OXFORD BROOKES UNIVERSITY

FACULTY OF TECHNOLOGY, DESIGN AND ENVIRONMENT

THE DEPARTMENT OF MECHANICAL ENGINEERING AND MATHEMATICAL SCIENCES

A SYSTEM TO PREDICT THE S&P 500 USING A BIO-INSPIRED ALGORITHM

ANDREW J. REGAN

A thesis submitted for the degree of *Doctor of Philosophy*

October 2014

Abstract

The goal of this research was to develop an algorithmic system capable of predicting the directional trend of the S&P 500 financial index. The approach I have taken was inspired by the biology of the human retina. Extensive research has been published attempting to predict different financial markets using historical data, testing on an in-sample and trend basis with many employing sophisticated mathematical techniques. In reviewing and evaluating these in-sample methodologies, it became evident that this approach was unable to achieve sufficiently reliable prediction performance for commercial exploitation. For these reasons, I moved to an out-of-sample strategy and am able to predict tomorrow's ***(t+1)*** directional trend of the S&P 500 at 55.1%.

The key elements that underpin my bio-inspired out-of-sample system are:

1. Identification of 51 financial market data (FMD) inputs, including other indices, currency pairs, swap rates, that affect the 500 component companies of the S&P 500.
2. The use of an extensive historical data set, comprising the actual daily closing prices of the chosen 51 FMD inputs and S&P 500.
3. The ability to compute this large data set in a time frame of less than 24 hours.

The data set was fed into a linear regression algorithm to determine the predicted value of tomorrow's (t+1) S&P 500 closing price. This process was initially carried out in MatLab which proved the concept of my approach, but (3) above was not met. In order to successfully meet the requirement of handling such a large data set to complete the prediction target on time, I decided to adopt a novel graphics processing unit (GPU) based computational architecture. Through extensive optimisation of my GPU engine, I was able to achieve a sufficient speed up of 150x to meet (3).

In achieving my optimum directional trend of 55.1%, an extensive range of tests exploring a number of trade offs were carried out using an 8 year data set. The results I have obtained will form the basis of a commercial investment fund.

It should be noted that my algorithm uses financial data of the past 60-days, and as such would not be able to predict rapid market changes such as a stock market crash.

Acknowledgements

Firstly, I would like to thank Professor Chris Toumazou of Imperial College London for suggesting that I should consider doing a PhD.

Secondly, I would like to thank Oxford Brookes University for enabling me to register for a PhD there, accepting my experience in finance in lieu of not having a first degree. My supervisory team of Professors Chris Toumazou, Khaled Hayatleh and especially John Lidgey deserve my thanks for their enduring inspiration, support and confidence in me throughout. A further special mention goes to Pantelis Georgiou and Konstantin Nikolik of Imperial College for their helpful insight.

Also thanks are due to Mark Betteridge and his team at Preciousbluedot [1] for introducing me to the power of Graphics Processing Units and assisting me in producing the hardware needed for my algorithm. I am very grateful to all my colleagues at Corvus Capital [2] for indulging me during my research, in particular Jack Dibble for his extensive efforts in supporting me throughout my PhD.

Finally, I would like to thank my fiancée, Missy M^{c}Kee and my amazing children for their support and patience; everyone has been extremely tolerant and helpful throughout my research studentship.

[1] Preciousbluedot Ltd. http://www.preciousbluedot.com (contact@preciousbluedot.com)
[2] Corvus Capital Ltd. http://www.corvus.com (ar@corvus.com)

Table of Contents

Abstract.......... **i**

Acknowledgements.......... **ii**

Table of Contents.......... **iii**

List of Figures and Tables.......... **vi**

Symbols, Acronyms and Abbreviations.......... **viii**

Definitions.......... **x**

Chapter 1.......... **1**

2.1 Introduction and Explanation of S&P 500.......... 4

2.2 In-sample and out-of-sample forecasting.......... 9

2.3 Current forecasting methods.......... 10

2.3.1 ANN Review.......... 10

2.3.2 SVMs: description and review.......... 12

2.4 Review of ANNs and SVMs.......... 14

Chapter 3.......... **32**

3.1 Introducing a novel bio-inspired approach to financial forecasting.......... 33

3.2 Initial concept for the proposed RPA.......... 33

3.2.1 Selection criteria for the chosen FMD inputs.......... 34

3.2.3 Historical data.......... 41

3.3 Architecture of the proposed RPA.......... 41

3.3.1 Potential implementation challenges.......... 42

3.4 Concept – Explanation of RPA architecture.......... 45

3.5 Proof of concept.......... 50

3.5.1 MatLab explanation.......... 51

3.5.2 Implementation of RPA using MatLab.......... 52

3.5.3 Initial implementation and 'proof of concept' of the RPA using MatLab.......... 53

3.5.4 Further implementation and feasibility of RPA (using MatLab).......... 56

3.6 Implementation of model using FPGA.......... 63

Chapter 4.......... **64**

4.1 Developments in parallel computing.......... 64

4.2 Advancing the implementation method for the RPA.......... 65

4.2.1 A solution to test a greater number of FMD inputs to establish an optimum.......... 66

4.3 Graphics Processing Unit (GPU) 66
4.3.1 How a GPU differs from a CPU 68
4.3.2 Unique approach: benefits of using GPUs in financial prediction 69
4.3.3 Specifications of GPU system used to implement the RPA 70
4.4 Computational implementation and optimisation of the RPA onto GPU 71
4.4.1 Step 1: Optimise existing C++ code on CPU (No GPU acceleration) 72
4.4.2 Step 2: Initial C++ AMP implementation to run on a GPU 72
4.4.3 Step 3: Unroll the algorithm to feed the data sympathetically to the GPU 74
4.4.4 Step 4: Re-engineer the algorithm to use the smallest dataset possible 75
4.5 Effectiveness in implementing the RPA onto the GPU system 75
4.6 Early results 76
Chapter 5 79
5.1 Importance in establishing the optimum number of FMD inputs to successfully predict the S&P 500 at (t+1) 79
5.1.1 Results: Optimum number of FMD inputs for the RPA 80
5.2 Significance of each set of chosen FMD inputs 86
Chapter 6 90
6.1 Exploring factors that can increase the directional trend accuracy of the RPA 90
6.2 Re-testing the relationship between number of days historical data and directional trend accuracy 91
6.2.1 Results: testing a varying number of historical days data 91
6.3 Establishing whether tomorrow (t+1) is the optimum time period in predicting the S&P 500 93
6.3.1 Results: testing the responsiveness of the selected FMD data to the movement of the S&P 500 94
6.4 Implications of test results in optimisation of RPA 95
6.5 Future work 96
6.5.1 Linking the RPA with a 'bio-inspired' system 96
6.5.2 FMD Inputs 98
6.5.3 GPU 99
6.5.4 Figure of Merit (FoM) 99
6.5.5 Time frame of prediction result 100
6.5.6 Historical days data 100
6.5.7 Establishing a reliable data source 101
6.5.8 Predict other indices 101

