_t < 0)$. The portfolio manager then updates his or her portfolio cumulative returns according to $\mathbf{S}_t = \mathbf{S}_{t-1}\left(\mathbf{b}_t^T\mathbf{x}_t\right)$ and makes a new prediction before proceeding to the next round. Of course the objective of the portfolio manager is to maximise his or her total wealth in the long run, without any assumed knowledge of the statistical properties of the stock prices. This typical problem specification has its origins in the perceptron algorithm (Rosenblatt 1958) and has now been studied extensively in the machine-learning community where some Online algorithms have been shown to achieve very robust performance in historical simulations.

In order to make a prediction regarding the future evolution of the market sequence and the resulting portfolio weight vector **b** the MSSAPS algorithm follows three steps that allow us to construct asymptotically optimal investment strategies in the financial market.

The first step consists of decomposing the time series of stock prices into the various components including trend and oscillatory components using multivariate singular spectrum analysis. The main purpose here is to derive a measure of trend adjusted price relative, also called "detrended price relative", that enables more effective pattern identification in the whole historical evolution of the price series.

In the second step the MSSAPS searches the entire history of stock prices for matching (similar) features with the most recent window of detrendend price relative.

The last step of our algorithm designs a fixed portfolio vector that optimises the return for the trading periods following each matching instance. More particularly, MSSAPS seeks to locate the market windows that are similar to the most recent market window via a generalised correlation coefficient metric, and, thereafter, constructs a log-optimum portfolio according to the idea of the best constant rebalanced portfolio (Cover 1991) strategy. The specific optimisation routine we employed here is an adaptation of the idea of the passive aggressive mean reversion strategy of Li et al. (2012). The basic steps followed by our proposed nonparametric Online PS algorithms are detailed below.

7.4.1 Pattern Formation

Our proposed MSSAPS algorithm starts by creating a time-lagged version of the original individual stock price relative in order to discover hidden patterns normally not detected in a linear space. This approach provides the basis for our data-mining-based stock selection process. As argued in the previous chapter, the theoretical justification for this new complex nonlinear PS system is based on the Takens (1981) delay-embedding theorem. In fact, Takens (1981) proved that if the dimension of the embedding space is large enough, then the RPS is topologically equivalent to the original state space that generated the time series. Therefore, characterisations and predictions based on the RPS are considered valid and similar to those made if the original state space had been available.

Our first task in the pattern formation is to decompose stock price series using the MSSA algorithm proposed in Table 7.2. To this aim we set the embedding dimension $d = 5$ and the embedding lag $\tau = 1$. Alternative values for these two parameters do not seem to impact by much on the overall performance of our algorithm. At time t we call $\mathbf{x}_{t-k}^{t-1} \in \mathbb{R}^{k \times m}$ the most recent window of TAPR for a given window length k $(1 \leq k \leq 10)$.

Now let us consider any other discretised return vectors, $\mathbf{x}_{i-k}^{i-1}$, in the whole history of the market sequence whose correlation with the most recent window return vectors, $\mathbf{x}_{t-k}^{t-1}$, is larger or equal to ρ. Such time instant:

$$N_i = \left\{ k < i < t; correl\left(\mathbf{x}_{i-k}^{i-1}, \mathbf{x}_{t-k}^{t-1}\right) \geq \rho \mid 1 \leq k \leq 10,\ -1 \leq \rho \leq 1 \right\}$$

is called matching time. If $N_i == \varnothing$,we simply set $b = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$.

In order to search for matching patterns with the most recent window, we use a multi-dimentional version of the pearson correlation measure as a measure of similarity (Section 6.5.2). Given two vectors $\mathbf{X}$ and $\mathbf{Y}$ of dimension $k \times m$,our correlation coefficient is given by the following equation

$$correl\left(\mathbf{X}, \mathbf{Y}\right) = \frac{\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)\left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)}{\sqrt{\left(\sum_i \sum_j \left(\mathbf{X}_{ij} - \overline{\mathbf{X}}\right)^2\right)\left(\sum_i \sum_j \left(\mathbf{Y}_{ij} - \overline{\mathbf{Y}}\right)^2\right)}}$$

where $\overline{\mathbf{X}}$ and $\overline{\mathbf{Y}}$ represent the average of $\mathbf{X}$ and $\mathbf{Y}$ of dimension $k \times m$.

After locating these historical matches the MSSAPS algorithm constructs a fixed portfolio vector to optimise the returns for the trading periods following each matching.feature.

$$\mathbf{h}\left(\mathbf{x}_1^{t-1}\right) = \underset{b \in \Delta_d}{\arg\max} \prod_{k<i<t, d>0, \tau>0} \langle \mathbf{b}, \mathbf{X}_{N_i} \rangle \tag{7.9}$$

7.4.2 Portfolio Optimisation

To optimise the return for the trading periods following each matching sequence we use the routine proposed in Section 6.5.3. Table 7.3 displays the pseudo code of our implementation of the MSSAPS algorithm.

Table 7.3: Pseudo Code of the MSSAPS(P, k) Algorithm

```
Inputs    $P$ : stock price matrix
          $k$ $(1 \leq k \leq 10)$ : window
          $\varepsilon =:$ the sensitivity parameter
          $\rho =:$ correlation coefficient threshold
Output    $\mathbf{b}$ : expert's portfolio weights for the $t^{th}$ trading day
Step 1    $RC \Longleftarrow MSSA\,(P, L)$
Step 2    $\mathbf{x} \Longleftarrow \frac{\mathbf{P}}{\mathbf{RC}}$
Step 3    Reconstruct the state space of $\mathbf{x}_t$ using multivariate time delay embedding
for       $t = 1, 2, ..., n$ do
  1         if $t \leq w + 1$ then
  2             $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
  3         end
  4         for $i = w + 1$ to $t - 1$ do
  5             calculate $N_i = \underset{w+1 \leq i \leq t-1}{\mathbf{correl}} \left\{\mathbf{x}_{i-k}^{i-1}, \mathbf{x}_{t-k}^{t-1}\right\} \geq \rho$
  9              if $N_i == \varnothing$ then
 10             $\mathbf{b}_t = \left(\frac{1}{m}, \frac{1}{m}, ...\frac{1}{m}\right)$
 11         else
 12         Suffer a loss $\mathbf{l}_{\varepsilon}^{t} = \max\left(0, \mathbf{b}_t \mathbf{x}_t - \varepsilon\right)$
 13         Set parameter $\tau_t = \frac{l_{\varepsilon}^{t}}{\|\mathbf{x}_t - \overline{\mathbf{x}}_t 1\|^2}$
 14         Update portfolio weights $\mathbf{b}_{t+1} = \mathbf{b}_t - \tau_t\left(\mathbf{x}_t - \overline{\mathbf{x}}_t 1\right), \overline{\mathbf{x}} = \frac{\mathbf{x}_t}{m}$
end
```

Our nonparametric MSSAPS algorithm, therefore, uses ideas from nonlinear time-series prediction approach with a time-delay embedding technique in order to make accurate predictions of future stock returns. Because we are able to discover hidden structures in the reconstructed phase spaces of the stock price time series our prediction on future stock price movements is expected to generate superior growth in portfolio wealth when compared to existing PS algorithms. Similar to the Nearest Neighbour (NN) algorithm of Gyorfi et al. (2008) our algorithm is a pattern-driven learning-to-trade strategy that optimises the trading strategy by mining potentially similar patterns from historical market sequences in a higher dimensional space.

7.4.3 Discussion on Parameter Settings

The discussion so far in this chapter has focussed on explaining the various steps that should be taken by the portfolio manager in the design of the MSSAPS. From the previous sections we showed that the MSSAPS follows four steps, each of which involves different parameters or set of parameter. In this section we discuss some alternative parameters settings that could help the portfolio manager implement our new strategy. Of course, there are many ways to optimise these parameters and our discussion here should be indicative and not conclusive of what these parameters should be in practice.

The very first step in the implementation of our proposed nonparametric Online PS algorithms is the stock price decomposition phase using MSSA. The main objective of this step is to separate the long-term trend in stock prices from its short dynamics. Applying SSA to each time series of stock price requires an input of the embedding dimension d. Owing to the computational complexity of re-estimating the trend for all stock prices as new data become available we decided to set the embedding dimension as $d = 4$ for all stock prices and for all markets under consideration. Other embedding dimension values could generate better performance.

The second step in the implementation of the MSSAPS consists of searching the entire history of stock prices for matching features with the most recent window, k, of detrendend price relatives. In practice, the window sise has to be set in advance by the portfolio manager as different values of window (expert) k are likely to generate different terminal wealth growth. Setting an optimal window sise could be very complicated in practice as a window sise that has worked well in the past is not guaranteed similar performance in the future. A more robust approach is simply allocate, at time $t = 0$, the available capital amongst a fairly large numbers of experts indexed by the window sise ($1 \leq k \leq 10$ for example) and never rebalance after that. This is the preferred methodology of this chapter.

In the third step, we first discretise the most recent return vectors, $\mathbf{x}_{i-k}^{i-1}$, and search the whole history of the market sequence for instances where the correlation with the most recent window return vectors, $\mathbf{x}_{t-k}^{t-1}$, is larger than or equal to a pearson correlation coefficient ρ $(-1 \leq \rho \leq 1)$. As usual, different values of ρ will likely generate different paths in cumulative portfolio wealth. The search for highly correlated matching historical patterns (higher values of ρ) will generate a smaller number of matching instances, making the optimisation potentially sensitive to fewer inputs and, therefore, unreliable. A threshold value of ρ closer to -1 will generate too many matching instances with the risk that many of these might be spurious. In this chapter we decided to set the threshold value to any positive correlation between the most recent discretised window and any window of the same length in the entire history of TAPR ($\rho = 0.5$).

The fourth step of our algorithm designs a fixed portfolio vector that optimises the return for the trading periods following each matching instance according to the idea of the PAMR strategy of Li et al. (2012). The main parameter here is the mean reversion parameter, ε, of the PAMR algorithm. Important issues around this sensitivity parameter have been discussed elsewhere. See Li et al. (2012). Using similar arguments as in the case of the window sise, we choose to allocate, at time $t = 0$, the available capital amongst a fairly large numbers of experts indexed by a discretised mean reversion parameter ε ($0 \leq \varepsilon < 1$ for example) and never rebalance after that.

7.4.4 Mixture of Experts

Our proposed MSSAPS algorithm has therefore two main parameters that need to be fine tuned by the portfolio manager: the window length , k, and the mean reversion parameter ε. The experts are mixed in the usual way.

7.5 EMPIRICAL RESULTS

This section presents numerical results obtained by applying the MSSAPS-based algorithm to six financial market datasets as described in Chapter 2. As usual, the back-testing experiments in this section will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed on a daily basis.

7.5.1 Analysis of Cumulative Wealth

The first experiment evaluates the compounded wealth achieved by the MSSAPS-based learning-to-trade algorithm including a 10 basis points transaction cost over the entire sample period. Figure 7.1 summarises the total cumulative wealth achieved by the MSSAPS algorithm on the six market datasets. The market index is calculated as an equal weighted Buy-and-Hold portfolio on all stock available for that particular market.

Figure 7.1: MSSAPS Cumulative Performance Comparison

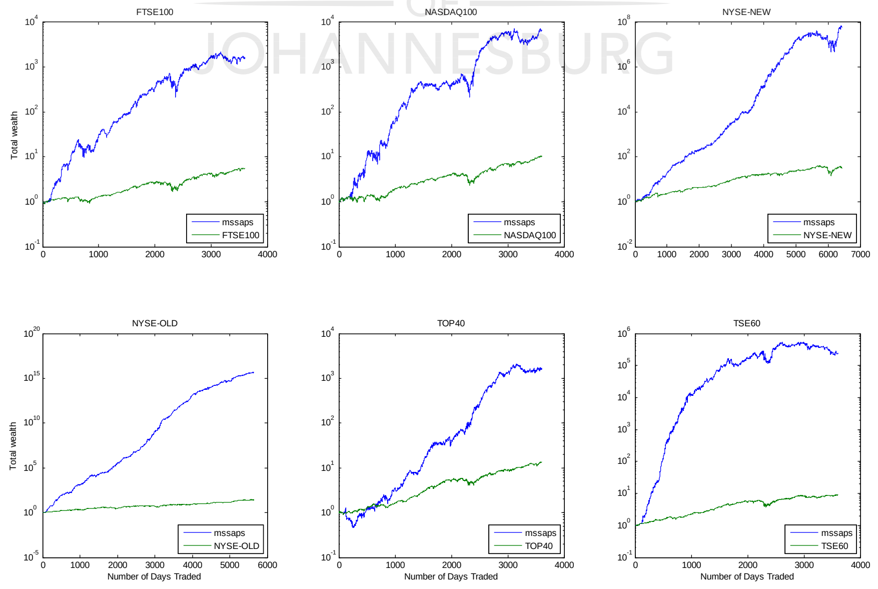


Figure 7.1 demonstrates that the MSSAPS-based PS algorithm achieves very good performance on old data (NYSE(O), NYSE(N)), as well as on more recent and previously untested datasets. For example, on the South African TOP40 index data set, the total wealth achieved by the MSSAPS strategy impressively increases from \$1 to almost \$1207. This wealth growth is much higher than the \$13.6 achieved by the market index over the same 15-year periods. During that period, the best stock generates \$87.8 and the best constantly rebalanced portfolio in hindsight achieves only a growth in wealth of \$115. In general, the performance of our algorithm on all datasets is very impressive when one compares it to the market index performance in the same period . The annualised percent yield (AY) depicts the same picture (Table 7.4). On an annualised basis, the MSSAPS algorithm has achieved at least three times the performance of its benchmark in the case of the NASDAQ in the US. This performance is also very impressive given that this is the index of the biggest 100 companies by market capitalisation listed on the NASDAQ. The same pattern can be seen in the TSE60 in Canada, where our methodology achieved an annualised compounded growth rate that is more than 10 times that achieves by its benchmark algorithm. It is worth noting that performance of the MSSAPS has been quite impressive of late as can be seen in the fourth row of Table 7.4. This one-year performance correspond to the period starting October 2012 to October 2013 for all the recent data sets. During that period the MSSAPS algorithm deliver 31% returns in the UK FTSE100 and 26% in the South African TOP40 index. This represents a significant improvement compared to existing state-of-the-art PS algorithm.

Table 7.4: CAGR over Selected Trading Periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.73	0.13	0.74	0.20	0.78	0.17	0.69	0.18	0.91	0.15	1.99	0.16
1Yr	0.31	0.21	0.26	0.17	-0.24	0.11	0.03	0.39	9.54	0.29	1.40	-0.01
2Yrs	0.30	0.22	0.27	0.22	-0.10	0.10	-0.04	0.26	0.13	-0.04	1.54	0.16
3Yrs	0.43	0.12	0.34	0.17	0.01	0.05	-0.01	0.18	0.06	-0.07	1.30	0.23
4Yrs	0.55	0.16	0.41	0.19	0.06	0.09	0.02	0.21	-0.01	0.03	0.97	0.19
5Yrs	0.84	0.23	0.61	0.23	0.36	0.17	0.35	0.30	-0.03	0.05	1.00	0.20
7Yrs	0.58	0.12	0.74	0.16	0.12	0.09	0.35	0.17	0.11	0.08	1.25	0.20
10Yrs	0.55	0.15	0.86	0.22	0.30	0.14	0.41	0.17	0.41	0.07	1.36	0.19
15Yrs	0.58	0.13	0.92	0.21	0.45	0.15	0.59	0.18	0.61	0.07	1.61	0.13

7.5.2 MSSAPS Performance Risk Analysis

Table 7.5 shows some of the MSSAPS risk measures on the six datasets considered. From the results, we see that the MSSAPS-based strategy has a volatility of returns that is higher than the one achieved by its respective benchmarks. However, the Sharpe Ratio generated by the MSSAPS-based algorithms are also significantly higher than the respective stock market indices. This is an indication that the MSSAPS-based strategy generates much better risk-adjusted returns.

Table 7.5: Risk Statistics of the MSSAPS-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.41	0.18	0.38	0.16	0.48	0.17	0.61	0.23	0.51	0.19	0.46	0.13
α	0.16	0.00	0.11	0.00	0.20	0.00	0.12	0.00	0.22	0.00	0.37	0.00
β	1.43	1.00	1.26	1.00	1.55	1.00	1.45	1.00	1.30	1.00	1.44	1.00
γ	0.77	0.00	3.59	0.00	-2.97	0.00	2.71	0.00	0.68	0.00	2.41	0.00
$\beta \uparrow$	1.48	1.00	1.36	1.00	1.49	1.00	1.62	1.00	1.35	1.00	1.46	1.00
$\beta \downarrow$	1.41	1.00	1.20	1.00	1.60	1.00	1.36	1.00	1.22	1.00	1.35	1.00
SR	1.36	0.37	1.46	0.78	1.28	0.59	1.04	0.53	1.39	0.44	2.45	0.64
MDD	-0.67	-0.40	-0.41	-0.30	-0.77	-0.41	-0.59	-0.47	-0.95	-0.64	-0.39	-0.37
PP	0.57	0.56	0.57	0.57	0.56	0.58	0.56	0.56	0.56	0.55	0.56	0.53
PP$\uparrow$	0.77	1.00	0.74	1.00	0.74	1.00	0.77	1.00	0.73	1.00	0.72	1.00
PP$\downarrow$	0.32	0.00	0.33	0.00	0.32	0.00	0.29	0.00	0.36	0.00	0.37	0.00

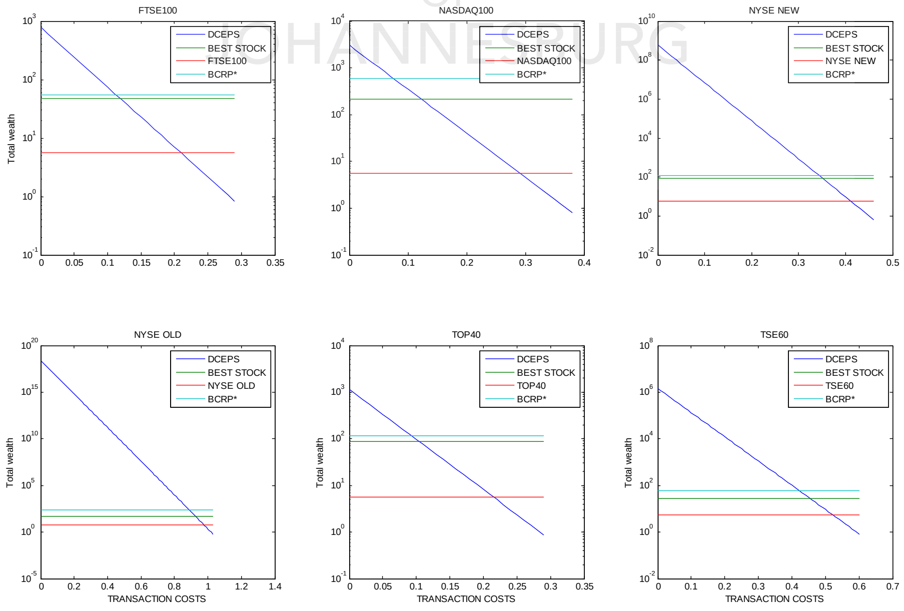
Further, we observe that maximum drawdown (row called MDD in Table 7.5) on the six stock datasets is.much higher than that achieved by their respective benchmarks. The maximum drawdown is -74% for FTSE100, -59% for the TOP40, -71% for the TSE60 and -73% for the NASDAQ100. We also note that there is very little difference between the percentage of time the MSSAPS-based algorithm and the respective market index generate positive returns (PP) returns. However, when the market index was up, the MSSAPS-based algorithm generated positive returns more than 70% of time on all datasets. When the market was down the MSSAPS-based PS algorithm generated positive returns more than 30% of the time on all datasets. These results suggest that the value add of the MSSAPS-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market.

In summary, the simulation results have demonstrated that the MSSAPS-based PS algorithm achieves very impressive cumulative wealth growth on all datasets. These encouraging results show that the MSSAPS-inspired PS algorithm is capable of achieving an excellent trade-off between return and risk. In our view the MSSAPS-based PS algorithm is a very robust investment strategy.

7.5.3 MSSAPS and Brokerage Costs Analysis

The MSSAPS investment algorithm can tolerate relatively small proportional commission rates and still beat competing benchmarks, including: the best stock,;the equally weighted market index; and the best constantly rebalanced portfolio in hindsight, called BCRP*. The graphs in Figure 7.2 depict the total returns of the MSSAPS algorithm for varying proportional commission factor $c = 0.1\%, 0.2\%, \ldots$ It does clearly appear that our strategy can withstand reasonable brokerage commissions. For example, with a commission cost of $c = 0.1\%$ or 10 basis points (10 bps), the algorithm still beat the best stock, the market and the BCRP* portfolio in all markets we considered. In fact even with $c < 0.25\%$ our algorithm beats all its respective market indices. As discussed in Chapter 3 some current online brokers charge very small proportional commissions. This means that a large investor can scale up the investment and suffer only a small proportional transaction rate, which can be as low as 0.05% for a round trip.

Figure 7.2: Transaction Cost Analysis for Selected Markets



7.6 CONCLUSION

In this chapter, we have presented a new Online approach to PS algorithm called MSSAPS. Building on Takens (1981) delay-embedding theorem, MSSAPS combines powerful Online PS algorithms with ideas from signal processing and statistical learning to produce portfolios that substantially outperform their benchmarks equivalent on real-market datasets even after accounting for modest but reasonable brokerage commissions.

8.0 DYNAMIC STOCK SELECTION AND MACHINE LEARNING ALGORITHMS FOR PORTFOLIO SELECTION

8.1 INTRODUCTION

The emergence of research in PS driven by advances in machine-learning theory is enabling investors to achieve returns greater than those that could be obtained by simply buying and holding a randomly selected portfolio of individual stock prices, even after accounting for trading costs. Both theoretically well grounded algorithms as well as algorithms based on simple heuristics have been shown to exist in the field of Online learning for PS problems. Algorithms such as Gyorfi et al. (2004, 2005, 2006), Borodin et al. (2006) and Li et al. (2012a, 2012b, 2013) have shown very impressive performance with a growth in portfolio wealth that, in most cases, is significantly higher than any that could have been achieved by investing in the broader market index using only publically available information in the form of stock price relative.

Despite this promising performance, existing state-of-the-art machine-learning PS algorithms still face some limitations that do not allow them to fully exploit the complex structure observed in stock prices. One of the major limitations is related to the fact that all existing state-of-the-art Online PS algorithms focus only on the PS problem, also referred to as the portfolio construction problem. Their exclusive focus on the portfolio weight vector ignores the basic fact that for an institutional portfolio manager, the stock selection problem is as important (if not more important than) as the portfolio construction problem. This chapter is primarily dedicated to the stock selection problem and we show a very flexible way to combine the stock selection and portfolio construction problems.

The importance of stock selection strategies cannot be overstated in financial markets.

Investment management companies employ a large number of fundamental analysts whose role is to look at the financial ratios of a company and determine the financial strength of its balance sheet and income statement, the potential market value of the company equity and finally the future potential of the company that will drive its stock price. Determining what the company is worth at today's market price enables the analyst to make a decision to either buy or sell equity stocks of the company. However, stock selection based on fundamental analysis is not the only tool available to professional money managers. There is now a large number of professional money managers who subscribe not only to the fundamental analysis paradigm but also to technical or quantitative analysis for which machine-learning techniques are gaining momentum.

This thesis has so far demonstrated that machine-learning algorithms that formulate the investment problem as a weight vector prediction problem can generate substantial growth in portfolio wealth, way above what a radomly generated portfolio could achieve. However, it is relatively easy to argue that the Online learning-inspired portfolio weight-vector prediction is only a partial representation of the portfolio management problem faced by professional investors. In actual applications, fund managers care as much about the portfolio construction problem as they care about their stock selection abilities. The objective of the present chapter is therefore two folds: first we propose a new nonparametric empirical stock selection model that builds on Support Vector Machines (SVM) and, second, we propose a new framework to recombine the stock selection and any pre-existing Online PS (interchangeably called "portfolio construction" or "portfolio optimisation") algorithm in one simple and flexible framework.

SVMs were originally developed by Vapnik (1998) (for a detailed introduction to the subject see Burges (1998) and Evgeniou et al. (2000)). The biggest difference between SVMs and other traditional methods of learning is that SVMs do not focus on an optimisation protocol that makes few in-sample errors as other techniques like classical regressions do. Traditionally, most learning algorithms have focused on minimising in-sample errors generated by the model. These methods are based on what is called "Empirical Risk Minimisation" (ERM). The goal of SVM is different. It does not seek to reduce the empirical risk of making just a few mistakes, but focuses on building reliable models that perform very well out-of-sample.

This principle is called "structural risk minimisation". In other words, the SVM searches a structural model that has little risk of making mistakes with future data.

A general limitation of SVM-based directional prediction is that the input variables lie in a high-dimensional feature space. Given that stock markets generally consist of several hundreds of stocks, this could lead to the high dimensionality of the variables to be trained. To deal with this problem, we first conduct a dimension reduction of our feature space in order to acquire an efficient and discriminative representation before proceeding to the classification step. A common unsupervised feature extraction method is Principal Component Analysis Pearson (1995) by which principal components are obtained through an eigen analysis of the covariance matrix of original data. The PCA has been widely used to deal with high dimensional datasets in many areas, such as protein dynamics reduction, spectral-data reduction, and pattern recognition.

In comparison with classical financial modelling, our approach allows continual adaptation to changing market conditions and a nonparametric solution-representation. One of the striking feature of our new algorithm is that we make no assumptions about the statistical properties of stock prices and the model itself requires very few paramters to fine tune. Compared with other state-of-the-art empirical PS approaches, the focus here is on a holistic design that integrates robust stock-selection procedures with portfolio construction assisted by machine-learning. A major aim of the chapter is, therefore, to develop methodologies for learning investment-decision models for portfolio management that can adapt with market processes, the application's performance and the environment.

To demonstrate the robustness of our methodology we evaluate our proposed algorithm against benchmark Online portfolio allocation techniques using six stock market datasets of which four were previously untested. We evaluate the methods developed in out-of-sample trading over historical data. The testing is designed to be realistic; for instance, we take into account factors such as transaction costs, stock mergers and data snooping issues. In all these datasets our algorithm substantially outperforms existing state-of-the-art Online stock-selection techniques-sometime in a spectacular fashion.

The rest of the chapter is organised as follows. After a quick literature review (Section 8.2), we present the mathematical model in Section 8.3. In Section 8.4, we present the general foundation of our online stock selection algorithm using SVM. In section 8.5 we present the main mathematical tool for understanding and designing SVMs. Section 8.6 presents our support vector machine for portfolio selection (SVMPS). Numerical results based on various datasets are described in Section 8.7. Section 8.8 concludes this chapter.

8.2 LITERATURE REVIEW

In recent years, many attempts have been made at predicting the direction of the broad market indices using a time series adaptations of the Support Vector Machine (Huang et al, (2005)) paradigm. Kim (2003), for example, showed that the SVM outperformed the artificial neural networks in predicting future direction of a stock market using an experiment with the Korean composite stock price index 200 (KOSPI 200). Huang et al (2005) reported remarkable performance with high hit ratios using a SVM-based model to predict Nihon Keisai Shimbun Index 225 (NIKKEI 225) in a single period. Since the SVM implements the structural risk minimisation principle, it often achieves better generalisation performance and lower risk of overfitting than the artificial neural networks (Cortes and Vapnik,(1995)).

Recently, Fan and Palaniswami, (2001) utilised a SVM algorithm to perform stock selection. In Tay and Cao (2003), SVMs are applied to the problem of forecasting several futures contracts from the Chicago Mercantile Market. Kim, (2003) uses SVMs to predict the direction of change in the daily Korean composite stock index. Their experimental results show that SVMs outperform other methods and that SVM should be considered as a promising methodology for financial time-series forecasting. In a similar way, Huang et al. (2005) investigated the predictability of financial movement direction with SVM by forecasting the weekly movement direction of NIKKEI 225 index. The methodology used by these authors demonstrated some promising results. Van Gestel et al. (2001) used an SVM for one-step-ahead prediction of the 90-day T-bill rate in secondary markets and the DAX 30. SVMs were used for regression purposes instead of classification, and the feature vector was based on lagged returns of the index, bond rates, S&P500, FTSE and CAC40.

Applying computational intelligence and machine-learning to stock selection is nothing new. Levin (1995), for example, applied artificial neural networks to select valuable stocks. Chu et al.(1996) used fuzzy multiple-attribute decision analysis to select stocks for portfolios. However, these approaches have some drawbacks in solving the stock-selection problem. For example, the fuzzy approach usually lacks learning ability, while the neural networks approach has an overfitting problem and is often easy to trap into local minima. In order to overcome these shortcomings, the SVM approach is used in this thesis for stock selection problems.

Perhaps, the work most related to the one presented in this chapter is the work of Fan and Palaniswami (2001) and Ruei-Shan (2008), where an SVM classifier is applied to the stock markets in Australia and Taiwan. The SVM outputs are used to rank the best long stocks in these markets. This is achieved by assigning the top 25% to the positive class and the remaining 75% to the negative class. These authors achieve significant excess returns in long-only equally weighted portfolios.

8.3 MATHEMATICAL MODEL

As usual, we consider a market of m securities that has the same characteristics as the market investigated by Gyorfi et al. (2006,2007,2008) and Algoet (1996). In this setting, the market vector $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^m)$ represents the vector of prices for $j = 1, 2, ..., m$ stocks. The change in security prices during the t^{th} trading period is represented as a stock market vector $\mathbf{x}_t$ $= (x_t^1, x_t^2, ..., x_t^m) \in \mathbf{R}_m^+$ where $\mathbf{x}_t$ is the vector m of non-negative numbers representing price relatives for the trading period t. The j^{th} component $x_t^j \geq 0$ of $\mathbf{x}_t$ expresses the ratio of two consecutive closing prices of asset j such that $x_t^j = \frac{p_t^j}{p_{t-1}^j}$. Thus an investment of d dollars in the j^{th} security just before the start of the t^{th} trading period yields dx_t^j dollars by the end of the t^{th} trading period

The investor in our model is allow to distribute his or her capital at the beginning of each trading period according to a portfolio vector $\mathbf{b}_t = (b_t^1, b_t^2, ..., b_t^m)$. Here the j^{th} component b_t^j of $\mathbf{b}_t$ denotes the proportion of the investor's capital invested in asset j at time t. Throughout this chapter we assume that the portfolio manager is not allowed any short sale of securities, meaning that the portfolio vector is such that $\mathbf{b}_t \geq 0$ and that the portfolio manager is always fully invested $\left(\sum_{j=1}^{m} b_t^j = 1\right)$, including reinvestment of dividends.

Define stock i individual contribution to the total portfolio returns as

$$c_t^i = b_t^i \cdot x_t^i \tag{8.1}$$

At time t, the overall contribution to the portfolio from all the stocks in our universe can be expressed as

$$\mathbf{c}_t = \mathbf{b}_t\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{x}_t \in \mathbb{R}^{1\times m} \tag{8.2}$$

where $\circ$ represents an entry-wise (also called "Hadamart") product

Let V be a m-dimensional subspace of $\{0,1\}^n$, such that the unitary vector $\overrightarrow{\mathbf{I}}_t = (1,1,...,1) \in V$. Standard linear algebra shows that it is possible to find a $(k-1)$-dimensional space W such that $\left\langle \overrightarrow{\mathbf{I}}_t, W \right\rangle = V$. However, this choice is not unique. Because $\overrightarrow{\mathbf{I}}_t$ is an all-one vector in unitary vector space and without loss of generality we can write $\mathbf{c}_t$ as

$$\mathbf{c}_t = \overrightarrow{\mathbf{I}}_t \circ \left(\mathbf{b}_t\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{x}_t\right) = \widetilde{\mathbf{b}}_t\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{x}_t \in \mathbb{R}^{1\times m} \tag{8.3}$$

In practice and for reasons that will become clearer later on, it helps to consider $\overrightarrow{\mathbf{I}}_t$ as a binary vector space in the subspace of subspace of $\{0,1\}^m$. In that case we can write Equation 8.3 as

$$\mathbf{c}_t = \widetilde{\mathbf{b}}_t\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{x}_t, \text{where } \widetilde{\mathbf{b}}_t = \frac{\mathbf{b}_t\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{I}_t}{\sum_{j=1}^{m} \mathbf{b}_t^j\left(\mathbf{x}_1^{t-1}\right) \circ \mathbf{I}_t^j} \text{ and } \left(\sum_{j=1}^{m} \widetilde{b}_t^j = 1\right) \tag{8.4}$$

As in Equation 8.3, Equation 8.4 represents the stock prices' individual contributions to the portfolio total returns where the vector $\widetilde{\mathbf{b}}_t$ represents the new normalized portfolio weight vector. $\widetilde{\mathbf{b}}_t$ itself is made up of two important components: the first one, $\mathbf{b}_t\left(\mathbf{x}_1^{t-1}\right)$, is as usual the portfolio vector derived from any Online PS algorithm and can be regarded as the result of the portfolio construction problem. The second element, $\mathbf{I}_t$, can be understood as the manager controlled variable that helps him or her decide what stocks could/shoul be included in the portfolio. In the case where $\mathbf{I}_t$ is a unitary vector, the manager is allowed to hold all stock in the universe selected provided that the weight vector, $\mathbf{b}_t$, allocated to those stocks is strictly greater than zero. It is therefore evident in this framework that the portfolio allocation through the weight vector $\mathbf{b}_t$ is no longer the sole criterion for investment decisions. Given stock j, the portfolio manager will take a position on stock j only when the portfolio construction as expressed in the weight vector b_t^j and the stock selection found in I_t^j all provide positive signals on stock j.

Let S_0 denote the investor's initial capital. A PS algorithm is a mechanistic procedure that produces any sequence of portfolios $\widetilde{\mathbf{b}}_n = \left(\widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2, ..., \widetilde{\mathbf{b}}_n\right)$ by specifying how to reinvest the current wealth from trading period to trading period. Starting with an initial wealth $\mathbf{S}_0$, after n trading periods, the investment strategy $\mathbf{B}$ achieves the wealth:

$$\mathbf{S}_n = \mathbf{S}_0 \prod_{i=1}^{n} \sum_{j=1}^{m} \widetilde{b}_t^j \left(\mathbf{x}_1^{t-1}\right) x_t^j = \ \mathbf{S}_0 \exp\left\{\sum_{i=1}^{n} \log \widetilde{\mathbf{b}}_t^{\intercal} \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t\right\} \tag{8.5}$$

where $\widetilde{\mathbf{b}}_t \left(\mathbf{x}_1^{t-1}\right) = \left[\overrightarrow{\mathbf{I}}_t \circ \mathbf{b}_t\right] \left(\mathbf{x}_1^{t-1}\right)$ denotes the portfolio vector chosen by the investor on the t^{th} trading period, upon observing the past behaviour of the market represented by $\mathbf{x}_1^{t-1}$. $\widetilde{\mathbf{b}}_1$ is a constant portfolio vector such that every asset receives the same initial investment; that is, $\widetilde{\mathbf{b}}_1 = \frac{1}{m}, ..., \frac{1}{m}$.