6.5.9 Predicting currency pairs 104

Chapter 7 105

References 110

Appendices 119

References 125

List of Figures and Tables

Figure 2.1 S&P 500 index price 1970 - 2013 9

Figure 2.2 An artificial neuron 11

Figure 2.3 Classification - linear separable model 13

Figure 2.4 Overall framework of GAFD 19

Figure 2.5 Procedures of the SVM prediction protocol 28

Figure 3.1 FMD inputs vs. possible combinations 44

Figure 3.2 Layer of FMD input indexes 46

Figure 3.3 Error term of the sets of 6 FMD inputs to the price of the S&P 500 48

Figure 3.4 Flow diagram of model implemented on MatLab 53

Figure 3.5 The current state of shift registers (top); the regression errors of the current state (2nd); the historical record of error (3rd); the record of the optimal solution shown by the smallest error (lowest) 54

Figure 3.6 Final calculation of minimum error using the RPA 55

Figure 3.7 January 2000 – Individual FMD inputs selected by RPA 58

Figure 3.8 October 2005 – Individual FMD inputs selected by RPA 60

Figure 3.9 September 2008 – Individual FMD inputs selected by RPA 62

Figure 4.1 CPU versus GPU architecture comparison 68

Figure 4.2 Directional results, accuracy and speed of RPA when processed using MatLab, GPU card and Dedicated GPU card 77

Figure 4.3 Processing time of the RPA when implemented in three separate hardwares 78

Figure 5.1 Optimum number of FMD inputs vs. number of possible combinations 83

Figure 5.2 Pie chart displaying figure of merit for one cycle using 60 days historical data ... 85

Figure 5.3 NASDAQ 100 (1 day) price chart 20.05.2014 87

Figure 5.4 S&P 500 (1 day) price chart 20.05.2014 87

Figure 5.5 VIX (1 day) price chart 20.05.2014 87

Figure 6.1 Historical days data vs. RPA directional trend accuracy 93

Figure 6.2 Time scale vs. RPA directional trend accuracy 95

Figure 6.3 Block diagram of the LNP model.. 97

Figure 6.4 CAC40 component companies 102

Figure 6.5 Dax30 component companies 103

Figure 6.6 NASDAQ and NYSE dollar value in billions (USD) 104

Table 2.1 Top 100 S&P 500 component companies based on each company's individual percentage weight of whole index **5**

Table 2.2 Average predictive performance - hit ratio: % **20**

Table 2.3 A comparison of RMSE between different ensemble methods **24**

Table 2.4 Summary Table – advantages and disadvantages of SVM and ANN **30**

Table 3.1 Number of combinations against number of FMD inputs **42**

Table 3.2 Directional results, accuracy and speed of algorithm on a PC- platform in October 2005 .... **56**

Table 3.3 January 2000 .... **57**

Table 3.4 October 2005 .... **59**

Table 3.5 September 2008 .... **61**

Table 4.1 Key differences between a CPU and GPU **69**

Table 4.2 Specifications of hardware used to implement the RPA **71**

Table 5.1 Establishing optimum number of days historical data to test FMD inputs; 4 and 6 FMD inputs **82**

Table 5.2 Optimum number of FMD inputs **84**

Table 5.3 Figure of Merit for the 60 days ending 24/11/2008 using 6 FMD inputs **85**

Table 6.1 Testing for optimum number of historical days data **92**

Table 6.2 Testing responsiveness of the RPA **94**

Symbols, Acronyms and Abbreviations

$	**United States Dollar**
£	**British Pound**
AGP	**Accelerated Graphics Port**
AMP	**Accelerated Massive Parallelism**
ANN	**Artificial Neural Network**
BPLT	**Back-Propagation Linear Transformation**
CAC40	**Cotation Assistée en Continu 40**
CHF	**Swiss Franc**
CPU	**Central Processing Unit**
CUDA	**Compute Unified Device Architecture**
DAX	**Deutscher Aktien Index**
EMH	**Efficient Market Hypothesis**
EUR	**Euro**
Fig	**Figure**
FMD	**Financial Market Data**
FoM	**Figure of Merit**
Forex	**Foreign Exchange**
FPGA	**Field-Programmable Gate Array**
GAFD	**Genetic Algorithms for Feature Discretisation**
GA	**Genetic Algorithm**
GALT	**Genetic Algorithm Linear Transformation**
GB	**Gigabyte**
GBP	**British Pound**
GHz	**Gigahertz**
GPU	**Graphics Processing Unit**
Hr(s)	**Hour(s)**
IC	**Integrated Circuit**
JPY	**Japanese Yen**

LIBOR	**London Interbank Offered Rate**
LNP	**Linear-Nonlinear-Poisson**
M	**Month**
MAD	**Mean Absolute Deviation**
MatLab	**Matrix Laboratory**
MB	**Megabyte**
Min(s)	**Minute(s)**
Ms	**Millisecond**
MSE	**Mean Square Error**
NASDAQ	**National Association of Securities Dealers Automated Quotations**
NYMEX	**New York Mercantile Exchange**
PCI	**Peripheral Component Interconnect**
PC	**Personal Computer**
RAM	**Random Access Memory**
RBF	**Radial Basis Function**
RFV	**Receptive Field Vectors**
RMSE	**Root Mean Square Error**
RPA	**Regan Predictive Algorithm**
RW	**Rescorla–Wagner**
S&P 500	**Standard and Poor's 500 index**
Sec(s)	**Second(s)**
SVM	**Support Vector Machine**
SVMR	**Support Vector Machine Regression**
t	**Time**
U.S.	**United States**
USD	**United States Dollar**
VIX	**Chicago Board Options Exchange Market Volatility Index**
WTI	**Western Texas Intermediate**
Yr(s)	**Year(s)**

Definitions

Central Processing Unit (CPU)

A central processing unit (CPU) (formerly also referred to as a central processor unit) is the hardware within a computer that carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.