Equation 8.5 may be written as:

$$\mathbf{S}_n = \mathbf{S}_0 \exp\left\{n\mathbf{W}_n\left(\mathbf{B}\right)\right\} \tag{8.6}$$

where $\mathbf{W}_n\left(\mathbf{B}\right)$ denotes the average growth rate and is given by:

$$\mathbf{W}_n\left(\mathbf{B}\right) = \frac{1}{n} \sum_{i=1}^{n} \log \widetilde{\mathbf{b}}_t^{\intercal} \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t. \tag{8.7}$$

Given that the goal is to maximise $\mathbf{S}_n = \mathbf{S}_n\left(\mathbf{B}\right)$, Equation 8.7 is equivalent to maximising the following average growth rate:

$$\mathbf{W}_n\left(\mathbf{B}\right) : \widetilde{\mathbf{b}}_i^{*} \left(\mathbf{x}_1^{i-1}\right) = \arg_{\widetilde{b}} \max \mathbf{E}\left\{\log \left[\overrightarrow{\mathbf{I}}_t \circ \mathbf{b}_t\right]^{\intercal} \left(\mathbf{x}_1^{t-1}\right) \mathbf{x}_t \mid \mathbf{x}_1^{i-1}\right\} \tag{8.8}$$

Equation 8.8 shows the final optimisation problem as faced by investment managers. The portfolio manager in our specification has to solve two seemingly independent but ultimately inter-connected optimisation problems. More particularly, both the stock selection problem-that is finding optimal values for $\overrightarrow{\mathbf{I}}_t$ and the portfolion construction (finding the optimal value of $\mathbf{b}_t$)-must be simultaneously determined by the portfolio manager. In this specification, $\overrightarrow{\mathbf{I}}_t$ solves the stock selection problem at time t while $\mathbf{b}_t$ deals with the portfolio construction problem. Of course when $\overrightarrow{\mathbf{I}}_t$ is an unitary vector space, the problem in Equation 8.8 is a special case of to the standard Online learning portfolio optimisation problem which could be approximated using any of the algorithms already discussed in the preceeding chapters of this thesis.

We therefore conclude that the complex problem faced by money managers is far from being that of determing an optimal weighting scheme only, but also that of choosing what stocks should be bought or sold. Therefore, focussing exclusively on one part of the problem will fail to adequately resolve fund mangers anxieties.

Our proposed nonparametric Online PS algorithm follows two steps. In the first step, we offer an empirical framework to apply the SVM strategy and predict the cross-sectional directional movement of stock prices in our universe using the features estimated in the first step. The second step of our algorithm proposes a new recombination algorithm that demonstrates excellent historical performance. In this step, we combine the SVM technique with conventional Online learning portfolio selection models to form our final strategy portfolio. This combined strategy is compared with the benchmark historical average strategy so as to evaluate the predictive power of this proposed strategy.

8.4 ONLINE STOCK SELECTION ALGORITHM USING SUPPORT VECTOR MACHINES

In many supervised learning problems in general and SVM in particualar feature selection is criticially important. More precisely, selecting relevant features for support vector machine (SVM) classifiers is important for a variety of reasons, including generalization performance, computational efficiency, and constraints and interpretational issues. Traditionally stock trading strategies using SVM have used technical or fundamental company attributes to xtract features and learn SVM parameters independently. This chapter proposes a purely statistical method based on principal component analysis for feature extraction. Instead of pre-defining an arbitrary number of fundamental features to be included in a classifier we extract the features from a linear-beta multi-factor stock pricing model (see Section 8.4.1) for a given training set. After the feature set is determined, the SVM is trained once daily, in order to adjust to changing market conditions. Portfolios are formed by ranking stocks using the classifier output. The highest ranked stocks are used for long positions and the lowest ranked ones could be used for short sales.

8.4.1 A Linear-Beta Multi-Factor Stock-Pricing Model

Our support vector stock selection machine has its foundations in the so-called "linear beta multifactor pricing models". According to the linear-beta-pricing-models paradigm, few economy-wide factors are sufficient to represent the systematic risk that one faces by investing in affected securities. The logical conclusion is that the expected return on an asset is a linear function of its factor betas (Ross (1976), Connor (1984)). Although the linear-beta-pricing-model do not define what the economy-wide factors are, one example of factors could be the market portfolio as in the standard Capital Asset Pricing Model (CAPM) of Sharpe (1964) and Lintner (1965). Another example of factors is seen in the work of Fama and French (1993) who showed that size and book-to-market are sufficient to capture all economy-wide pervasive sources of risk. In this chapter we will identify the pervasive risk factors based on statistical analysis of historical return data using Principal Component Analysis, also called PCA (see Connor and Korajczyk (1988) and Lehmann and Modest (1988)). Our factor model aims to capture the returns of assets through a set of statistical factors or components, extracted purely from historical price data. These factors are the drivers behind asset returns and are responsible for the co-variation across different assets.

Let us now consider an economy with a large collection of assets prices. The econometrician has observations on the returns on m of the assets in the economy. Denote by

$R_t = [R_{1t}, R_{2t}, ..., R_{mt}]'$ the vector of gross total returns for m securities at time t. We also denote by $\Sigma_R = E\left[(R_t - E[R_t])(R_t - E[R_t])'\right]$ the covariance matrix of the return vector and by $f_t = [f_{1t}, f_{2t}, ..., f_{mt}]'$ the vector of time-t values taken by the K factors.

According to the linear factor model paradigm, the return-generating process for a security is driven by the presence of the various common factors and the security's unique sensitivities to each factor (factor loadings). Because the returns on the N securities are generated according to a linear factor model with the same K factors we can write mathematically:

$$R_{it} = \alpha_i + \sum_{j=1}^{m} \beta_{ij} f_{jt} + \mu_{it} = \alpha_i + f_t^\top \beta_i + \mu_{it} \tag{8.9}$$

α_i is the expected level of return for stock i if all factors have a value of zero, f_j represents the value of the j^{th} factor that impacts the return on stock i and β_{ij} stands for the beta (sensitivity) of stock $i's$ return to the j^{th} factor given by:

$$\beta_i = \Sigma_F^{-1} E\left[(R_{it} - E[R_{it}])(f_t - E[f_t])\right] \tag{8.10}$$

and Σ_F is the variance-covariance matrix of f_t given by

$$\Sigma_F = E\left[(f_t - E[f_t])(f_t - E[f_t])'\right] \tag{8.11}$$

μ_i a random error term with mean equal to zero and variance equal to $\sigma^2_{\varepsilon_i}$

For our purpose, statistical factors here will be extracted such that the factors are assumed to capture all systematic behaviour:

$$\begin{aligned} &E\left(f_i \mu_j\right) = 0 \text{ for all stocks and indexes} \\ &E\left(\mu_i \mu_j\right) = 0 \text{ for all } i \text{ and } j \text{ such that } i \neq j \end{aligned} \tag{8.12}$$

that is common factor and asset-specific returns are uncorrelated and the asset-specific returns across different assets are also uncorrelated.

Sharpe (1964) and Lintner (1965) developed the first linear beta pricing model, the CAPM. Merton (1973) derived the first linear-multi-beta pricing model by examining the intertemporal portfolio choice problem of a representative investor in continuous time. Long (1974) proposed a related multibeta pricing model in discrete time.

If an investor holds a well-diversified portfolio, residual risk will tend to go to zero and only systematic risk will matter. The only terms in the Equation 8.10 that affect the systematic risk in a portfolio are b_{ij}. Because the investor is assumed to be concerned with expected return and risk. Under these assumptions and following Connor (1984) the following proposition can easily be proven

Proposition 1. *Suppose that there are n assets whose rates of return are governed by $m < n$ factors according to the equation*

$$r_{it} = a_i + \sum_{j=1}^{m} b_{ij} f_j \quad , \qquad \textit{for} \;\; i = 1, 2, ..., n \tag{8.13}$$

Then there are constants $\lambda_0, \lambda_1, ..., \lambda_m$ such that

$$\mu_i = \lambda_0 + \sum_{j=1}^{m} b_{ij} \lambda_j \quad , \qquad \textit{for } i = 1, 2, ..., n \tag{8.14}$$

Proof. We recall that each random factor f_j is simply an investable portfolio of nominated assets that best represents the sector or component of randomness we want to model. In particular, we can construct n special portfolios $\{p_i\}_{i=1}^{n}$, where the strategy for portfolio p_i say, is invest b_{ij} in the portfolio representing factor j for $j = 1, ..., m$; and invest the remaining $1 - \sum_{j=1}^{m} b_{ij}$ at the risk free rate. Upon doing this we have effectively constructed n portfolios whose return rates are given by

$$r_{p_i} = \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0 + \sum_{j=1}^{m} b_{ij} f_j \quad , \qquad \text{for } i = 1, 2, ..., n \tag{8.15}$$

□

If we compare equation 8.14 with the model for the return rates of the individual portfolios in Equation 8.16 we can consider making a suitable combination of the special portfolio p_i and asset i and for it to be free of all risk. Before we make this combination we examine the risk-free components of the two. Specifically, if:

$$a_i < \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0 \tag{8.16}$$

then our strategy is to invest long one unit in p_i and invest short one unit in r_i. This strategy is cost-free and risk-free yet it provides a positive return

$$r_{p_i} - r_i = \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0 - a_i > 0. \tag{8.17}$$

This represents an arbitrage opportunity, which is not allowed. Hence, we conclude that Equation 8.17 cannot hold. In a similar manner we can argue that.

$$a_i > \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0 \tag{8.18}$$

cannot hold. Thus we have

$$a_i = \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0. \tag{8.19}$$

and thus,

$$r_i = \left(1 - \sum_{j=1}^{m} b_{ij}\right) r_0 + \sum_{j=1}^{m} b_{ij} f_j \quad = \quad r_0 + \sum_{j=1}^{m} b_{ij} \left(f_j - r_0\right) \quad \text{for } i = 1, 2, ..., n \ . \tag{8.20}$$

Taking expectations leads to the results:

$$\mu_i = \lambda_0 + \sum_{j=1}^{m} b_{ij} \lambda_j \quad , \qquad \text{for } i = 1, 2, ..., n \ , \tag{8.21}$$

where $\lambda_0 = r_0$ and $\lambda_j = \mu_{f_j} - r_0$, for $j = 1, 2, ..., m$.

It can be shown that these results hold true for the more general case where:

$$r_i = a_i + \sum_{j=1}^{m} b_{ij} f_j + \varepsilon_i \quad , \qquad \text{for } i = 1, 2, ..., n \tag{8.22}$$

To get a better understanding of the implications of this framework we will demonstrate the expected returns that must arise from the model with a two-index model . Suppose that the following two-index model describes returns: .

$$r_i = a_i + b_{i1} f_1 + b_{i2} f_2 + \varepsilon_i \tag{8.23}$$

with the usual condition $E(\varepsilon_i \varepsilon_j) = 0$ for all i and j such that $i \neq j$. As argued earlier if an investor holds a well-diversified portfolio, residual risk will tend to go to zero and only systematic risk will matter. The only terms in the Equation 8.23 that affect the systematic risk in a portfolio are b_{i1} and b_{i2}. Because the investor is assumed to be concerned with expected return and risk, he or she need be concerned with only three attributes of any portfolio (p): $\overline{R}_p$, b_{i1} and b_{i2}.

Let us hypothesise the existence of the three widely diversified portfolios shown in the following table:

Portfolio	**Expected Returns**	b_{i1}	b_{i2}
A	15	1	0.6
B	14	0.5	1
C	10	0.3	0.2

It is straightforward to see that the equation of the plane in $\overline{R}_p$, b_{i1} and b_{i2}. space defined by these three portfolios is:

$$\overline{R}_i = 7.75 + 5b_{i1} + 3.75b_{i2}$$

Given a vector of portfolio weights w, the expected return and risk measures of any portfolio of these three portfolios are given by:

$$\overline{R}_p = \sum_{i=1}^{3} w_i \overline{R}_i$$
$$b_{p1} = \sum_{i=1}^{3} w_i b_{i1}$$
$$b_{p2} = \sum_{i=1}^{3} w_i b_{i2}$$
$$\sum_{i=1}^{3} w_i = 1$$

Because a weighted combination of points on a plane (where the weights sum to one) also lies on the plane, all portfolios constructed from portfolios A, B, and C lie on the plane described by portfolios A, B, and C. What happens if we consider a new portfolio not on this plane? For example, assume a portfolio E exists with an expected return of 15%, a b_{i1}

of 0.6, and a b_{i2} of 0.6. Compare this with a portfolio (call it D) constructed by placing equal weighting on portfolio A, B and C. The sensitivities on this portfolio are:

$$b_{p1} = \tfrac{1}{3}(1) + \tfrac{1}{3}(0.5) + \tfrac{1}{3}(0.3) = 0.6$$
$$b_{p2} = \tfrac{1}{3}(0.6) + \tfrac{1}{3}(1) + \tfrac{1}{3}(0.2) = 0.6$$

The risk for portfolio D is identical to the risk on portfolio E. The expected return on portfolio D is:

$$\frac{1}{3}(15) + \frac{1}{3}(14) + \frac{1}{3}(10) = 13$$

Alternatively, because portfolio D must lie on the plane described earlier, we could have obtained its expected return from the equation of the plane:

$$\overline{R}_i = 7.75 + 5(0.6) + 3.75(0.6) = 13$$

By the law of one price, two portfolios that have the same risk cannot sell at different expected returns. In this situation it would pay arbitrageurs to step in and buy portfolio E while selling an equal amount of portfolio D short. Buying portfolio E and financing it by selling D short would guarantee a riskless profit with no investment and no risk. We can see this quite easily. Assume the investor sells \$100 worth of portfolio D short and buys \$100 worth of portfolio E.

The arbitrage portfolio involves zero investment, has no systematic risk (b_{i1} and b_{i2}), and earns \$2. Arbitrage would continue until portfolio E lies on the same plane as portfolios A, B, and C. Therefore, all investments and portfolios must be on a plane in expected return, (b_{i1} and b_{i2} for our simple example) space or hyperspace. If an investment were to lie ABOVE or BELOW the plane, an opportunity would exist for riskless arbitrage. The arbitrage would continue until all investments converged to a plane.

8.4.2 Eonometric Estimation of Linear-Beta Stock Pricing Models

The econometric methods that have been used to evaluate linear-beta pricing models (Equations 8.14 and 8.15) using historical return data on a large cross-section of stocks remains a highly debated issue. Jagannathan et al. (2010) provide a detailed analysis of the main econometric approaches available to the portfolio manager in estimating the main two equations of the linear-beta multifactor pricing model. In general, three approaches have been suggested in the literature for examining linear beta pricing models. These econometric methods include the cross-sectional regressions method, maximum-likelihood (ML) methods and generalised method of moments (GMM).

Fama and MacBeth (1973) developed the two-pass cross-sectional regression method to examine whether the relation between expected return and factor betas are linear. Betas (Equation 8.14) are estimated using time series regression in the first pass and the relation between returns and betas (Equation 8.15) are estimated using a second pass cross-sectional regression.

Gibbons (1982) showed that the classical maximum likelihood method can be used to estimate and test linear beta pricing models when stock returns are i.i.d and jointly normal. Kandel (1984) developed a straightforward computational procedure for implementing the maximum-likelihood method.

MacKinlay and Richardson (1991) show how to estimate the parameters of the CAPM by applying the GMM to its beta representation. These authors illustrate the bias in the tests based on standard maximum-likelihood methods when stock returns exhibit contemporaneous conditional heteroscedasticity and show that the GMM estimator and the maximum-likelihood method are equivalent under conditional homoscedasticity. An advantage of using the GMM is that it allows estimation of model parameters in a single pass, in this manner avoiding the error-in-variables problem.

As in Fama and MacBeth (1973), the current chapter proposes a two-pass cross-sectional analysis to examine predictability of future stock returns from historical returns and their factor betas. During the first-pass regression we estimate both the sensitivities (beta) and the latent factors using a PCA (see Equation 8.14). Of course any macroeconomic or fundamental factors could be used here to estimate the factor sensitivities using time series analysis for each individual stock. This first step therefore uses the factor loading as feature inputs in the second pass analysis. In the second pass we estimate the optimal separating hyperplane(Equation 8.15) using SVM.

8.4.3 Beta Estimation Via Principal Component Analysis

In most SVM applicatoins to financial market data, the feature space has in most cases been determined by firm-specific attributes (Tay and Cao, (2001), Yang et al., (2002) and Ince and Trafalis (2004)). Many of these firm-specific attributes might be related to an asset's risk, including market capitalisation (sise), dividend yield, growth, price-earnings ratio (P/E), and so on. However, these fundamental characteristics present many drawbacks: First, using accounting data in general generates rules that may differ significantly across firms. Second, reporting dates differ accross companies and constructing time-synchronised inter-firm comparisons is actually a difficult exercise. Third, there is no rigorous theory that provides guidance as to which traditional accounting variables should be related to an appropriate measure of risk for computing the risk-return trade-off. Even if historical empirical relationships can be uncovered, without the foundation of a rigorous theory, one must be concerned that any historical correlation might be spurious and subject to sudden and material change. Given these problems our approach is to use a statistical factor model to jointly estimate the principal components and the factor loading and use these estimates as inputs to our support vector stock selection model.

PCA is a vector space transformation used to reduce multidimensional datasets to lower dimensions. New variables are created as a linear combination of the original data. Each of the new variables (principal components) is constructed to be uncorrelated with all others. Each successive principal component explains a decreasing amount of variation in the datasets.

PCA is realised with a eigen decomposition on a covariance matrix or a correlation matrix of assets values (standardised). Consider a collection of n observations of m asset returns. Let $\mathbf{x}$ denote the resulting $n \times m$ data matrix, and assume without loss of generality that $\mathbf{x}$ has full column rank. PCA main goal is to find a linear combination of the observed asset returns that "explains" as much as possible of the observed variability of the data (See Theil (1971)). Let $\mathbf{P}$ denote this $n \times \ m$ matrix of the eigenvectors of $\mathbf{xx}^\top$ that correspond to the m non-zero eigenvalues (sorted in descending order) of $\mathbf{xx}^\top$. (Since $\mathbf{xx}^\top$ is positive semi-definite, exactly m of its eigenvalues are positive and the remaining $n - m$ are zero.)