Exchange Rate

An exchange between two currencies is the rate at which one currency will be exchanged for another. It is also regarded as the value of one country's currency in terms of another currency.

Figure of Merit (FoM)

A figure of merit is a quantity used to characterise the performance of a device, system or method, relative to its alternatives.

Genetic Algorithm (GA)

Genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimisation and search problems.

Graphics Processing Unit (GPU)

Graphics processing units (GPUs) are a specialised electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel.

In-sample Forecasting

In-sample forecasts are forecasts *within* the range of the actual data used to build the regression model.

Interest Rate Futures

An interest rate future is a financial derivative (a futures contract) with an interest-bearing instrument as the underlying asset. It is a particular type of interest rate derivative.

Linear Regression

In statistics, linear regression is an approach for modelling the relationship between a scalar dependent variable and one or more explanatory variables denoted $\boldsymbol{X}$.

London Interbank Offered Rate (LIBOR)

The London Interbank Offered Rate is the average interest rate estimated by leading banks in London that they would be charged if borrowing from other banks.

Long Position

A long position in a security, such as a stock or a bond, or equivalently to be long in a security, means the holder of the position owns the security and will profit if the price of the security goes up.

Mean Squared Error (MSE)

The mean squared error (MSE) of an estimator measures the average of the squares of the "errors", that is, the difference between the estimator and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss.

Market Capitalisation

Market capitalisation (or market cap) is the total value of the issued shares of a publicly traded company; it is equal to the share price times the number of shares outstanding. As outstanding stock is bought and sold in public markets, capitalisation could be used as a proxy for the public opinion of a company's net worth and is a determining factor in some forms of stock valuation.

MatLab

MatLab (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MatLab allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Ordinary Least Squares (OLS)

Ordinary Least Squares is a method for estimating the unknown parameters in a linear regression model. This method minimises the sum of squared vertical distances between the observed responses in the dataset and the responses predicted by the linear approximation.

Out-of-sample Forecasting

Out-of-sample forecasts are used to predict future values of dependent variables.

Short Position

Short selling is the practice of selling securities or other financial instruments that are not currently owned, and subsequently repurchasing them.

S&P 500

The S&P 500, or the Standard & Poor's 500, is a stock market index based on the market capitalisations of 500 large companies having common stock listed on the NYSE or NASDAQ. The S&P 500 index components and their weightings are determined by S&P Dow Jones Indices.

Swap Rate

Swap rate is the fixed rate that records demands in exchange for the uncertainty of having to pay the short-term LIBOR (floating) rate over time. At any given time, the market's forecast of what LIBOR will be in the future is reflected in the forward LIBOR curve.

Chapter 1

Introduction

The Chinese poet Lao Tzu (6th Century BC) stated that, “Those who have knowledge don’t predict. Those who predict don’t have knowledge”, from which we can conclude that as we do not have knowledge of the future, prediction is our only option and in all walks of life a natural human trait. In trying to devise a prediction system it is clear that certain knowledge of the future is not achievable. Therefore a perfect prediction system cannot be devised and we will only ever be able to devise a partially successful system.

In a business context the successful financial investor seeks an accurate prediction of future financial performance. Although the holy grail of a perfect prediction system is impossible there are many who strive to forecast the future. Some success can be had by considering all factors that are likely to influence, or be influenced by, future changes in any financial commodity and thus it is possible to devise a prediction system that is significantly more successful than an uninformed estimate.

The role of the financial investor is to consider all market information and use this, traditionally together with personal market experience, to make a decision on whether to place a trade and, if so, at what level. As in most walks of life, the availability of ever more powerful computer systems is welcomed by investors and most finance houses now use algorithmic prediction systems on a daily basis to support their trading decisions.

The principal goal of the research described in this thesis is the design and development of a financial algorithm capable of predicting (within 24 hours) today, the value of tomorrow’s Standard and Poor’s 500 (S&P 500) financial index. The S&P 500 is an aggregate index of 500 companies listed on the New York Stock

Exchange (NYSE) and National Association of Securities Dealers Automated Quotations (NASDAQ) that 'best' represents the U.S. economy as a whole, whilst satisfying certain market capitalisation requirements (Standard & Poor's, 2014). They are chosen as they represent this performance in a single figure and hence the S&P 500 is considered a useful 'financial barometer' of the U.S. economy. On the assumption that a successful prediction system can be developed, it is my ambition to set up an investment fund with investment decisions informed using the new system.

Assessing a large amount of data has clear parallels with the way the human eye is able to assess a large amount of visual data and rapidly identify a particular area of interest. This ability is referred to as foveation. The algorithm I have developed has been inspired by the biology of the human retina.

The thesis comprises 7 chapters and 3 appendices. Chapter 2 covers previously published work on algorithmic financial forecasting. The majority of the research papers fall into two main categories (i) Artificial Neural Networks (ANNs) and (ii) Support Vector Machines (SVMs). Although both have some merit, I found that neither was able to scale sufficiently to handle the extensive data requirement to predict the S&P 500 within the necessary target time window of less than 24 hours. Also evident from the review was that curve-fitting to historical data is not a sound basis for future prediction; the future, though influenced by the past, is not solely predictable from past performance.

My conclusion from the review work of Chapter 2 was that a radically different approach was needed and so, inspired my the biology of the human eye, I devised a new algorithm, presented in Chapter 3. The algorithm requires extensive historical data, not only of S&P 500 closing prices but also historical data of a carefully selected set of 51 financial metrics covering the same period. Implementation of the algorithm to prove the concept was undertaken using MatLab running on a 6-core conventional central processing unit (CPU) computer. The net result was positive in that the accuracy of the predicted value of tomorrow's S&P 500 algorithm was better than

50%. However, the computational time far exceeded 24 hours and it became clear that a much faster computer system was needed.