One can show that the first "principal component" (PC) of $\mathbf{x}$, maximises the explained variance of the multivariate regression of $\mathbf{x}$ on any linear combination of the columns of $\mathbf{x}$. Similarly, the second PC maximises the explained variability in the data, given the explanation already provided by the first PC. Since the eigenvectors are mutually orthogonal, all of the PCs are uncorrelated with each other, a very appealing feature for orthogonal regressoin. Note that PCs are not unique up to sign; i.e., multiplying a PC by -1 has no effect on the explanatory power of the PC.

One may write $\mathbf{x} = \mathbf{P} \times \mathbf{A}$, where $\mathbf{A}$ is the $m\times\, m$ matrix of "loadings" of the data on each of the PC. The fraction of the data variance explained by each of the successive PCs is given by $\lambda_i/\sum \lambda_i$ where λ_i is the i^{th} (sorted) eigenvalue of $\mathbf{x}\mathbf{x}^{\top}$, $i = 1, \ldots, m$. The cumulative fraction of the data variance explained by the first j PCs is given by $(\lambda_1 + \lambda_2 + ... + \lambda_j) / \sum \lambda_i$.

In empirical practice, when the data are correlated, the first few PCs tend to capture most of the variability. The leading PCs, then, can be used to represent the "meta-dimensions" in which the data fall. One could also say that the number of leading PCs; for example, those that capture between 50% and 90% of the total variance, represents the effective dimensionality of the data, which will be well less than in general. Table 8.1 shows a pseudo code implementation of the principal component analysis.

Table 8.1: $\boldsymbol{\beta} = \mathbf{PCA}\left(\mathbf{x}\right)$

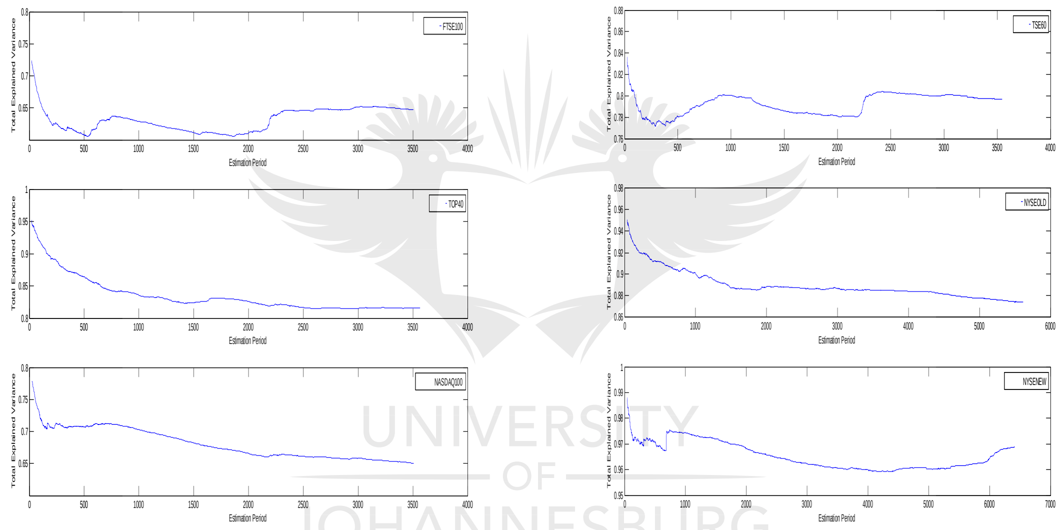
```
Inputs  $\mathbf{x}$ matrix of stock returns
        $0 < \gamma \leq 1$: fraction of total variance explained by PCA
Output  $\boldsymbol{\beta}$ structure containing factor Loadings
for     $t = 2 \times n, 2 \times n + 1, ...$
    1:  $\mathbf{D} = \mathbf{x}_t^1 - \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}_i^1$: standardise returns
    2:  $\mathbf{C} = \frac{1}{m-1}\mathbf{D} \times \mathbf{D}^\top$: covariance matrix
    3:  Calculate the eigen values and eigen vectors of $\mathbf{C}$
    5:  Sort the eigen values, $\lambda_{i,t}$, in descending order
    6:  Calculate the variance explained by each of the successive PC
    7:  $\left(\lambda_{1,t} + ... + \lambda_j\right) / \sum \lambda_i$: cumulative fraction of variance explained
    8:  $\beta\left\{t\right\} \longleftarrow$ associated eigen vectors such that $\left(\lambda_{1,t} + ... + \lambda_j\right) / \sum \lambda_i \geq \gamma$
end
```

8.4.4 Some Practical Considerations

PCA estimation is always done looking back at all available data from the trade date, thus simulating decisions, which might take place in real trading. We start with the first 252 days of trading data. Although this expanding window estimation procedure includes much older datasets at each estimation step, we believe there are at least two major advantages presented by our strategy. First, our methodology allows the portfolio manager to include all information available to him or her in the form of stock prices. Second, using an expanding window estimation technique also us to avoid the problem of chosing an optimal lookback window period.

A more important issue is that of selecting an optimal number of orthogonal factors. This leads to two possibilities: First, we take into account a fixed number of eigenvalues to extract the factors. Figure 8.4 shows the total variability in each data set explained by the first 20 principal components. It appears that the percentage of variance explained by the first 20 latent factors varies considerably with time. As a result we decided to take a variable number of eigenvectors, depending on the estimation date, in such a way that a sum of the retained eigenvalues exceeds a given percentage (set to 95%) of the trace of the correlation matrix. The latter condition is equivalent to saying that the truncation explains a given percentage of the total variance of the system.

Figure 8.4: Total Variance Explained by the Largest 20 Eigen Values



8.5 OPTIMAL SEPARATING HYPERPLANE VIA SUPPORT VECTOR MACHINES

This section propose a new SVM-inspired nonparametric Online stock selection algorithm, called Support Vector Stock Selection (SVSS). Not only does SVSS explicitly allow the separation of the stock selection and the portfolio construction problem, the algorithm also constructs investment strategies that have excellent historical performance in simulated trading without any assumptions being made about the statistical properties of stock prices.

8.5.1 Theoretical Foundations of Support Vector Machines

Since the publication of the Perceptron algorithm by Rosenblatt (1957), several machine-inspired algorithms have been proposed for classification problems. Among them is the SVM (Vapnik, 1995), which is known to improve the generalisation performance for binary classification tasks. Indeed, one of the key problem in machine-learning is the control of the generalisation ability of the models, and the margin idea introduced in SVMs appears to be a nice way to control it (Vapnik, 1995).

A classifier can frequently be represented as a function $f(x) : R^d \to R$. In a two-class case, a point is assigned to the positive class if $f(x) \geq 0$, and to the negative class otherwise. A classifier function $f(x)$ is linear if it can be expressed as $f(x; w, b) = w^\top x + b$ where w, b are parameters of the function and $\top$ denotes the transpose operator. A set of points $(x_i, y_i), i = 1, 2, ..., l$, where $y_i \in \{-1, 1\}$ are class labels, is called linearly separable if a linear classifier can be found so that $y_i f(x_i) > 0$, for all $i = 1, 2, ...l$.

A hyperplane $w^\top x + b = 0, \|w\| = 1$ is called a $\gamma-$margin separating hyperplance if $y_i \left(w^\top x + b\right) \geq \gamma$, for all (x_i, y_i) in a set S. Here $\gamma > 0$ is the margin and any separating hyperplane can be converted into this form. Suppose $y\left(w^\top x + b\right) \geq 1$, then by setting $w^* = \frac{w}{\|w\|}$ and $b^* = \frac{b}{\|w\|}$, we obtain a $\gamma-$margin separating hyperplane with $\gamma = \frac{1}{\|w\|}$.

Let us assume for a moment the data points $x_t \in R^d$, and labels $y_t \in \{1, -1\}$ are separated by the perceptron algorithm. The Perceptron algorithm maintains a weight vector $w \in R^d$ and classifies x_t according to the rule:

$$y^t = sign\left(w^\top x_t + b\right) \tag{8.24}$$

where b denotes the offset (intercept) parameter. The perceptron separates its domain R^d into two halfspaces; $\left\{x | w^\top x + b > 0\right\}$ and its complement. If $\widehat{y}_t = y_t$ then no updates are made. On the other hand, if $\widehat{y}_t \neq y_t$ the weight vector is updated as:

$$w \longleftarrow w + y_t x_t \text{ and } b = b + y_t$$

SVM can therefore be thought of as a method for constructing a special kind of rule, called a linear classifier, in a way that produces classifiers with theoretical guarantees of good predictive performance (the quality of classification on unseen data). The theoretical foundation of this method is given by the statistical learning theory of Vapnik (1995) and the so-called Novikov theorem.

Theorem 2. *Novikov's Theorem (see Novikov (1962)). Let S, $|S| = l$ be a training set i.e. a set of points with class labels, and let $R = \max \|x_i\|$. Suppose that there exist a $\gamma-$margin separating hyperplane (w, b) such that $y_i \left(w^\top x + b\right) \geq \gamma$, for all $1 \leq i \leq l$. Then the number of mistakes made by the Online perceptron algorithm on S is at most $\left(\frac{2R}{\gamma}\right)^2$*

This theorem proves that for a linearly separable set of points the number of mistakes made by the Perceptron algorithm is directly proportional to the ratio of the volume of the data to the measure of separation of the classes, γ. Although the Novikov's theorem bounds the number of errors made by the Perceptron algorithm we point that this is indeed the case only in the training sample. But in classifying stock price regimes we are interested in the accuracy of a classifier on unseen data, as this will have some correlation with the strategy's cumulative wealth growth. Such a number clearly cannot be computed exactly, but it turns out it can be bounded, as in the case of the support vector machines.

Definition 3. *The Vapnik-Chervonenkis (VC) dimension of a set of classifiers is the maximum number h of points that can be separated into all possible 2^h ways using classifiers in this set. If for any n there exists a set of n points that can be separated into two classes in all possible ways, the VC dimension of this set of functions is said to be infinite.*

Intuitively, VC dimension measures the complexity of the classifiers in the set. If the classifiers are simple, they have small VC dimensions. If they are complicated, the VC dimension is large. For example, the VC dimension of hyperplanes in R^d is known to be d + 1. The following two results bound the VC dimension of the set of γ-margin separating hyperplanes and the probability of misclassifying an unseen instance with such a hyperplane chosen on the training data.

Theorem 4. *Let $x \in X$ belong to sphere of radius R. Then the set of γ-margin separating* *hyperplanes has VC dimension h bounded by:*

$$h \leq \min\left(\left(\frac{R}{\gamma}\right)^2, d\right) + 1 \tag{8.25}$$

Corollary 5. *With probability 1-η the probability of a test example not being separated correctly by a $\gamma-$margin hyperplane has the bound*

$$P_{error} \leq \frac{m}{l} + \frac{E}{2}\left(1 + \sqrt{1 + \frac{4m}{lE}}\right) \tag{8.26}$$

where $E = 4\frac{h\left(\ln\frac{2l}{h}+1\right)-\ln\frac{\eta}{4}}{l}$, *m is the number of training samples not separated correctly by the $\gamma-$margin hyperplane, and h is the bound of the VC dimension given in the previous theorem.*

The result of the corollary is vital for our stock selection process. It shows that the bound on the probability of making a mistake on unseen stock price data is proportional to the VC dimension of the set of classifiers. In other words, everything else being equal, a classifier with a lower VC dimension is likely to be a better out-of-sample predictor of stock returns. Since the upper bound on VC dimension is inversely proportional to the margin, the strategy for building a good stock price classifier is to have as large a margin as possible while keeping the number of errors on the training set low. Given that the probability of making a mistake is inversely proportional to the sise of the margin, our strategy is therefore to find a classifier with the largest margin that still correctly separates the training points. The maximal-margin separating hyperplane in this chapter is found by using the SVM algorithm.

8.5.2 Mechanics of the SVM

In many applications, including stock price forecasting, SVM has been shown to generate remarkable generalisation properties. The SVM algorithm is based on the idea of the structural risk minimisation principle, which seeks to minimise an upper bound of the generalisation error rather than the empirical error commonly implemented in many econometric and statistical models. Therefore, SVM not only minimises training error but its learning capacity is limited by VC dimension so as to reduce the probability of over-fitting the model.

8.5.2.1 Linearly Separable Problems Consider a two-class, linearly separable classification problem as shown in Figure 8.1. According to the idea of large margin classifiers, the decision boundary should be as far away from the data of both classes as possible. We can achieve this by maximising the margin between the origin and the line $w^\top x = k$. This implies that $d = \frac{2}{\|w\|}$. Let $\{x_1, x_2, ..., x_n\}$ be our data (represented here by the cross-sectional values of the beta coefficients) set and let $y_i \in \{1, -1\}$ be the class label of x. The decision boundary should classify all points correctly, that is $y_i \left(w^\top x + b\right) \geq 1$ for $i = 1, 2, ..., n$. The decision boundary can be found by solving the following constrained optimisation problem.

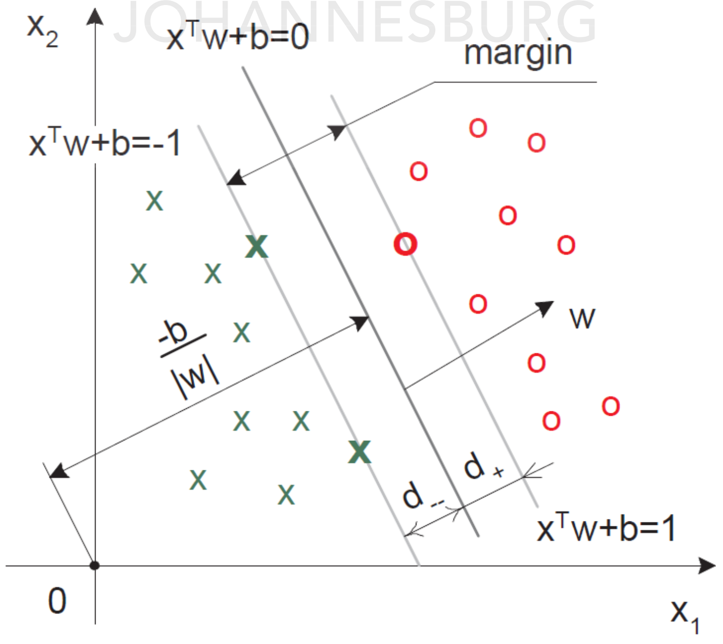
$$\begin{array}{ll} \textit{minimise} & \frac{1}{2}\|w\|^2 \\ \textit{subject to} & y_i \left(w^\top x + b\right) \geq 1 \text{ for } i = 1, 2, ..., n. \end{array}$$

This is a constrained optimisation problem. Solving it requires some new tools. Suppose we want to minimise $f(x)$ subject to $g(x) = 0$ A necessary condition for x_0 to be a solution:

$$\begin{cases} \frac{\partial}{\partial x}\left(f(x) + \alpha g(x)\right)|_{x=x_0} & = & 0 \\ g(x) & = & 0 \end{cases}$$

where α is the lagrange multiplier

Figure 8.1: Large Margin Classification



For multiple constraints $g_i(x) = 0, i = 1, \ldots, m$, we need a Lagrange multiplier α_i for each of the constraints

$$\begin{cases} \frac{\partial}{\partial x}\left(f(x) + \sum\limits_{i=1}^{n}\alpha_i g_i(x)\right)|_{x=x_0} & = & 0 \\ \text{s.t. } g_i(x) & = & 0 \end{cases}$$

The case for the inequality constraint $g_i(x) \leq 0$ is similar, except that the Lagrange multiplier α_i should be positive

Now let us rewrite our optimisation problem as:

$$\begin{array}{ll} \textit{minimise} & \frac{1}{2}\left\|w\right\|^2 \\ \textit{subject to} & 1 - y_i\left(w^\top x + b\right) \leq 0 \text{ for } i = 1, 2, ..., n. \end{array}$$

The Lagrangian could be written as:

$$L = \frac{1}{2}w^\top w + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right)\right)$$

Setting the gradient of L w.r.t. w and b to zero, we have:

$$\begin{cases} w + \sum\limits_{i=1}^{n}\alpha_i\left(-y_i\right)x_i & = & 0 & \Longrightarrow & w & = & \sum\limits_{i=1}^{n}\alpha_i y_i x_i \\ \sum\limits_{i=1}^{n}\alpha_i y_i & = & 0 \end{cases}$$

If we substitute $w = \sum\limits_{i=1}^{n}\alpha_i y_i x_i$ into L, we have:

$$\begin{array}{rcl} L & = & \frac{1}{2}\sum\limits_{i=1}^{n}\alpha_i y_i x_i^\top \sum\limits_{j=1}^{n}\alpha_j y_j x_j + \sum\limits_{i=1}^{n}\alpha_i\left(1 - y_i\left(\sum\limits_{j=1}^{n}\alpha_j y_j x_j^\top x_i + b\right)\right) \\ & = & \frac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum\limits_{i=1}^{n}\alpha_i - \sum\limits_{i=1}^{n}\alpha_i y_i\sum\limits_{j=1}^{n}\alpha_j y_j x_j^\top x_i - b\sum\limits_{i=1}^{n}\alpha_i y_i \\ & = & -\frac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum\limits_{i=1}^{n}\alpha_i \end{array}$$

We note that this new expression of the Lagrangian is a function of α_i only, and we use this to formulate the so called "dual problem". The original problem is known as the primal problem. The objective function of the dual problem needs to be maximised!