The focus of Chapter 4 is on the hardware requirements to reduce the computational time to below 24 hours. It became evident that a computer system capable of extensive parallel processing would give the time reduction needed. This entailed the development of a bespoke computer system using 3 multicore state-of-the-art Graphics Processing Units (GPUs) running in parallel together with a controlling CPU. The development demanded significant software optimisation to obtain the required performance and also dynamic modification of the algorithm (all of which is described in Chapter 4).

A key element in the algorithm is the selected 51 financial metrics. If all 51 are used, the resultant computational time is too long. The solution I found is to modify the algorithm by reducing the set of 51 down dynamically to a smaller sub-set for the particular day in question and look for an optimal number that will give the 'best' accuracy within the 24 hours time constraint. This aspect of the optimisation process is the topic of Chapter 5.

Optimisation of the directional trend accuracy is explored further in Chapter 6, this time in relation to the number of historical days of data that are used in the algorithm. In the last part of the chapter additional optimisation ideas are presented and discussed. These include refining the algorithm to improve the directional trend accuracy and adapting the algorithm for different financial index prediction tasks other than the S&P 500.

The final chapter of the thesis is Chapter 7, 'Conclusions', in which I summarise the key end points of each chapter.

Chapter 2

Financial Forecasting

This thesis focuses on the development of a new approach to forecasting financial markets. The Standard and Poor's 500, commonly called the S&P 500, will be used as the test financial index to evaluate the prediction algorithms within this thesis. The S&P 500 was chosen because it is the strongest litmus test of the state of the U.S. economy (Karolyi & Stulz, 1996) and the larger industrial companies within it.

2.1 Introduction and Explanation of S&P 500

Origins of the S&P 500

- The origin of the S&P 500 harks back to 1923, when Standard & Poor's presented a series of indices that included 233 companies and covered 26 industries. The S&P 500, as it is now known, was introduced in 1957.

- The S&P 500 is often observed as a proxy for the U.S. equity market and it is the only stock market benchmark in The Conference Board's Index of Leading Economic Indicators. It has stood for U.S. stock market performance in that context since 1968 (S&P Dow Jones Indices, 2014).

How the S&P 500 is composed

- The Standard and Poor's 500 index (S&P 500) is a stock market index based on the market capitalisation of 500 (and only ever 500) large companies having common stock listed on the NYSE and NASDAQ. The S&P 500 index components and their weightings are determined by S&P Dow Jones Indices billion (S&P Indices, 2012).

- The S&P 500 represents over 80% of the total domestic U.S. equity float-adjusted market capitalisation (S&P Dow Jones Indices, 2014).

- In order to be eligible for inclusion in the S&P 500 index as one of the 500, a company must satisfy these liquidity-based size requirements:
  - Market capitalisation is greater than, or equal to, US$ 4.0 billion (Standard and Poor's, 2012)
  - Annual dollar value of shares traded must be more than the market capitalisation of the company billion (Standard and Poor's, 2012) and
  - Minimum quantity of shares traded must be more than 250,000 in each of the six months leading up to the evaluation date billion (S&P Indices, 2012).

- According to the Global Industry Classification Standard (GICS) the S&P 500 comprises ten sectors: Energy, Materials, Industrials, Consumer Discretionary, Consumer Staples, Health Care, Financials, Information Technology, Telecommunications Services, and Utilities (S&P Dow Jones Indices, 2014).

- Table 2.1 displays a list of the 100 largest weighted component companies of the S&P 500 index (based on the market capitalisation index weighting), June 2014.

Table 2.1: Top 100 S&P 500 component companies based on each company's individual percentage weight of whole index

Symbol	Company	% Weight
XOM	Exxon Mobil Corp	3.1172
AAPL	Apple Inc.	2.4098
MSFT	Microsoft Corp	2.0327
PG	Procter & Gamble	1.8204
GE	General Electric Co	1.7028
IBM	Intl Business Machines Corp	1.6814
JNJ	Johnson & Johnson	1.6238
JPM	JP Morgan Chase & Co	1.6101
T	AT&T Inc	1.5419
CVX	Chevron Corp	1.5091
WFC	Wells Fargo & Co	1.4578
BAC	Bank of America Corp	1.4035
CSCO	Cisco Systems Inc	1.3615
KO	Coca-Cola Co	1.2812
GOOG	Google Inc	1.2213
PFE	Pfizer Inc	1.2013

INTC	Intel Corp	1.1918
MRK	Merck & Co Inc	1.1113
HPQ	Hewlett-Packard Co	1.0546
WMT	Wal-Mart Stores	1.0542
PEP	PepsiCo Inc	1.0466
ORCL	Oracle Corp	0.9687
PM	Philip Morris International	0.9562
C	Citigroup Inc	0.9243
COP	ConocoPhillips	0.8175
VZ	Verizon Communications Inc	0.8096
ABT	Abbott Laboratories	0.7697
MCD	McDonald's Corp	0.768
GS	Goldman Sachs Group Inc	0.7271
SLB	Schlumberger Ltd	0.7029
UTX	United Technologies Corp	0.6742
OXY	Occidental Petroleum	0.6719
BRK/B	Berkshire Hathaway B	0.6603
DIS	Walt Disney Co	0.6347
QCOM	QUALCOMM Inc	0.6308
MMM	3M Co	0.6275
AXP	American Express Co	0.5484
KFT	Kraft Foods Inc A	0.5269
BA	Boeing Co	0.525
AMGN	Amgen Inc	0.516
HD	Home Depot Inc	0.4796
MO	Altria Group Inc	0.4695
UPS	United Parcel Service Inc B	0.4686
USB	US Bancorp	0.4639
CAT	Caterpillar Inc	0.4435
F	Ford Motor Co	0.4324
BMY	Bristol-Myers Squibb	0.4317
CVS	CVS Caremark Corp.	0.4282
AMZN	Amazon.com Inc	0.4278
EMC	EMC Corp	0.4225
LLY	Lilly, Eli & Co	0.4139
CL	Colgate-Palmolive Co	0.4103
CMCSA	Comcast Corp A	0.4077
MDT	Medtronic Inc	0.4036
TGT	Target Corp	0.3895
EMR	Emerson Electric Co	0.3877
V	Visa Inc	0.382
MS	Morgan Stanley	0.3817
UNP	Union Pacific Corp	0.3805
TWX	Time Warner Inc	0.365
UNH	Unitedhealth Group Inc	0.3569

DD	DuPont, E.I. de Nemours	0.3544
HON	Honeywell Intl Inc	0.3431
PNC	PNC Finl Services Group	0.327
DTV	DIRECTV Class A	0.3258
BK	The Bank of New York Mellon Corp	0.325
MON	Monsanto Co.	0.3211
APA	Apache Corp	0.3183
DOW	Dow Chemical	0.3169
FCX	Freeport McMoRan Copper & Gold	0.3113
TXN	Texas Instruments Inc	0.3089
LOW	Lowe's Cos Inc	0.3068
SO	Southern Co	0.3025
GILD	Gilead Sciences Inc	0.3021
WAG	Walgreen Co	0.2984
NKE	NIKE Inc B	0.2913
NEM	Newmont Mining Corp	0.2913
GLW	Corning Inc	0.2864
HAL	Halliburton Co	0.2828
LMT	Lockheed Martin	0.2802
DE	Deere & Co	0.2797
EXC	Exelon Corp	0.2769
DVN	Devon Energy Corp	0.2748
KMB	Kimberly-Clark	0.2688
BAX	Baxter Intl Inc	0.2675
PX	Praxair Inc	0.2668
EOG	EOG Resources	0.2635
PRU	Prudential Financial Inc	0.2623
SPG	Simon Property Group	0.258
D	Dominion Resources Inc	0.2571
COST	Costco Wholesale Corp	0.2486
CELG	Celgene Corp	0.2479
NWSA	News Corporation A	0.2464
APC	Anadarko Petroleum Corp	0.2458
EBAY	eBay Inc.	0.2427
ESRX	Express Scripts Inc	0.2403
TRV	Travelers Cos Inc	0.2399
MET	Metlife Inc	0.2386
AFL	AFLAC Inc	0.2382
MRO	Marathon Oil Corp	0.2378

At what rate does the S&P 500 change?

- The index value is updated every 15 seconds during trading sessions and the market is informed by Reuters America, Inc. (S&P Dow Jones Indices, 2014).

Key factors I believe influence the movement of the S&P 500

- **Individual Company Performance**
    - The individual S&P 500 component companies have a direct effect on the price and therefore performance of the index.
- **Macro/US Economics**
    - Sentiment with regard to U.S. and global economic confidence driving investor willingness to have either more or less exposure to the S&P 500.
- **Weight of Investment**
    - This is the amount of daily inflow/outflow of investor capital entering/leaving the S&P 500 index.
- **Divisor of S&P 500 index**
    - The S&P 500 uses a divisor number, which has little mathematical rationale behind it, but remains consistent and therefore enables comparability within the S&P 500 index over time. This number can alter each day upon close of trading, if the number of shares in issue in any of the companies included in the S&P 500 index changes (S&P Dow Jones Indices, 2014).

- **Performance of the S&P 500**

  - As Fig 2.1 depicts, since 1970 the S&P 500 index has recorded a generally positive trend, which has largely tracked inflation, but has also reacted to times of financial instability, namely the 2007 economic crisis.

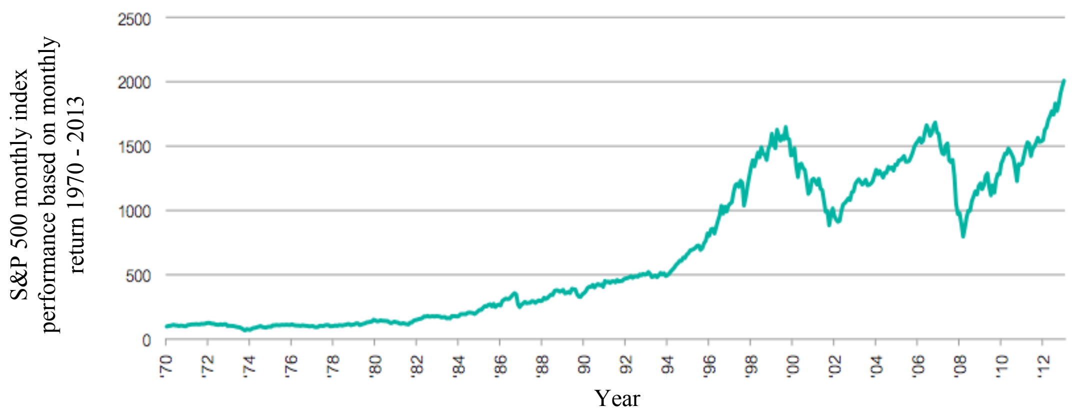


Figure 2.1: S&P 500 index price 1970 - 2013 (S&P Dow Jones Indices, 2014)

2.2 In-sample and out-of-sample forecasting

When forming statistical forecasting models, there are two types of forecast: in-sample and out-of-sample.

In-sample forecasting does not attempt to forecast the future path of one or several economic variables. Instead, in-sample forecasting uses today's information to forecast what today's outcome should be (depending on what inputs are included and the desired outcome of the model), therefore meaning the fitted values estimated in a regression are in-sample forecasts.

Out-of-sample forecasting attempts to use today's information to forecast the future behaviour of a particular entity (in the case of this study, the S&P 500 index) at tomorrow (t+1). When developing different statistical models, the term 'out-of-sample performance' refers to the model's performance (i.e. accuracy, predictive power) on data that were not included in the sample used to calibrate the models parameters.

Whilst in-sample data are important when progressing through the development stages of a particular statistical forecasting model, out-of-sample testing is a way to guard against curve-fitting, and hence imperative in the successful application of a particular forecasting model at t+1.

As mentioned in Yuan (2011), it is crucial that any methodology used for financial prediction must be greater than 50% accurate out-of-sample on any chosen time scale to be successful.

2.3 Current forecasting methods

Current methods of forecasting in financial markets require very intensive algorithms as the parameter inputs and computational complexity needed to create a meaningful prediction is large. Two of the most popular methods used are Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) (Huang et al., 2004), both of which are able to predict with some degree of success. Different approaches have inherent advantages and disadvantages and will be reviewed in the next section to provide the context for the new predictive algorithm which is presented in Chapter 3.

2.3.1 ANN Review

Since the first neural model by McCulloch and Pitts (1943), many different models referred to as ANNs have been developed to derive meaning from complicated data. A neural network is a computational technique that is designed to imitate the capability of the human brain in order to process large amounts of information and formulate patterns from often highly complex data sets. More specifically, the term 'neural' is derived from the 'three-dimensional lattice of network' among brain cells

(Hamid, 2004) and, like the human brain, uses a high number of parallel processing elements in order to acquire high computation rates.