The maximisation of the dual problem is therefore:

$$\begin{array}{llcl} \max & W(\alpha) & = & \sum\limits_{i=1}^{n}\alpha_i - \frac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\ \text{subject to} & \alpha_i & \geq & 0 \\ & \sum\limits_{i=1}^{n}\alpha_i y_i & = & 0 \end{array}$$

This is a quadratic programming (QP) problem for which a global maximum of α_i can always be found. Finding α_i allows us to recover w as $w = \sum\limits_{i=1}^{n}\alpha_i y_i x_i$

Many of the α_i are zero, which indicates that w is a linear combination of a small number of data points. This "sparse" representation can be viewed as data compression. x_i with non-zero α_i are called SVs. The decision boundary is determined only by the SV. Let t_j $(j = 1, ..., s)$ be the indices of the s support vectors. We can write

$$w = \sum_{j=1}^{s}\alpha_{t_j} y_{t_j} x_{t_j}$$

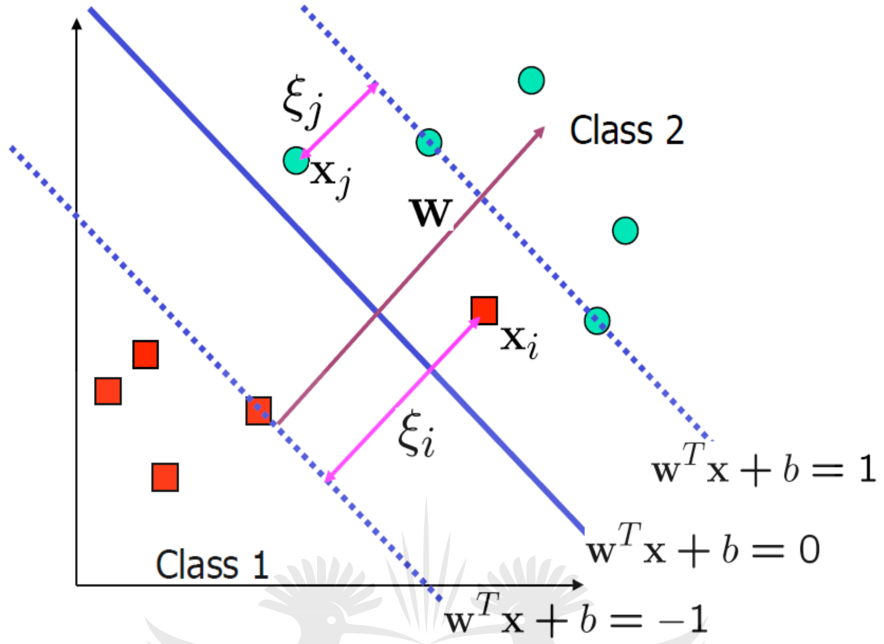
For testing with a new data z, compute

$$w^\top z + b = \sum_{j=1}^{s}\alpha_{t_j} y_{t_j}\left(x_{t_j}^\top z\right) + b$$

and classify z as class 1 if the sum is positive, and class 2 otherwise Note: w need not be formed explicitly. To solve the quadratic programming problem we use the very popular sequential minimal optimisation (SMO).

8.5.2.2 Nonlinearly Separable Problems and Soft Margin Hyperplane We can allow "error" ξ_i in classification; it is based on the output of the discriminant function $w^\top x + b$. ξ_i approximates the number of misclassified samples (see Figure 8.2)

Figure 8.2: SVM and Slack Variables



ξ_j
$\mathbf{x}_j$
Class 2
$\mathbf{w}$
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=0$
Class 1
$\mathbf{w}^T\mathbf{x}+b=-1$

If we minimise Σ_i, ξ_i, ξ_i can be computed by

$$\begin{cases} w^\top x+b & \geq & 1-\xi_i & y_i & = & 1 \\ w^\top x+b & \leq & \text{-1}+\xi_i & y_i & = & -1 \\ \xi_i & \geq & 0 \end{cases}$$

ξ_i are "slack variables" in the optimisation process. We note that $\xi_i = 0$ if there is no error for x_i. ξ_i is an upper bound of the number of errors. To reformulate the optimisation problem we introduce a parameter C that represents a tradeoff parameter between the error and the margin. The optimisation problem becomes:

$$\begin{array}{ll} \textit{minimise} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \textit{subject to} & y_i\left(w^\top x+b\right) \geq 1-\xi_i,\ \xi_i \geq 0 \end{array}$$

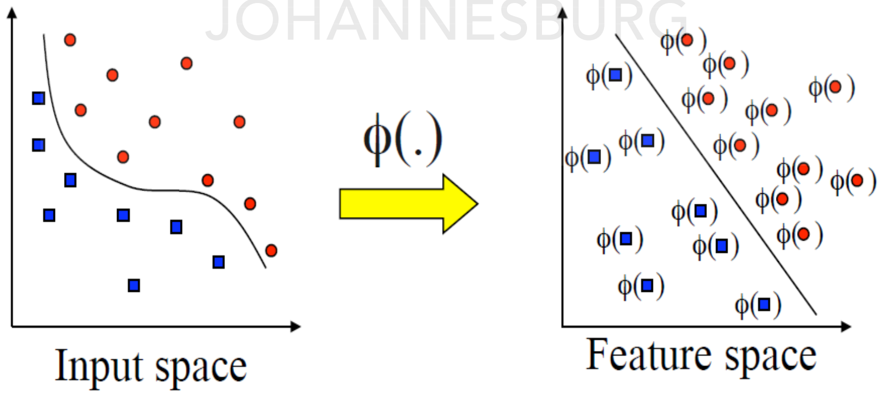
The dual of this new constrained optimisation problem is

$$\begin{aligned} \max \quad & W(\alpha) &=& \sum_{i=1}^{n}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\ \text{subject to} \quad & C \geq \alpha_i &\geq& 0 \\ & \sum_{i=1}^{n}\alpha_i y_i &=& 0 \end{aligned}$$

Finding α_i allows us to recover w as $w = \sum_{i=1}^{s}\alpha_{t_i} y_{t_i} x_{t_i}$. This is very similar to the optimisation problem in the linear separable case, except that there is an upper bound C on α_i now. Once again, a QP solver can be used to find α_i.

8.5.2.3 Extension to nonlinear Decision Boundary So far, we have only considered a large-margin classifier with a linear decision boundary. In order to generalise the classifier to a nonlinear decision boundary, the trick use by the SVM algorithm is to transform x_i to a higher dimensional space. Given the space of input point x_i we define the feature space as the space of $\phi(x_i)$ after transformation. Therefore, a linear operation in the feature space is equivalent to a nonlinear operation in the input space. Classification can become easier with a proper transformation.

Figure 8.3: Nonlinear to Linear Mapping Function



Computation in the feature space can be costly because it is high dimensional. The feature space is typically infinite-dimensional the so-called "Kernel Trick" makes the calculations less tedious.

Recall the SVM optimisation problem:

$$\begin{array}{llcl} \max & W(\alpha) & = & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\ \text{subject to} & C \geq \alpha_i & \geq & 0 \\ & \sum_{i=1}^{n}\alpha_i y_i & = & 0 \end{array}$$

The data points only appear as inner product As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly Many common geometric operations (angles, distances) can be expressed by inner products Define the Kernel function K by $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$. Example of such Kernel functions are:

Table 8.2: Kernel Functions

Polynomial Kernel with degree d	$K(x, y) = \left(x^\top y + 1\right)^d$
Radial basis function Kernel with width σ	$K(x, y) = \exp\left(-\lVert x - y\rVert^2 / (2\sigma^2)\right)$
Sigmoid with parameter κ and θ	$K(x, y) = \tanh\left(\kappa x^\top y + \theta\right)$

which yield the final optimisation problem as follows

$$\begin{array}{llcl} \max & W(\alpha) & = & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \\ \text{subject to} & C \geq \alpha_i & \geq & 0 \\ & \sum_{i=1}^{n}\alpha_i y_i & = & 0 \end{array}$$

Of course the same modification must apply for testing the new data z. z is classified as class 1 if $f \geq 0$,and as class 2 if $f < 0$, where $w = \sum_{i=1}^{s}\alpha_{t_i} y_{t_i}\phi(x_{t_i})$ and $f = w^\top \phi(z) + b = \sum_{j=1}^{s}\alpha_{t_j} y_{t_j} K\left(x_{t_j}, z\right) + b$

8.5.3 Some Practical Considerations

The second step in the estimation of our Support Vector Stock Selection (SVSS) model is to construct an optimal separating hyperplane in the hidden feature space of factor loadings, using SVM techniques. Following Fan and Palaniswami, (2001), we assign a target class $y_i \in \{-1, 1\}$ of each stock to indicate its expected performance in the following way. At any given trading period t we rank the expected performance of stocks in ascending order, with the top 25% (or lower quartile called Q_1) being labelled as expected high-performing stock and assigned a $class = +1$ and the others stocks labelled as expected low-yielding stocks and assigned a $class = -1$. We approximate the expected returns by the average returns of individual stock returns over the recent past. The lookback period for this averaging is given by a parameter called the window sise, w. The estimated factor loading from PCA of historical stock price returns up to time $t-1$ are the feature inputs to the SVM. There are two additional parameters that need to be chosen by the portfolio manager in our nonlinear SVM setting. The first involves the choice of the kernel function. As discussed previously, the SVM literature provides many candidate kernel functions that could be chosen for different problem formulations. Fortunately, our historical simulations demonstrate that our model is not too dependent on which kernel function we use. Our historical simulations are done with a Gaussian kernel. The second deal with the parameter C that represents a tradeoff parameter between the error and the margin. Generally, increasing C would improve classification accuracy on the training set, but also tends to lead to over-fitting problems. Although this parameter plays an important role in the performance of SVMs (Tay and Cao, 2001) we decided to use default setting values in the Matlab environment. Matlab default values were also used for all other parameters in order to avoid some form of over-fitting

8.6 SUPPORT VECTOR MACHINE FOR PORTFOLIO SELECTION (SVMPS) ALGORITHM

So far in this chapter we have presented a very efficient algorithm with theorectical guarantees for the stock selection problem inspired by the idea of SVM for classification. However, investing in the equity market requires not only the stock selection problem, but also involves portfolio allocation decision. Portfolio allocation is understood as the process of choosing the proportions of various stocks to be held in a portfolio, in such a way as to make the portfolio better than any other according to some criterion. All the models described in previous chapters of this thesis have been shown to be theoretically well-founded and empirically robust portfolio allocattion algorithms. One such PS algorithm is the so-call KMAC (see

Chapter 3), which builds on the idea of the TAPR and the Anticor (AC) algorithm of Borodin et al., (2006).

Given the SVM-inspired stock selection algorithm presented in this chapter and the KMAC PS algorithm a natural problem of how to combine these independent framework emerges. It turns out that one can recombine these two algorithms very effectively in a simple and intuitive way.

At time t, the portfolio manager is faced with two independent but related problems-the stock selection and portfolio optimisation/allocation problems. While the stock selection is encapsulated in the binary vector $\mathbf{I}_t \in \{0,1\}$ of length m the portfolio optimisation is represented by a weight vector $\mathbf{b}_t\left(\mathbf{x}_1^t\right)$ of length m. An effective but yet intuitive way to combine these two important problems is for the portfolio manager to invest in stock i only when there is some "agreement" ($\mathbf{I}_{t,j} = 1$ and $\mathbf{b}_{t,j}\left(\mathbf{x}_1^t\right) > 0$) between the stock selection and portfolio optimisation outputs. The final portfolio weight vector, $\widetilde{\mathbf{b}}_t\left(\mathbf{x}_1^t\right)$, is therefore given by:

$$\widetilde{\mathbf{b}}_{t,j} = \frac{\mathbf{b}_{t,j}\left(\mathbf{x}_1^t\right) \circ \mathbf{I}_{t,j}}{\displaystyle\sum_{j=1}^{m} \mathbf{b}_{t,j}\left(\mathbf{x}_1^t\right) \circ \mathbf{I}_{t,j}} \text{and} \left(\sum_{j=1}^{m} \widetilde{b}_{t,j} = 1\right) \tag{8.27}$$

The final portfolio weights vector in Equation 8.27, $\widetilde{\mathbf{b}}_t$, is therefore expressed as a complex combination of the stock selection and portfolio optimisation weights. In the case of total or partial "disagreement" between the stock selection and the portfolio construction outputs ($\mathbf{I}_{t,j} = 0$ **or** $\mathbf{b}_{t,j}\left(\mathbf{x}_1^t\right) = 0$ for $j = 1, 2, ..., m$) we will arbitralily set $\widetilde{\mathbf{b}}_t = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$.

This simple and flexible representation presents major benefits. By designing individual and independent strategies for the portfolio construction and the stock selection, the portfolio manager is afforded full control over what stocks to buy or sell together with the respective quantities. Very often, fund managers have subjective insights into the workings of a business and the implied direction of its stock price. This insight cannot be easily taken into account if a typical "black box" Online PS algorithm is the sole tool available to the fund manager. Perhaps the most significant property of our proposed methodology is that it provides the fund manager with theorectical bounds in terms of the mistake he or she can make in the test sample while selecting stocks via the SVM algorithm. Our framework therefore seems to speak directly to the way portfolio managers construct their funds in real life. We expect this algorithm to provide very consistent performance relative to its peers and view it as a very effective way to manage clients' assets.

Table 8.2 presents the pseudo code for our final Support Vector Machine for PS (SVMPS) problems

```
Table 8.3:   SVMPS($\mathbf{x}, w, k$)
Inputs       $w$: window sise for the SVM for stock selection
             $k$: window sise for the KMAC
             $\mathbf{x} \in \mathbb{R}^{T\times m}$ matrix of price relatives
Output       $\widetilde{\mathbf{b}}$ Final portfolio weight vector
Initialise   $\boldsymbol{\beta} \longleftarrow \mathbf{PCA}\,(\mathbf{x})$: structure of estimated factor loading
             $\mathbf{b} \longleftarrow \mathbf{KMAC}\,(\mathbf{x}, k)$
for          $t = k, k+1, ...$
        1:   $\mathbf{ER}_t \longleftarrow \frac{1}{w}\sum_{i=1}^{w}\mathbf{x}_t^{t-w+1}$
        2:   $\mathbf{z} \longleftarrow \text{rank}(\mathbf{ER}_t)$
        3:   $y_{i,t} \longleftarrow \left\{\begin{smallmatrix} +1 \text{ if } z_i \in Q_1(\mathbf{z}) \\ -1 \text{ otherwise}\end{smallmatrix}\right.$ : vector of responses
        4:   $\boldsymbol{\beta}_{train} \longleftarrow \beta\,\{t-1\}$ and $\boldsymbol{\beta}_{test} \longleftarrow \beta\,\{t\}$
        5:   svm $\longleftarrow$ svmtrain($\boldsymbol{\beta}_{train}$,$\mathbf{y}_t$): train SVM
        6:   $\mathbf{I}_t \longleftarrow$ svmclassify(svm,$\boldsymbol{\beta}_{test}$): predicted classes
        7:   Retain only the predicted classes such that $I_{i,t} = 1$
        8:   $\widetilde{\mathbf{b}}_t = \frac{\mathbf{b}_t\left(\mathbf{x}_1^t\right)\circ\mathbf{I}_t}{\sum_{j=1}^{m}\mathbf{b}_t\left(\mathbf{x}_1^t\right)\circ\mathbf{I}_t}$
        9:   if $\widetilde{\mathbf{b}}_t = \mathbf{0}$
       10:       $\widetilde{\mathbf{b}}_t = \left(\frac{1}{m}, ..., \frac{1}{m}\right)$
       11:   end if
end for      end
```

8.6.1 Mixture of Experts

Our proposed SVMPS algorithm has two main parameters that need to be fine tuned by the portfolio manager. The first parameter is the window-length , k, applicable to the KMAC PS algorithm while the second parameter is the window-length w for the SVM-inspired stock selection algorithm.

As usual we argue that there is no universally agreed method to select the optimal window-sise parameter k and w. Our strategy is simply to define a large number of experts, each expert indexed by its window sise parameters k and w such that:

$$\{E(k,w) : 1 \leq k \leq 10, 1 \leq w \leq 10\}$$

As in Gyorfi et al. (2006) we form a mixture of all experts using a positive probability distribution $q_{k,w}$ on the set of all pairs (k,w) of positive integers. The investment strategy simply weights these experts $\mathbf{H}^{k,w}$ according to their past performances and the $q_{k,w}$ such that after the i^{th} trading period the investor wealth becomes:

$$S_i = \sum_{k,w} q_{k,w} S_i \left(\mathbf{H}^{k,w}\right) \tag{8.28}$$

where $S_i\left(\mathbf{H}^{k,w}\right)$ is the capital accumulated after i trading period using the expert $\mathbf{H}^{k,w}$ with initial capital $S_0 = 1$. We then form our final portfolio by weighting all expert portfolio using the following:

$$\mathbf{b}\left(\mathbf{X}_1^{i-1}\right) = \frac{\sum_{k,w} q_{k,w} S_{i-1}\left(\mathbf{H}^{k,w}\right) h^{k,w}\left(\mathbf{x}_1^{i-1}\right)}{\sum_{k,w} q_{k,w} S_{i-1}\left(\mathbf{H}^{k,w}\right)} \tag{8.29}$$

8.6.2 Trading Costs

As usual, at the beginning of the i^{th} trading day, the portfolio manager rebalances the portfolio from the previous closing price adjusted portfolio $\mathbf{b}_{i-1}$ to a new portfolio $\mathbf{b}_i$. Specifically, we consider a transaction cost rate $c \in (0,1)$, so the transaction cost will be charged according to:

$$\frac{c}{2}\sum_{k}\left|\overset{\sim k}{\mathbf{b}_t} - \mathbf{b}_t^k\right| \tag{8.30}$$

Thus, with the transaction cost rate the total wealth achieved by the strategy becomes

$$S_T^c = S_0\prod_{i=1}^{T}\left[(\mathbf{b}_i, \mathbf{x}_i)\left(1 - \frac{c}{2}\sum_{k}\left|\overset{\sim k}{\mathbf{b}_t} - \mathbf{b}_t^k\right|\right)\right] \tag{8.31}$$

8.7 EMPIRICAL RESULTS

This section presents numerical results obtained by applying the SVMPS-based algorithm to six financial market datasets described in Chapter 3. As usual, the back-testing experiments in this section will consist of running the signals through historical data, with the estimation of parameters, signal evaluations and portfolio re-balancing performed on a daily basis.