Akin to the human brain, generating optimum performance using ANNs requires a significant and often lengthy training period. Hamid (2004) likens ANNs 'learning period' to the way a young child learns to distinguish and process different patterns, shapes and sounds within its brain when growing up. Drawing on the same logic, the training period for ANNs is correlated by the different weights input into the activation function of the network. Each neuron consists of inputs (like synapses) that are then multiplied by certain weights. Through changing the weights of an artificial neuron, the desired output needed for each specific input can be obtained. The extent of the degree of these receptive signals is then computed by a given mathematical function that determines the activation of the neuron. This process is represented in Fig 2.2.

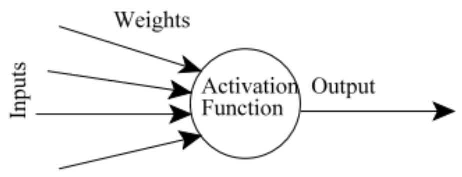


Figure 2.2: An artificial neuron (Gershenson, 2003)

The training process can be viewed as an optimisation problem, where the core aim is to minimise the mean square error of the entire set of training data. However, even when a data set has irrelevant inputs, the networks can learn key features of the data.

This problem can be solved in a range of ways with recent literature exploring a variety of methods including standard optimisation heuristics like simulated annealing, but also more unique optimisation techniques such as genetic algorithms and methods utilising back-propagation. In recent literature, the back-propagation algorithm has been most commonly used in the ANN training process, however this method has been met with a number of limitations, primarily concerning the extent of adjustment to the weights in each iteration.

ANNs present particular promise in generating patterns from economic data due mainly to their effectiveness in unstructured decision-making tasks, which enables them to be able to deal with, and make sense of 'fuzzy' non-linear data patterns. The use of ANNs in financial prediction also allows deeper analysis of large sets of economic data, especially data sets that have the tendency to fluctuate within a short period of time, as most economic data does. Researchers believe ANNs offer an alternative response to successful stock market prediction, something that has been disregarded by the rise in exposure of the efficient market hypothesis in recent discourse.

2.3.2 SVMs: description and review

The second most common computational technique used for algorithmic prediction of financial markets is known as the SVM algorithm, which was initially proposed by Cortes and Vapnik (1995, p.273) in their seminal research paper, "Support Vector Networks" as "a new learning machine for two-group classification problems". Throughout the last two decades, SVMs have been developed and utilised extensively in a variety of fields as a technique for data classification, regression and prediction.

The use of SVMs focuses on class separation and seeks to find the optimal separating hyper plane between the two classes by maximising the margin between the closest points of the classes. Fig 2.3 (Meyer, 2012) depicts the theoretical notion of SVMs using a classification model; the points lying on the boundaries are referred to as support vectors and the centre of the margin is the optimal separating hyperplane.

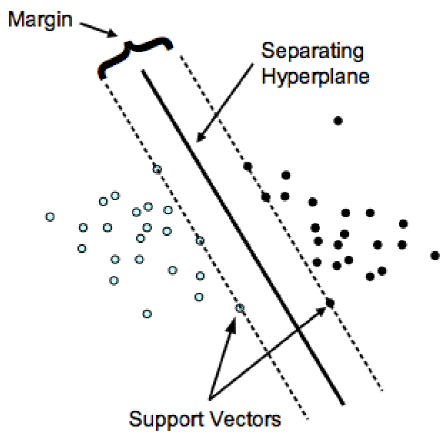


Figure 2.3: Classification - linear separable model (Meyer, 2012)

Hearst et al. (1998) cite the fundamental advantage of the support vector algorithm as 'lying at the intersection of learning theory and practice'. Real world applications often mandate the use of more complex algorithms that are more difficult to analyse theoretically. However, SVMs are analysed with relative ease mathematically due to the algorithms correspondence to a linear method in a high dimensional feature space, nonlinearly related to input space.

It is also imperative to understand that SVMs belong to the general theory of kernel methods (Scholkopf and Smola, 2002) and rely on the data only through scalar products (an algebraic operation that takes two equal-length sequences of numbers and returns a single number). Therefore, when dealing with complicated algorithms for non-linear pattern recognition or extraction, for the sake of analysis, the technique can continue to be based on a simpler linear algorithm; through the use of kernels, all necessary computations are performed directly in input space.

Tay and Cao (2001) expressed that SVMs had been utilised as a tool to aid problems associated with financial forecasting, however to a limited extent. Due to the degree of accuracy of certain financial forecasts being calculated via the estimates' divergence from the observed values, these methods of forecasting based on reducing forecast error are not entirely sufficient to fulfil its objectives; financial trading decisions propelled by forecasts containing any form of forecasting error are unlikely to be as profitable as trading guided by a more accurate and unconditional prediction in stock market movement (Yuan, 2011).

2.4 Review of ANNs and SVMs

In this section I will be reviewing significant papers written on ANNs and SVMs. My intention is to consider the advantages and disadvantages of both approaches. The findings of this literature review were used to inform the basis in the development of a novel approach, presented in Chapter 3.

As shown later in Kim & Han (2000) it is crucial that any methodology used for financial prediction must be more than 50.1% accurate out-of-sample on any chosen time-scale.

The core aim of White's (1998) paper titled 'Economic prediction using neural networks: the case of IBM stock returns' is to demonstrate how the search for previously undetected regularities using neural network methods might proceed, using the case of IBM daily common stock returns as an example.

White (1998) opens by drawing on literature by Lapedes and Farber (1987) that highlight the value of neural networks in performing complicated pattern recognition and non-linear forecasting tasks. Their work applied neural networks to decoding protein sequences as well as demonstrating how neural networks are capable of decoding deterministic chaos. However, the question still remains as to whether this theory can be used in extracting non-linear regularities from economic time series. The paper then dampens the optimism previously offered by neural networks in predicting the stock market, by introducing the concept of the "efficient market hypothesis" (EMH). This concept states that the movement of an assets price based on public information is random and hence completely unpredictable; the author attributes this absence of predictability to the reason that there is a lack of $100 notes lying on public pavements. Alternatively the basis for EMH validation relies on the fact that prediction of a truly random market is not possible.

However, the paper counters the claims put forward in the EMH by utilising the work of Simon (1982) and in particular, his bounded rationality arguments which underline how humans are inherently limited in their ability to process information. White (1988) then switches back to the functions of neural networks to act as the processor of available information, citing how an advance in technology would effectively allow the creation of a form of inside information and thus create a margin for the successful prediction of economic time series.

The study presented a target variable of interest, r_t , which is the single day rate of return on IBM common stock on day t. The definition of r_t used was

(2.1) $$r_t = (p_t - p_{(t-1)} + d_t)/p_{(t-1)}$$

where p_t is the closing price of day t and d_t is the dividend paid on day t. White used 1,000 days of data, from 5000 days of data that was available, to train the neural network. Evaluation of the knowledge that the neural network had acquired was undertaken using 500 days of data either side of the training period. White's detailed analysis, based on a relatively simple neural network, did not indicate conclusively that the EMH was incorrect, as he states in the paper, "the present neural network is not a money machine." Despite the failure to find clear evidence against the EMH, the results did contribute insightful information to this debate. Firstly, this paper established that even less complicated networks are capable of misleadingly over-fitting an asset price series with as many as 1,000 observations. Secondly, the author stresses the significance of the optimisation methods being essentially local in this research study; there remains no guarantee that a global maximum is found, the author suggests global optimisation methods such as simulated annealing or the genetic algorithm. However, more positively, the findings established how such simple networks are capable of "extremely rich dynamic behaviour" (White, 1988), as evidenced by time-series plots of $\mathbf{r_t}$. Through its progression of computationally efficient methods for obtaining mature networks, this exercise also sets to highlight the significance of statistical inference in establishing the performance of neural network models.

Moving beyond the arguably limited scope of this study, the author stresses the need to expand the search range to gain evidence against the EMH. Elaborating the network and allowing additional inputs such as other stock prices and volume, leading indicators and macroeconomic data will increase the scope of the findings; however the author stresses how any elaborations *must* be supported with vast infusions of data for the training period because the more connections, the greater the danger of over-fitting.

Bergerson and Wunsch (1991) open by explaining the application of neural networks in solving difficult problems that algorithmic computer programming has failed to solve, including predicting the stock market. The analogy of an insect is used to display the different pattern processing, learning and other capabilities that evade even the most powerful computers.

Similar to the study undertaken by Rumelhart, et al. (1986), this research utilises the back-propagation network, a method of training an ANN; it requires a dataset of the desired output for many inputs, which makes up the training set. The authors believe their study is set apart from other research predicting stock market returns using neural networks due to the provision of a good set of training data in order to control precisely what the network learned about market prediction. The paper states how it is imperative not only to draw on historical data but also on human expert input to implicitly define patterns, using hindsight, that an intelligent system might have been able to use for an accurate prediction. Therefore, the framework of the network learned to produce signals is based on data that looked favourable to human experts, whilst upholding the requirement that anything considered to be a ‘good example’ must also be accompanied by profitability.

The authors stressed the labour intensive nature of their research model in producing a substantial set of training examples; four technical indicators were plotted against the daily S&P 500 data (between September 1980 and January 1981) and displayed clear 'buy' and 'sell' decisions based on the human expert. In attaining the framework to establish these plotted 'buy' and 'sell' positions, the authors added to the selection of parameters for the neutral architecture. The number of hidden units (determined by pruning) and input units remained the same at 54, however after training had stabilised, the units with the smallest weights, therefore least significance, were removed. The amount of training was then monitored; when the error on the test data and the error on the training data were the same, no further training was carried out.

The results of this study displayed a theoretical 660% growth over the 25 months commencing January 1989 with the maximum drawdown for the month of September 1989 which saw a 2.3% loss. However, the conservative risk management rule, which governs where stop loss points are put to control losses when an incorrect prediction is made, was applied to allow maximum advantage of profit making opportunities.

In conclusion, this paper's use of a hybrid methodology offers a clear insight into the advantages gained over either rule-based or unaided neural network approaches. The method described ensures that the rigidity of rule-based approaches is overcome whilst benefiting from pattern recognition gained through unaided neural networks. From this paper, it is clear that despite the hard work and demands on the experts' time, the synergy of the rule-based and neural system ensures the design of an attractive risk-to-reward ratio trading model that produces a positive set of results.

The next paper I reviewed, written by Malliaris (1994) cites the U.S. stock market crash in October 1987 as the reason for questioning the 'classical' approach to financial modelling, namely that financial indices, such as the S&P 500, are modelled as a purely random walk. By its very nature 'random' means that systematic forecasting of stock market prices cannot be achieved. She claimed that the crash suggested that financial markets were not simply random. The new approach proposed a non-random structure term 'chaotic dynamics'. To verify her claim the author devised an ANN which was used to predict market prices but, if the ANN produced better predictions of the S&P 500 than a Rescorla–Wagner (RW) model, then the structure of the financial market was confirmed as non-random. In addition to confirming the non-random nature of the S&P 500 the author hoped that it would (i) confirm the chaotic nature of the index and therefore indicate that management of the S&P 500 portfolio was possible, and (ii) if she could show that the ANN outperforms the random walk model then this suggests that further research should be undertaken to establish links between the unknown but deterministic patterns of the S&P 500 to explanatory variables.

The neural network was constructed and developed using the Brainmaker software, produced by California Scientific Software Inc. When assembling the neural network a decision has to be made regarding number of nodes to have in the input and hidden layers. An optimal number of nodes were determined using Brainmaker's genetic training option. Having decided the number of layers and nodes per layer the weights assigned to each node were altered until the 'best' performance was achieved.

The author concluded that her ANN did outperform the random walk model. The average Mean Absolute Deviation (MAD) for the ANN was 3.167 compared with 6.188 for the random walk model and the average Mean Square Error (MSE) for the network was 15.609 compared with 41.32 for the random walk model. Having demonstrated that her ANN was a more successful predictor than the RW model, her conjecture that the S&P 500 is indeed non-random opened the door for further research into the nature of the deterministic model that describes the time variation of the S&P 500.