8.7.1 Analysis of Cumulative Wealth

For the first experiment we evaluate the compounded wealth achieved by the SVMPS-based learning-to trade algorithm, including a 10 basis-point transaction cost over the entire sample period. Figure 8.5 summarises the total cumulative wealth achieved by the SVMPS algorithm for the six market datasets. The market index is calculated as an equally weighted Buy-and-Hold portfolio on all stock available for that particular market. Figure 8.5 demonstrates that the SVMPS-based PS algorithm achieves very good performance on all datasets, including the more recent and previously untested stock market data. For example, on the South African TOP40 index data set, the total wealth achieved by the SVMPS strategy impressively increases from \$1 to almost \$3006. Not only is the wealth growth much higher than the \$13.6 achieved by the market index over the same 15-year periods, but also this wealth is significantly higher than that demonstrated by existing state-of-the-art PS algorithms. During that period, the best stock generates \$87.8 and the best constantly rebalanced portfolio in hindsight achieves only a growth in wealth of \$115. In the case of the FTSE100 in London

1$ rises to $2359 after 15 years. This rise appears spectacular when one compares it to the market index performance in the same period (see Figure 8.5). One impressive fact to note is that, unlike most of the algorithms presented earlier, the performance of the SVMPS-based PS algorithm seems to be very strong in recent times as well. We view this as a good sign that the strategy is very robust accross different markets and time frames.

Figure 8.5: SVMPS Performance Comparison

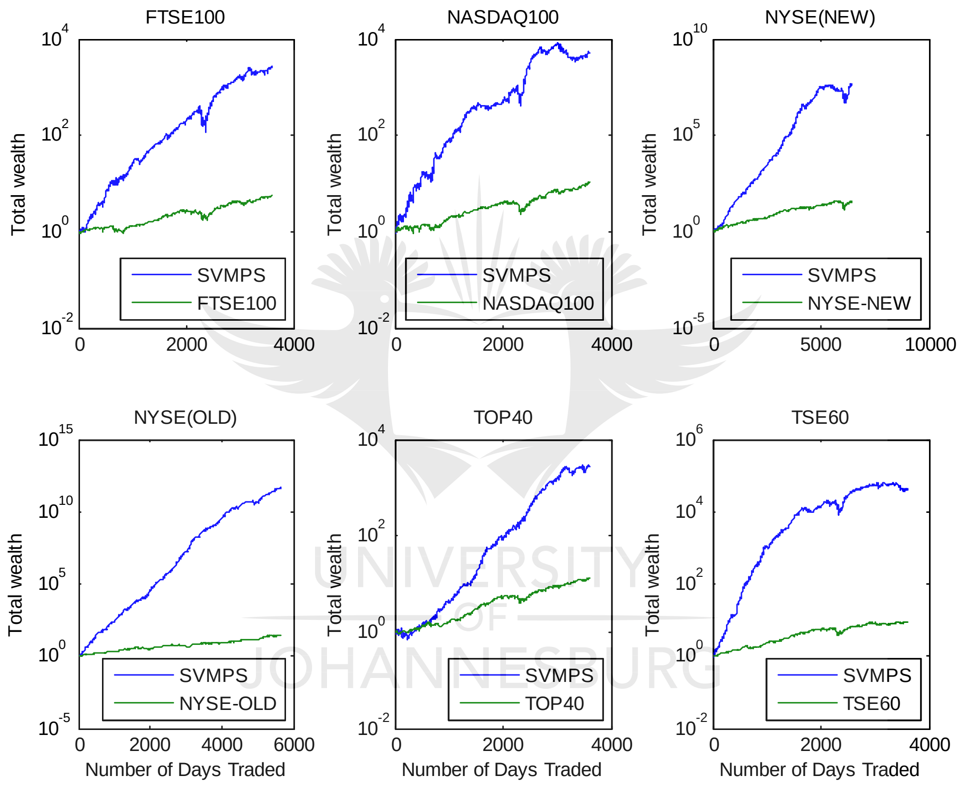


Table 8.4 presents the full sample and some selected sub-periods cumulative performance of the SVMPS algorithm against respective benchmarks in the data set considered. From these results it is easy to see that our proposed algorithm significantly outperforms the respective market indices for all datasets and for most time periods. Over the full sample, the SVMPS algorithm displays a very impressive CAGR of around 50% and 101%, using the TOP40 and TE60 datasets respectively. This compares very farourably with a CAGR of 20% and 17% for both market indices. Similar CAGR can be observed from all other datasets, including the NASDAQ, the NYSE (O) and the NYSE (N). Table 8.4 also presents the year

by year cumulative performance of the SVMPS algorithm against its respective benchmarks for the data set considered. At the height of the financial crisis in 2008, for example, our proposed methodology outperformed in all considered markets except the FTSE100 in the UK. For most datasets, the cumulative annualised growth rate for the SVMPS-based PS algorithm surpasses that achieved by the respective market indices for most subsamples under consideration.

Table 8.4: Cumulative Returns Over Subsample Trading Periods

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
Full	0.30	0.13	0.50	0.20	1.01	0.17	0.41	0.18	0.64	0.15	3.46	0.16
1Yr	0.10	0.21	0.29	0.17	-0.01	0.11	0.70	0.39	0.89	0.29	0.24	-0.01
2Yrs	0.06	0.22	0.31	0.22	0.36	0.10	0.19	0.26	0.35	-0.04	0.87	0.16
3Yrs	0.02	0.12	0.30	0.17	0.25	0.05	0.13	0.18	0.37	-0.07	2.27	0.23
4Yrs	0.18	0.16	0.27	0.19	0.32	0.09	0.27	0.21	0.30	0.03	2.28	0.19
5Yrs	0.42	0.23	0.30	0.23	0.32	0.17	0.46	0.30	0.33	0.05	2.42	0.20
7Yrs	0.44	0.12	0.46	0.16	0.12	0.09	0.23	0.17	0.51	0.08	3.02	0.20
10Yrs	0.37	0.15	0.59	0.22	0.30	0.14	0.36	0.17	0.81	0.07	4.44	0.19
15Yrs	0.37	0.13	0.57	0.21	0.57	0.15	0.38	0.18	0.93	0.07	5.80	0.13

8.7.2 Risk Analysis

Table 8.5 shows some of the SVMPS risk measures for the six datasets considered. From the results, we see that SVMPS-based strategy has an annualised volatility of returns, σ, that is higher than the one achieved by its respective benchmarks. However, when returns are adjusted for risk (volatility), the Sharpe Ratio (SR) generated by the SVMPS-based algorithms are generally significantly higher than the one generated by the respective stock market indices. The SVMPS-based strategy therefore generates much better risk-adjusted returns.

Table 8.5: Risk Statistics of the SVMPS-Based PS Algorithm

	FTSE100		TOP40		TSE60		NASDAQ		NYSE(N)		NYSE(O)	
	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT	ALG	MKT
σ	0.30	0.18	0.28	0.16	0.58	0.17	0.42	0.23	0.38	0.19	0.54	0.13
α	0.01	0.00	0.06	0.00	0.25	0.00	0.12	0.00	0.14	0.00	0.53	0.00
β	1.19	1.00	1.14	1.00	1.62	1.00	1.23	1.00	1.09	1.00	1.45	1.00
γ	3.68	0.00	2.52	0.00	-1.21	0.00	-1.92	0.00	0.96	0.00	4.35	0.00
$\beta\uparrow$	1.29	1.00	1.18	1.00	1.42	1.00	1.13	1.00	1.11	1.00	1.55	1.00
$\beta\downarrow$	1.04	1.00	1.02	1.00	1.56	1.00	1.25	1.00	0.98	1.00	1.23	1.00
SR	0.78	0.37	1.33	0.78	1.37	0.59	0.87	0.53	1.32	0.44	2.90	0.64
MDD	-0.44	-0.40	-0.34	-0.30	-0.77	-0.41	-0.68	-0.47	-0.61	-0.64	-0.37	-0.37
PP	0.54	0.56	0.55	0.57	0.56	0.58	0.55	0.56	0.55	0.55	0.59	0.53
PP$\uparrow$	0.79	1.00	0.78	1.00	0.75	1.00	0.81	1.00	0.75	1.00	0.72	1.00
PP$\downarrow$	0.22	0.00	0.26	0.00	0.30	0.00	0.23	0.00	0.30	0.00	0.43	0.00

Further, the SVMPS-based PS algorithm generates maximum drawdowns on five out of six stock datasets.that are very similar to those generated by the respective uniform Buy-and-Hold benchmarks. The maximum drawdown is -44% for FTSE100 against -40% for the FTSE100 benchmark, -34% against -30% for the TOP40, -77% against -41% for the TSE60, -68% against -47% for the NASDAQ100. When the market index was up, the SVMPS-based algorithm generated positive returns more than 75% of the time on average for all datasets. And when the market was down the SVMPS-based PS algorithm generated positive returns about 30% of the time on average on all datasets. Another important indicator is given by the strength and direction of the market-timing parameter, γ. This parameter is positive and statistically significant for all datasets except for the TSE60 in Canada. This suggests that the value add of the SVMPS-based algorithm lies in its ability to find stocks that are likely to rise more in a rising market or fall less in a falling market.

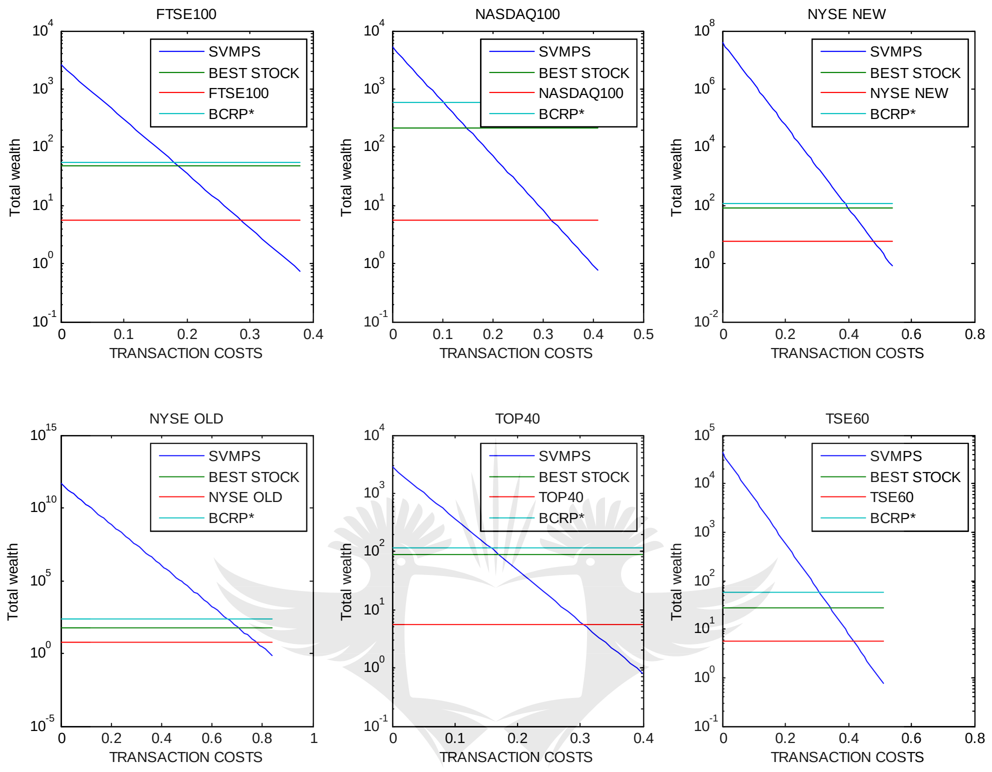
In summary, the simulation results seem to demonstrate that the SVMPS-based PS algorithm achieves very impressive cumulative wealth growth on all datasets. These encouraging results show that the SVMPS-based PS algorithm is capable of achieving an excellent trade-off between return and risk. In our view the SVMPS-based PS algorithm is indeed a very robust investment strategy.

8.7.3 SVMPS and Brokerage Costs Analysis

As in the case of most algorithms presented in this thesis, the performance of the SVMPS algorithm is determined not only by the quality of the investment strategy but also by the terms of execution. The potential outperformance of the SVMPS strategy will be heavily affected if the commissions paid in order to execute every transaction are high or if the prices of stocks that are selected for inclusion in the portfolio rise systematically between investment decision and completion of trade execution.

Figure 8.6 shows that the SVMPS investment algorithm can tolerate relatively small proportional commission rates and still beat competing benchmarks-including the best stock, the equally weighted market index and best constantly rebalanced portfolio in hindsight called BCRP*. The graphs in Figure 8.6 depict the total returns of the SVMPS algorithm for varying proportional commission factor $c = 0.1\%, 0.2\%, ...$ It appear that our strategy can withstand reasonable brokerage commissions. For example, with a commission cost of $c = 0.1\%$ or 10 basis points (10 bps), the algorithm still beat the best stock, the market and the BCRP* portfolio in all markets we considered. In fact, even with $c < 0.25\%$, our algorithm beats all its respective market indices.

Figure 8.6: Transaction Cost Analysis for Selected Markets



8.8 CONCLUSION

In this chapter, we have presented a new Online approach to PS algorithms called SVMPS. Building on PCA and SVM, SVMPS produces portfolios that substantially outperform their benchmarks equivalent on real-market datasets even after accounting for modest but reasonable brokerage commissions. To arrive at these portfolios SVMPS treats the stock selection and portfolio construction problems faced by professional portfolio managers as two distinct but equally important problems. The rationale for this is quite simple to grasp. An Online learning PS model can be good at optimally allocating the available capital amongst securities, but the non-Quantitative managers may know things about different stocks that are not captured by the model. Therefore, we need to blend Quantitative and Fundamental approaches into a Hybrid approach. Any reasonable approach to investing needs to consider ways in which Quantitative techniques could enhance a Fundamental manager's portfolio performance. The solution proposed in this chapter does exactly that. Although we did not explicitely use fundamental data files in our stock selection approach, the portfolio manager could use firm-specific attributes or any other information to select candidate stocks and simply override the output of our SVM based stock selection output. Our approach is not only novel, but it is also very flexible and allows the portfolio manager to have full control on how to design better portfolios using both state-of-the-art Online machine-learning algorithms and stock selection algorithms that can possess out-of-sample bounds.

9.0 SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

9.1 SUMMARY

This thesis has developed and empirically analyzed a rich set of new, original and robust Online PS algorithms. The proposed algorithms are simple in nature, easy to implement, and have very few parameters that are easy to set. Our empirical studies show that all our proposed algorithms not only substantially beat the market and the best stock, but also consistently surpass a variety of so-called state-of-the-art PS algorithms.

The first contribution, Chapter 4, was to empirically analyse some existing Online PS algorithms using both existing and more recent and previously untested datasets, including the South African datasets. Instead of covering as many algorithms as one can possibly do (see Li et al. (2013)), this thesis took a deeper look at those algorithms that have been shown to be very robust investment strategies with very good finite-horison performance. The algorithms studied in this chapter have been shown to outperform, by an exponential factor, both the best stock and the best constant rebalanced portfolio benchmark. The algorithms surveyed in this chapter included the nonparametric Kernel-based sequential investment strategies of Gyorfi et al. (2006), the nonparametric Nearest Neighbor-based sequential investment strategies of Gyorfi et al. (2008), Correlation-Driven Nonparametric learning algorithm for PS (Li et al. (2011)), the Anticor algorithm (Borodin et al. (2006)), the Passive Aggressive Mean Reversion strategy for PS (Li et al. (2012a)) and the Confidence Weighted Mean Reversion strategy for PS (Li et al. (2013)). The main finding in this chapter was that all the selected algorithms performed remarkably well for old datasets but more recent and previously untested data have shown to be more challenging.

Applying numerically intensive algorithms to data dating back to the 1970s or 1980s amounts to some kind of "data snooping" itself as both the techniques and the computational power could not have been available to test these algorithms at that particular time. As the computational power required to participate successfully in today's financial markets continues to grow exponentially, more and more portfolio managers are implementeing complex algorithms and by default eroding some of the complex inefficiencies that might have been present. Our ultimate view is that any Online PS algorithm should be evaluated on its ability to perform on recent and previously untested datasets. That is what we have done throughout this thesis.

In Chapters 5, 6, 7 and 8 some heuristic preprocessing procedures of the relative price sequences are suggested. We notice that by transforming the relative price sequences we loose information, which means that the asymptotic growth rate cannot be increased. The only advantage of such procedures might be that they improve the performance for finite time horizon. These procedures are motivated by some optimization criteria, which are usually far from the growth optimality. The main power of the log-optimal strategies is that they can utilize the significant hidden cross-correlation between the assets.

In Chapter 5, we studied and analyzed the Online PS problem by introducing a novel idea called the Trend-Adjusted Price Relatives (TAPR). Using TAPR allowed us to derive a whole new family of algorithms that combine powerful Online PS algorithms with ideas from signal processing and statistical learning. Our new algorithms produce portfolios that substantially outperform their benchmark equivalents for real-market datasets More particularly, three new Online PS algorithms, the ACM, the KAC and the KACM were introduced, described and their empirical performance has been thoroughly analyzed. The empirical results of these new algorithms were investigated through a rich set of numerical experiments using six major stock market datasets. The results emphasised the relevance of the proposed set of new Online PS algorithms and underlined the relevance of their specification to outperform existing benchmark PS algorithms.

In Chapter 6 we presented a novel nonparametric learning-to-trade sequential PS algorithm called "Delay Coordinate Embedding algorithm for PS" (DCEPS). Our proposed DCEPS algorithm built on Takens (1981) delay coordinate embedding theorem, which allowed us to construct a data matrix of overlapping samples in order to increase the precision of parameter estimates. We derived the DCEPS algorithm from a time-delay embedding of multiple time series in order to capture nonlinear information found in complex dynamical systems of stock returns. Our technique created a time-lagged version of the original stock returns in order to discover hidden patterns normally not detected in a linear space. The main conclusion here was that the DCEPS algorithm outperformed all existing state-of-the-art PS algorithm for all datasets and for most risk measures.

Another novel nonparametric similarity-driven empirical PS algorithm was introduced in Chapter 7. We termed this new algorithm multivariate singular spectrum analysis for PS or MSSAPS. As in the case of the DCEPS, the MSSAPS builds on Takens (1981) delay coordinate embedding theorem in order to discover hidden patterns normally not detected in a linear space. We showed that the MSSAPS not only exploits effective higher-dimensional statistical correlations between market windows via the delay embedding feature, but also benefits from the exploration of powerful nonparametric machine-learning techniques. As in most of the algorithms proposed in this thesis the MSSAPS made no assumtpions about the statistical properties of stocks prices and the algorithm itself requires very few paramters to be fine tuned. The empirical results from our data simulations demonstrated that the MSSAPS is a very robust investment strategy that outperforms existing state-of-the-art PS algorithms.

A significant constribution we made to the growing literature on Online learning for PS came from Chapter 8. We presented a new Online approach to PS algorithm called the SVMPS. Most existing state-of-the-art PS algorithms implicitely assume that portfolio managers only face one important investment decision, namely the decision related to efficient resource allocation, also called portfolio construction. In truth, all investment managers consider the stock selection problem as important as the portfolio construction. the SVMPS in our view is the first and only algorithm we know of that tries to combine both stock selection and portfolio construction in the design of Online PS algorithms. Building on multifactor linear beta models and SVM, SVMPS produces very stable portfolios with consistent returns that substantially outperform existing state-of-the-art algorithms on real-market datasets after accounting for reasonable brokerage commissions. To arrive at these portfolios, the SVMPS treats the stock selection and portfolio construction problems as two distinct but equally important problems. The solution proposed in this chapter is not only novel, but it is also very flexible and allows the portfolio manager to have full control on how to to select stock with out-of-sample bounds on the number of mistakes.

9.2 DIRECTIONS FOR FUTURE RESEARCH

In the light of what emerged in the study and the development of novel algorithms presented in this thesis, we think there are unanswered questions that could form the basis fo future research.

9.2.1 Volatility Clustering and Pattern Selection

In the Pattern-Matching based PS algorithms (DCEPS and MSSAPS), efficiently recognising patterns in a portfolio of stock prices is one of the most important and very challenging undertakings. One area of research could be to exploit the idea of volatility clustering and search for patterns in similar volatility regimes. There are a number of models that capture this tendency. Some of the better known are the ARCH (Engle, (1982)), GARCH (Bollerslev, (1986)) and EGARCH (exponential generalised autoregressive conditional heterskedastic) models by Nelson (1991)). Essentially, this ARCH family of models sets the volatility in such a way that volatile markets tend to have more volatility in their future and quiet markets tend to stay quiet (see Mandelbrot (1963)) until some random volatility shock hits the market. It is therefore likely that Pattern-Matching in the volatility space could be less noisy and provide better stock portfolio predictions than the traditional ones derived from stock market returns only.

9.2.2 Stock Market Regime Shifts and Pattern Filtering

Financial markets often change their behaviour abruptly and more often than not these changed behaviours of asset prices persist for many periods. Regime-switching models can capture these sudden changes of behaviour. Models with regimes shifts have been used to characterise bull and bear markets or calm versus turbulent markets. This pattern has been found since the earliest studies of regime switches on equity returns (Hamilton and Susmel (1994)), who found that for excess equity returns, there is a high volatility regime that has, on average, low returns. This regime naturally corresponds to bear markets. Therefore, alpha signals can be generated more effectively by developing models that are adapted to take account of different market regimes.

It is therefore reasonable to expect that if the market portfolio exhibits regime switches, then portfolios of stocks would also switch regimes and the regimes and behaviour within each regime of the portfolios should be related across portfolios (see Perez-Quiros and Timmermann (2000) and Gu (2005)). One could exploit these regime changes in Online learning algorithms for PS in a simple and intuitive way. We could start by modelling the market portfolio to infer its current state and use that information to create today's portfolio vector from similar historical states. The main advantage is that if the patterns are obtained from similar historical regimes, the response is likely to be estimated more precisely. We expect this state dependent Online learning for PS strategies to generate better performance.

9.2.3 Directional Predictions and Online Learning for PS Algorithms

Most of the studies in stock price forecasting have focused more on the accurate forecasting of the value of stock returns as this genenrally is one of the inputs into the mean variance analysis. However, some recent studies have suggested that trading strategies derived from the forecasts of the direction of stock market change may be more effective and generate higher profit. Relevant research on this topic includes Breen, Glosten and Jaganathan (1989), Leitch and Tanner (1991), Wagner, Shellans and Paul (1992), Pesaran and Timmerman (1995), Kuan and Liu (1995), Larsen and Wozniak (1995), Womack (1996), Gencay (1998), Leung Daouk and Chen (1999), Elliott and Ito (1999) White (2000), Pesaran and Timmerman (2001), and Cheung, Chinn and Pascual (2003). Leung et al. (2000) found that the classification models based on the directional forecast of stock returns outperform those based on the level of stock returns in term of both predictability and profitability.

As argued earlier, the particular problem addressed by traditional machine-learning-based PS strategies is the construction of efficient distribution of wealth among several assets with the view of achieving maximal growth in wealth. Almost all existing prediction schemes focus solely on price relative and ignore other useful side information, such as general stock market directional forecast. Cover and Ordentlich (1996) presented a model to incorporate side information in a universal portfolio setting. One direction for future research will be therefore to combine Online PS algorithms with additional information provided by the directional forecast of the general stock market. One way to achieve this goal would be to derive the optimal asset weights subject to the stock market directional forecast. More generally, what is missing in this thesis is an analytical model that better explains why our active trading strategies are so successful.

9.3 GENERAL CONCLUSION

Any report of abnormal returns using historical markets has to overcome the suspicion of "data snooping". In other words, when a datasets is excessively mined by testing many strategies there is a substantial chance that one of the strategies will be successful by simple overfitting. In this thesis, we have largely mitigated the risk of such "data snooping" by applying the algorithms on a larger set of data accross different geographical locations. Another mitigating factor is the sensitivity of the proposed algorithms to some parameter choices. In general, our algorithms have at least one parameter (the maximal window sise W) to be set by the fund manager before trading is executed. All the experiments conducted here have indicated that the algorithm's performance is robust with respect to the applicable parameters, using the the expert's combination paradigm.

Another data-snooping hazard is related to the so-called survivorship bias. The idea here is that stocks selected for the various market experiments are all known to have survived for the full period under consideration, including the dotcom buble of the early 2000 and the financial crisis of 2008. Although we acknowledge this as a potential weakness in any historical backtest analyis, we are also quick to mention that our algorithms were fully developed using only the NYSE(O) and NYSE(N) and the application to all remaining datasets was obtained after the algorithms were fixed.

In conclusion we argue that the impressive performance generated by our PS algorithm using data that are widely available casts some doubt on the market efficiency hypothesis at least in its weak form specification. Only in the presence of weakly inefficient markets can these algorithms give the exceptional performance that we have demonstrated in these examples. While consistently beating the market is considered to be already a great challenge, our approach to PS has clearly indicated that beating the best stock is an achievable goal. However, it is important to point that the algorithms presented here are intended to show investment decisions that would have been made had a strategy been utilised in the past. Without adhering to best practices as presented here, the algorithms can also lead to misleading results.

BIBLIOGRAPHY

[1] Admati, A. R., S. Bhattacharya, P. Pfleiderer, and S. A. Ross 1986, 'On timing and selectivity'. Journal of Finance 41, 715–730.

[2] Agarwal, A, Hazan, E, kale, S and Schapire, R.E, 2006, Algorithms for portfolio management based on the newton method. In Proceedings of the International Conference on machine-learning (ICML 2006), pages 9–16.

[3] Algoet, P., 1992, Universal schemes for prediction, gambling, and PS. Annals of Probability 20 901–941.

[4] Algoet, P., and Cover, T, 1988, Asymptotic Optimality Asymptotic Equipartition Properties of Log-Optimum Investments, Ann. Prob. 16, 876–898.

[5] Ammermann, P. and Patterson, D. 2003. The cross-sectional and cross-temporal universality of nonlinear serial dependencies: Evidence from world stock indices and the Taiwan stock exchange. Pacific-Basin Finance Journal, 11 (2), 175-195.

[6] Bak, P. and Chen, K., 1991, Self-organised criticality. Scientific American 264, 26-33.

[7] Balvers, R. J, Wu, Y., 2006, "Momentum and mean reversion across national equity markets", Journal Empirical Finance, Vol.13, No. 1, pp.24-48.

[8] Beine, M.; Capelle-Blancard, G. and Raymond H. 2008. International nonlinear causality between stock markets, The European Journal of Finance, 14 (8), 663-686.

[9] Blum, A. and Kalai, A, 1997, Universal portfolios with and without transaction costs. In Proceedings of the 10th Annual Conference on Learning Theory.

[10] Blum, A. and Mansour, Y. 2007. From external to internal regret, Journal of machine-learning Research 8: 1307–1324.

[11] Bollerslev, T, 1986, Generalised Autoregressive Conditional Heteroscedasticity. Journal of Econometrics, 31, 307–27.

[12] Borodin, A. and El-Yaniv, R. 1998. Online computation and competitive analysis, Cambridge University Press, Cambridge.

[13] Borodin, A., El-Yaniv, R. and Gogan, V. 2000. On the competitive theory and practice of PS (extended abstract), Proceedings of the Latin American Symposium on Theoretical Informatics, pp. 173–196.

[14] Borodin, A., El-Yaniv, R., AND Gogan, V. 2004. Can we learn to beat the best stock. Journal of Artificial Intelligence Research 21, 579–594.

[15] Breen, W, Glosten, L.R, and Jagannathan, R., 1989, Economic significance of predictable variations in stock index returns, Journal of Finance 44, 1177-1189.

[16] Breiman,L., 1961, Optimal gambling systems for favorable games, Fourth Berkeley Symposium, vol. I, 65-78.

[17] Brock, W. A., Hsieh, D. A., LeBaron, B., 1991, nonlinear dynamics, chaos and instability: Statistical theory and Economic evidence. MIT Press, Massachusetts, MA.

[18] Broomhead, D. S. & King, G. P. 1986 Extracting qualitative dynamics from experimental data, Physica D, 20, pp. 217–236.

[19] Burges, C., 1998, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery, vol. 2, pp. 121-167.

[20] Cao, L. and Tay, T., 2001, Financial forecasting using support vector machines, Neural Computing and Applications, vol. 10, pp. 184-192.

[21] Cao, L. and Tay, T., 2003, Support vector machine with adaptive parameters in financial time series forecasting, IEEE Trans. on Neural Networks, vol. 14, pp. 1506-1518.

[22] Casdagli, M., Eubank, S., Farmer, J. D., Gibson, J., 1991, State space reconstruction in the presence of noise. Physica D 51, 52-98.

[23] Chen, N., Roll, R., and Ross., S., 1986, Economic Forces and the Stock Market, Journal of Business, 59, pp. 386–403.

[24] Cheung, Y-W, Chinn, M.D and Pascual, A.G., 2003, Empirical exchange rate models of the nineties: are any fit to survive?, NBER Working thesis 9393.

[25] Chu, T.C. Tsao, C.T. Shiue, Y.R., 1996, Application of Fuzzy Multiple Attribute Decision Making on Company Analysis for Stock Selection. Proceedings of Soft Computing in Intelligent Systems and Information Processing 509-514

[26] Connor, G., and Korajczyk, R., 1986, Performance Measurement with the Arbitrage Pricing Theory: A New Framework for Analysis, Journal of Financial Economics, 15, No. 3, pp. 373–394.

[27] Connor, Gregory., 1984, A Unified Beta Pricing Theory, Journal of Economic Theory, 34, No. 3, pp. 13–31.

[28] Conrad, J. Kaul, G. 1998, "An anatomy of trading strategies". Review of Financial studies, Vol. 11, pp. 489-519.

[29] Cover, T. M. 1991. Universal portfolios. Mathematical Finance 1, 1, 1–29.

[30] Cover, T.M. and Ordentlich, E., 1996, Universal portfolios with side information. IEEE Transactions on Information Theory, 42(2).

[31] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., AND Singer, Y. 2006. Online passive-aggressive algorithms. Journal of machine-learning Research 7, 551–585.

[32] Crammer, K., Dredze, M., AND Kulesza, A. 2009. Multi-class confidence weighted algorithms. In Proceedings of the

[33] Crammer, K., Dredze, M., AND Pereira, F. 2008. Exact convex confidence-weighted learning. In Proceedings of the Annual Conference on Neural Information Processing Systems. 345–352.

[34] Diks, C., 1999, nonlinear Time Series Analysis: Methods and applications.World Scientific, Singapore.

[35] Dredze, M., Crammer, K., AND Pereira, F. 2008. Confidence-weighted linear classification. In Proceedings of the International Conference on machine-learning. 264–271.

[36] Dredze, M., Kulesza, A., AND Crammer, K. 2010. Multi-domain learning by confidence-weighted parameter combination. machine-learning 79, 1-2, 123–149.

[37] Elliott, G and Ito, T., 1999, Heterogeneous expectations and tests of efficiency in the Yen/Dollar forward foreign exchange market, Journal of Monetary Economics 43, 435-456.

[38] Engle, R, 1982, Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica, 50, 987–1007

[39] Evgeniou, T., Pontil, M. and Poggio, T., 2000, Regularisation networks and support vector machines, Advances in Computational Mathematics, vol. 13, pp. 1-50.

[40] Fama, E. 1972, 'Components of investment performance'. Journal of Finance 27, 551–567.

[41] Fama, E. and French, K., 1993, The Cross Section of Expected Stock Returns, Journal of Finance, 47, No. 2, pp. 427–466.

[42] Fama, E. F., and MacBeth, J. D., 1973, Risk, return and equilibrium: empirical tests, Journal of Political Economy 81, 607-636.

[43] Fan, A., Palaniswami, M., 2001, Stock Selection Using Support Vector Machines. Proceedings of International Joint Conference on Neural Networks 3 1793-1798

[44] Fan, A., Palaniswami, M., 2001. Stock selection using support vector machines. Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on, Vol. 3, IEEE, pp. 1793–1798. Washington, DC.

[45] Fernholz, E.R. (2002) Stochastic Portfolio Theory. Springer-Verlag, New York

[46] Fraedrich, K. 1986 Estimating the dimension of weather and climate attractors Journal of Atmospheric Science, 43, (5) 419-432.

[47] Gencay, R, 1998, Optimisation of technical trading strategies and the profitability in security markets, Economics Letters 59, 249-254.

[48] Ghil, M., M. R. Allen, M. D. Dettinger, K. Ide, D. Kondrashov, M. E. Mann, A. W.Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou 2002: "Advanced spectralmethods for climatic time series," Reviews of Geophysics, 40(1), 1–41.

[49] Gibbons, M., 1982, Multivariate tests of nancial models: a new approach, Journal of Financial Economics 10, 3-27.

[50] Golyandina, N., Nekrutkin, V., and Zhigljavsky, A. 2001. Analysis of Time Series Structure: SSA and related techniques, Chapman & Hall/CRC, New York – London.

[51] Gu, L., 2005, Asymmetric Risk Loadings in the Cross Section of Stock Returns, SSRN working thesis.

[52] Guidolin, M., and A. Timmermann, 2008b, Sise and Value Anomalies under Regime Shifts. Journal of Financial Econometrics, 6, 1-48.

[53] Gyorfi, L. and Schafer, D. 2003. Nonparametric prediction. In Advances in Learning Theory: Methods, Models and Applications,

[54] Gyorfi, L. and Vajda, I. 2008. Growth optimal investment with transaction costs. In Proceedings of the Internationa Conference on Algorithmic Learning Theory. 108–122.

[55] Gyorfi, L. and Walk, H. 2012. Empirical PS strategies with proportional transaction costs. IEEE Transactions on Information Theory 58, 10, 6320 – 6331.

[56] Gyorfi, L., Lugosi, G., and Udina, F. 2006. Nonparametric kernel-based sequential investment strategies. Mathematical Finance 16, 2, 337–357.

[57] Gyorfi, L., Udina, F., and Walk, H. 2008. Nonparametric nearest neighbor based empirical PS strategies. Statistics and Decisions 26, 2, 145–157.

[58] Gyorfi, L., Urban, A., and Vajda, I. 2007. Kernel-based semi-log-optimal empirical PS strategies. International Journal of Theoretical and Applied Finance 10, 3, 505–516.

[59] Hamilton, J.D., and R. Susmel, 1994, Autoregressive Conditional Heteroskedasticity and Changes in Regime. Journal of Econometrics, 64, 307-333.

[60] Hassani, H. 2007. Singular Spectrum Analysis: Methodology and Comparison. Journal of Data Science. 5, 239-257.

[61] Hassani, H.; Dionisio, A., Ghodsi, M. 2010b, "The effect of noise reduction in measuring the linear and nonlinear

[62] Hassani, H; Heravi, H; and Zhigljavsky, A. 2009. Forecasting European Industrial Production with Singular Spectrum Analysis, InternationalJjournal of Forecasting, 25, 103-118.

[63] Hassani, H; Soofi, A; and Zhigljavsky, A. 2010a. Predicting Daily Exchange Rate with Singular Spectrum Analysis., nonlinear Analysis: Real World Applications, 11, 2023-2034.

[64] Hassani, H; Thomakos, D. 2010. A Review on Singular Spectrum Analysis for Economic and Financial Time Series, Statistics and Its Interface, Forthcoming

[65] Hazan, E. 2006. Efficient Algorithms For Online Convex Optimisation And Their Applications, PhD thesis, Princeton University.

[66] Helmbold, D. P., Schapire, R. E., Singer, Y. and Warmuth, M. K. 1996. Online PS using multiplicative updates, Proceedings of the International Conference on machine-learning, pp. 243–251.

[67] Helmbold, D., Schapire, R., Singer, Y., & Warmuth, M. 1998. PS using multiplicative updates. Mathematical Finance, 8(4), 325–347.

[68] Hsieh, D. A., Chaos and nonlinear dynamics: Application to financial markets, 1991, The Journal of Finance 46, 1839-1887

[69] Huang, W, Y. Nakamori, Y, and Wang, S.Y., 2005,Forecasting stock market movement direction with support vector machine, Computers & Operations Research, vol. 32, pp. 2513-2522.

[70] Huang, W., Nakamori, Y., Wang, S.-Y., 2005. Forecasting stock market movement direction with support vector machine. Comput. Oper. Res. 32 (10), 2513–2522.

[71] Ince, H., and Trafalis, T.B. 2004, "Kernel principal component analysis and support vector machines for stock price prediction", Proceedings of the 2004 IEEE International Joint Conference on Neural Networks, 3: 2053-2058.

[72] Jagadeesh, N., and Titman, S.,1993,Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency Journal of Finance,Volue 8, Issue 1, 65-91.

[73] Jagannathan, R., and Wang, Z., 2002, Empirical evaluation of asset pricing models: a comparison of the SDF and beta methods, Journal of Finance 57, forthcoming.

[74] Jagannathan, R., Schaumburg, E., and Zhou, G., 2010, "Cross-Sectional Asset Pricing Tests," Annual Review of Financial Economics, 2, 49–74.

[75] Jensen, M. C. 1968, 'The performance of mutual funds in the period 1945-1964'. Journal of Finance 23, 389–416.

[76] Johnson, N. F., Jefferies, P. & Ming Hui, P. 2003. Financial Market Complexity, Oxford University Press.

[77] Kalai, A. and Vempala, S, 2005, Efficient algorithms for Online optimisation. Journal of Computer and System Sciences, 713:291–307.

[78] Kandel, S., 1984, The likelihood ratio test statistic of mean-variance e ciency without a riskless asset, Journal of Financial Economics 13, 575-592.

[79] Kantz, H., and Schreiber, T., 1997, nonlinear Time Series Analysis, Cambridge University Press, Cambridge.

[80] Kelly, J. L. Jr, 1956, A new interpretation of information rate, Bell System Technical Journal, 35, pp. 917-926.

[81] Kennel, M.B. and Abarbanel, H.D.I., 2002, False neighbors and false strands: A reliable minimum embedding dimension algorithm, Phys. Rev. E., 66, pp. 1 – 18.

[82] Kennel, M.B., Brown, R., and Abarbanel, H.D.I., 1992, Determining embedding dimension for phase-space reconstruction using a geometrical construction, Phys. Rev. A, 45, pp. 3403 – 3411.

[83] Kim, K., 2003. Financial time series forecasting using support vector machines. Neurocomputing 55 (1), 307–319.

[84] Korn, R, 1997, Optimal portfolios : stochastic models for optimal investment and risk management in continuous time, World Sientific.

[85] Kot, M., Schaffer, W., Truty, G., Graser, D., Olsen, L. 1988. Changing criteria for imposing order. Ecological modelling 43: 75-110.

[86] Kuan, CM, and Liu, T., 1995, Forecasting exchange rates using feed-forward and recurrent neural networks, Journal of Applied Econometrics 10, 347-64.

[87] Larsen, G A.Jr, and Wozniak, G.D., 1995, market-timing can work in the real world, Journal of Portfolio Management 21, 74-81.

[88] Latane, H., 1959, Criteria for choice among risky ventures, Journal of Political Economy, 67, pp. 144-155.

[89] Lehmann, B., and Modest, D., 1988, The Empirical Foundations of the Arbitrage Pricing Theory I: The Empirical Tests, Journal of Financial Economics, 21, pp. 213–254.

[90] Leitch, G, and Tanner, J.E, 1991, Economic forecast evaluation: profits versus the conventional error measures, American Economic Review 81, 580-590.

[91] Leland, H, 1999, 'Performance beyond mean-variance: Performance measurement in a nonsymmetric world'. Financial Analysts Journal 55, 27–36.

[92] Leung, M T., Daouk, H.and Chen, AS., 1999, Forecasting stock indices: a comparison of classification and level estimation models, working thesis, Indiana University.

[93] Leung, M T., Daouk, H.and Chen, AS., 2000, Forecasting stock indices: a comparison of classification and level estimation models, International Journal of Forecasting, vol. 16(2), 73-190.

[94] Levin, A.U., 1995, Stock Selection via nonlinear Multi-factor Models. Advances in Neural Information Processing Systems 966-972

[95] Levina, T. and Shafer, G. 2008. PS and online learning, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 16(4): 437–473.

[96] Li, B, Hoi, S. C. H. and Gopalkrishnan, V, 2011,. Corn: correlation-driven nonparametric learning approach for PS. ACM TIST, 2(3):21:1–21:29.

[97] Li, B, Hoi, S. C. H. and Gopalkrishnan, V, 2013. Confidence weighted mean reversion strategy for online PS. In ACM Transactions on Knowledge Discovery from Data. Vol. 7. 4:1–4:38.

[98] Li, B, Hoi, S. C. H. and Gopalkrishnan, V. 2012. Pamr: Passive aggressive mean reversion strategy for PS. machine-learning 87, 2, 221–258.

[99] Lintner, J., 1965, The Valuation of Risk Assets and the Selection of Risky Investments in Stock Portfolios and Capital Budgets, Review of Economics and Statistics, 47, pp. 13–37.

[100] Lorenz, H. W., 1993, nonlinear dynamical economics and chaotic motion. Springer, New York.

[101] Luenberger, D. G., 1998, Investment Science, Oxford University Press.

[102] MacKinlay, A. C., and Richardson, M. P., 1991, Using generalised method of moments to test mean-variance e ciency, Journal of Finance 46, 551-527.

[103] Malkiel, B., 1990, A random walk down Wall Street, Norton, New York.

[104] Mandelbrot, B. B, 1963, The Variation of Certain Speculative Prices, The Journal of Business 36, No. 4, 394-419

[105] Mandelbrot, B. B., 1998, Fractals and Scaling in Finance: Discontinuity, Concentration, Risk. Springer, New York.

[106] Mantegna, R. N. & Stanley, H. E., 2000. An introduction to Econophysics. Cambridge University Press.

[107] Mantegna, R. N. and Stanley, H.E., 1995, Scaling behaviour in the dynamics of an economic index. Nature 376, 46-49.

[108] Mantegna, R. N. and Stanley, H.E., 1996, Turbulence and financial markets. Nature 376, 46-49.

[109] Markowitz, H. 1952. PS. The Journal of Finance 7, 1, 77–91.

[110] Mees, A.I., Rapp, P.E., and Jennings, L.S., 1987, Singular-value decomposition and embedding dimension, Phys. Rev. A, 36(1), pp. 340 –346

[111] Meese, Richard A., and Kenneth S. Rogoff. 1983. Empirical exchange rate models of the seventies: Are any fit to survive? Journal of International Economics 14:3-24.

[112] Merton, R.C., 1973, An Intertemporal Capital Asset Pricing Model, Econometrica, 41, pp. 867–887.

[113] Nelson, D. B. 1991, 'Conditional heteroskedasticity in asset returns: A new approach', Econometrica 59, 347—370.

[114] Novikov, A., 1962, On convergence proofs for perceptrons. In Proceedings of the Symposium of the Mathematical Theory of Automata, volume 12, pages 615-622.

[115] Ordentlich, E. and Cover, T. M. 1996. Online PS, Proceedings of the Annual Conference on Learning Theory.

[116] Ormerod, P. and Campbell, M. 1997 Predictability and economic time series In System dynamics in economic and financial models(Eds, Heij, C., Schuacher, J. m., Hanzon, B. and Praagman, C.) John Wiley.

[117] Osborne, M. F.M., 1959, Brownian motion in the stock market. Oper. Res. 7, 145-173.

[118] Packard, N., Crutchfield, J., Farmer, D. and Shaw, R., 1981, Geometry from a time series. Phys. Rev. Lett. 45, 712-715.

[119] Perez-Quiros, G. and A. Timmermann, 2000, Firm Sise and Cyclical Variations in Stock Returns. Journal of Finance, 55, 1229-1262.

[120] Pesaran M.H., and A. Timmerman 1995: Predictability of Stock Retruns: Robustness and Economic Signi. . . cance, Journal of Finance, 50, 1201-1228.

[121] Pesaran, M. H, and Timmermann, A.G., 2001, How costly is it to ignore breaks when forecasting the direction of a time series?, Manuscript, Cambridge University and UCSD.

[122] Peters, E. E., 1996. Chaos and Order in the Capital Markets: a New View of Cycles, Prices and Volatility, 2nd Edition, Wiley, New York.

[123] Puja, D. and Arindam, B, 2011, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining Pages 1163-1171

[124] Rahman Khan, M, A.; and Poskitt, D.S. 2010. Description Length Based Signal Detection in Singular Spectrum Analysis. Working thesis. Department of Econometrics and Business Statistics, Monash University. Australia.

[125] Rosenblatt, F (1957), The Perceptron–a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

[126] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65, 386–407. (Reprinted in Neurocomputing (MIT Press, 1988).).

[127] Ross, S.A., 1976, The Arbitrage Theory of Capital Asset Pricing, Journal of Economic Theory, 13, pp. 341–360.

[128] Ruei-Shan, L., 2008. A study on application of smooth support vector classification to stock selection in taiwan's stock market. Joint Conference of WEHIA & CIEF. Taoyuan, Taiwan.

[129] Schaffer, W., Kot, M. 1985. Nearly one dimensional dynamics in an epidemic. J. Theor. Biol. 112: 403-427.

[130] Sewell, M., 2010. The Application of Intelligent Systems to Financial Time Series Analysis, PhD thesis, PhD thesis, Department of Computer Science, University College London, University of London.

[131] Sharpe,W., 1964, Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk, Journal of Finance, 19, pp. 425–442.

[132] Shlesinger, M. F., Zaslavsky, G. M., and Klafter, J., 1993, Strange kinetics. Nature 363, 31-37.

[133] Strozzi, F, Tenrreiro, EG, Noè, C, Rossi, T, Serati M, Comenges, JMZ, 2007, Applicatoin of nonlinear time series analysis techniques to nordic spot electricity market data, Liuc thesiss n. 200, Serie Tecnologia 11, marzo 2007.

[134] T. Evgeniou, T., Pontil, M., and Poggio, T., 2000, Regularisation networks and support vector machines, Advances in Computational Mathematics, vol. 13, pp. 1-50.

[135] Takens, F. 1980. Detecting strange attractors in turbulence. In: Rand, D., Young, L. (eds). Dynamical Systems and Turbulence (pp. 366-381). New York: Springer.

[136] Takens, F., 1981, in Dynamical Systems and Turbulence, Warwick 1980, vol. 898 of Lecture Notes in Mathematics, edited by A. Rand and L.S Young, Springer, Berlin, pp. 366-381.

[137] Takens, F., 1996, The effect of small noise on systems with chaotic dynamics. In Stochastic and Spatial Structures of Dynamical Systems, S. J. van Strien and S. M. Verduyn Lunel, Verhandelingen KNAW, Afd. Natuurkunde, vol. 45, pp. 3-15. North-Holland, Amsterdam.

[138] Tay, F., Cao, L., 2001. Application of support vector machines in financial time series forecasting. Omega 29 (4), 309–317.

[139] Tay, F., Cao, L., 2002. Modified support vector machines in financial time series forecasting. Neurocomputing 48 (1), 847–861.

[140] Theil, H. 1971. Principles of Econometrics. John Wiley & Sons, New York.

[141] Treynor, J. and K. Mazuy, 1966, 'Can mutual funds outguess the market?'. Harvard Business Review 44, 131–136.

[142] Van Gestel, T., Suykens, J., Baestaens, D., Lambrechts, A., Lanckriet, G., Vandaele, B., et al., 2001. Financial time series prediction using least squares support vector machines within the evidence framework. Neural Netw., IEEE Trans. 12 (4), 809–821.

[143] Vapnik, V.N 1995. The Nature of Statistical Learning Theory. Springer, New York.

[144] Vapnik, V.N.1998, Statistical Learning Theory, New York: Wiley.

[145] Wagner, J, Shellans, S. and Paul, R., 1992, market-timing works where it matters most: in the real world, Journal of Portfolio Management 18, 86-90.

[146] White, H., 2000, A reality check for data snooping, Econometrica 68, 1097-1126.

[147] Williams, G. 1997. Chaos Theory Tamed (Chapter 19: Attractor Reconstruction). Washington D.C.: Joseph Henry Press.

[148] Womack, K.L., 1996, Do brokerage analysts' recommendations have investment value?, Journal of Finance 51, 137-167.

[149] Wu, Y., and Zhang, H., 1997, Forward premiums as unbiased predictors of future currency depreciation: A non-parametric analysis. Journal of International Money and Finance, 16:609–623.

[150] Yang, H., Chan, L., and King, I. 2002, Support Vector Machine Regression for Volatile Stock Market Prediction, Third International Conference on Intelligent Data Engineering and Automated Learning, 391-396.

[151] Zargham, M.R., Sayeh, M.R., 1999, A Web-Based Information System for Stock Selection and Evaluation. Proceedings of the First International Workshop on Advance Issues of ECommerce and Web-Based Information Systems 81-83

.0 STATISTICAL PROPERTIES OF RATIO OF RANDOM VARIABLES

Consider two random variables R and S where S has support $[0,\infty[$. Let $G = g\left(R,S\right) = \frac{R}{S}$, our goal is to find an approximation for $E\left(G\right)$ and $Var\left(G\right)$ using the Taylor expansion around $g\left(.\right)$.

For any function $f\left(x,y\right)$ the bivariate first order Taylor expansion about θ is

$$f\left(x,y\right) = f\left(\theta\right) + f_x^{'}\left(\theta\right)\left(x-\theta_x\right) + f_y^{'}\left(\theta\right)\left(y-\theta_y\right) + remainder$$

Let $\theta = \left(EX,EY\right)$, the simplest approximation for $E\left(f\left(X,Y\right)\right)$ is therefore

$$E\left(f\left(x,y\right)\right) = f\left(\theta\right) + f_x^{'}\left(\theta\right)\left(0\right) + f_y^{'}\left(\theta\right)\left(0\right) + O\left(n^{-1}\right) \approx f\left(EX,EY\right)$$

The second order Taylor expansion is

$$\begin{aligned} f\left(x,y\right) &= f\left(\theta\right) + f_x^{'}\left(\theta\right)\left(x-\theta_x\right) + f_y^{'}\left(\theta\right)\left(y-\theta_y\right) \\ &+ \left\{f_{xx}^{''}\left(\theta\left(x-\theta_x\right)^2 + 2f_{xy}^{''}\left(x-\theta_x\right)\left(y-\theta_y\right) + f_{yy}^{''}\left(\theta\right)\left(y-\theta_y\right)^2\right)\right\} + residual \end{aligned}$$

So a better approximation is given by

$$E\left(f\left(X,Y\right)\right) = f\left(\theta\right) + \left\{f_{xx}^{''}\left(\theta\right)Var\left(X\right) + 2f_{x\mathrm{y}}^{''}\left(\theta\right)Cov\left(X,Y\right) + f_{yy}^{''}\left(\theta\right)Var\left(Y\right)\right\} + residual$$

For

$$g = \frac{R}{S}, g_{RR}^{''} = 0, g_{RS}^{''} = -S^{-2}, g_{SS}^{''} = \frac{2R}{S^3}$$

Then $E\left(\frac{R}{S}\right)$ is approximately

$$E\left(\frac{R}{S}\right) \approxeq E\left(g\left(R,S\right)\right) \simeq \frac{ER}{ES} - \frac{Cov\left(R,S\right)}{E^2S} + \frac{Var\left(S\right)ER}{E^3S}$$

Using the first order Taylor expansion, the variance is

$$\begin{aligned} Var\left(f\left(X,Y\right)\right) \quad & = E\left\{\left[f\left(X,Y\right) - E\left(f\left(X,Y\right)\right)\right]^2\right\} \\ & \simeq E\left\{\left[f\left(X,Y\right) - f\left(EX,EY\right)\right]^2\right\} \\ & = E\left\{\left[f_x'\left(\theta\right)\left(X-\theta_x\right) - f_y'\left(\theta\right)\left(Y-\theta_y\right)\right]^2\right\} + O\left(n^{-1}\right) \\ & \simeq f_x'\left(\theta\right)^2 Var\left(X\right) + 2f_x'\left(\theta\right)f_y'\left(\theta\right)Cov\left(X,Y\right) + f_y'\left(\theta\right)^2 Var\left(Y\right) \end{aligned}$$

Since

$$g_R^{'} = S^{-1}, g_S^{'} = -\frac{R}{S^2}$$