The following paper, written by Kim and Han (2000), proposes the use of an ANN as a data mining technique. However, as in the majority of studies, the authors draw on the limitations of ANNs in their inconsistent and unpredictable performance when faced with often complex and noisy stock market data. A novel hybrid approach is developed that combines both ANNs and genetic algorithms for feature discretisation (GAFD) to mitigate the limitations of a sole ANN approach. Instead of applying the commonly used and, arguably, unsuccessful technique of the gradient descent algorithm to train the network, this paper adopts Sexton et al. (1998) theory to apply the use of global search algorithms to obtain a weighting of the network. Their study employed a GA to find the weighting of the ANN, with the subsequent results displaying the superiority of a GA-derived solution to the corresponding back-propagation solution. The authors then draw on the work of Liu and Motoda (1998) to explain feature discretisation, highlighting its importance in improving the generalisability of the learned results by heavily simplifying the process of learning.

Therefore, drawing on these techniques, this study seeks to utilise GA in order to find the near-optimal thresholds for feature discretisation whilst, at the same time, searching the connection weights between layers in the ANN. The framework of this paper's GAFD research model is represented in Fig 2.4 and in the context of this study is applied to predict the direction of change in the daily Korea stock exchange index (KOSPI).

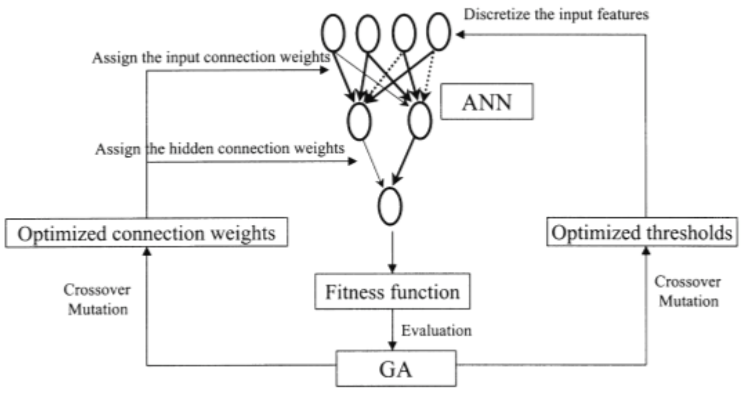

Discretize the input features
Assign the input connection weights
ANN
Assign the hidden connection weights
Optimized connection weights
Optimized thresholds
Crossover
Mutation
Fitness function
Crossover
Mutation
Evaluation
GA

Figure 2.4: Overall framework of GAFD (Kim & Han, 2000)

Of the 3 models compared, Table 2.2 confirms that the GAFD has higher prediction accuracy than back-propagation linear transformation neural network (BPLT) and GALT (linear transformation with ANN trained by GA) by 10% to 11% for the holdout data. The authors stress the importance to note how the difference in prediction accuracy between the training data and holdout data is likely to be attributed to the fact that the globally searched discretisation profoundly simplifies the learning process and eradicates all non-related patterns.

Table 2.2: Average predictive performance - hit ratio: % (Kim & Han, 2000)

Year	**BPLT**		**GALT**		**GAFD**	
	Training (%)	Holdout (%)	Training (%)	Holdout (%)	Training (%)	Holdout (%)
1989	59.05	48.28	57.33	49.12	68.10	59.65
1990	62.23	49.15	59.23	56.90	66.95	60.34
1991	58.97	53.45	53.42	50.00	63.25	56.90
1992	61.02	51.72	60.17	44.83	66.95	58.62
1993	54.01	44.07	54.43	44.07	67.09	61.02
1994	62.45	64.41	61.18	59.32	63.29	62.71
1995	63.83	44.83	63.83	53.45	69.36	65.52
1996	61.28	60.35	61.70	50.00	64.26	67.24
1997	46.15	50.00	50.43	50.00	64.10	62.07
1998	55.98	51.72	56.84	48.28	64.53	62.07
Total	**58.50**	**51.81**	**57.86**	**50.60**	**65.79**	**61.70**

In their closing remarks, the authors state how their hybrid model goes beyond other studies by seeking the optimal thresholds of feature discretisation for the dimensionality reduction, hence ensuring their GAFD model can discretise the original continuous data whilst simultaneously assigning the genetically evolved connection weights. The authors concluding remarks were, "GAFD reduces the dimensionality of the feature space then enhances the generalisability of the classifier from the empirical results" (Kim & Han, 2000 p.131).

The paper written by Khan et al. (2008) seeks to prove the superiority of genetic based algorithm techniques compared with the use of neural network back-propagation in stock rate prediction. The authors first outline the application of neural networks in stock market prediction, similar to the techniques in other neural network prediction studies already discussed.

The authors then apply the concept of genetic algorithms (GAs) to stock market prediction; genetic algorithms are viewed by Goldberg and Deb (1991) as being an "iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges" (*in* Khan et al., 2008 pp162-163). The authors use Darwin's principle 'survival of the fittest' to demonstrate the concept of GAs in their ability to create a set of viable solutions but also to channel these solutions in a way that identifies only the most promising of solutions.

According to Khan et al. (2008) generation of these new solutions is developed by drawing upon three genetic recombination operators. Firstly, 'biased reproduction' selects the fittest people to reproduce. Secondly, 'crossover' establishes the combination of parent chromosomes to create chromosomes of the children and passes the fittest genes to the next generation. The final recombination operator surrounds the notion of 'mutation'; this establishes the size of the total population, taking note that despite an increased population leading to an increased chance of establishing the global optimum, it would be at the expense of more central processing unit (CPU) time. GAs were then combined with the theory of back-propagation neural networks to form a hybrid approach, which the authors of this paper claim to be better than either rule based or unaided neural network approaches.

In the experimental results section of this paper, the authors stress the importance of data normalisation in training the GA based back-propagation neural network. Moreover, because the close rate and volume of stocks are the key identified quantitative factors for individual equities, both these variables were used as inputs whilst the next stock rate was utilised as the target of the study for training the networks.

This study used data from the Indian National Stock Exchange and the stock ‘Maruti’ to demonstrate the comparison of the two approaches to stock market prediction. The authors concluded that for this stock, GA based back-propagation neural network gave a more accurate prediction (98.31% success rate) compared with the back-propagation neural network (93.22% success rate). This study was conducted using in-sample data, however the only way of achieving true validation is through the use of out-of-sample data.

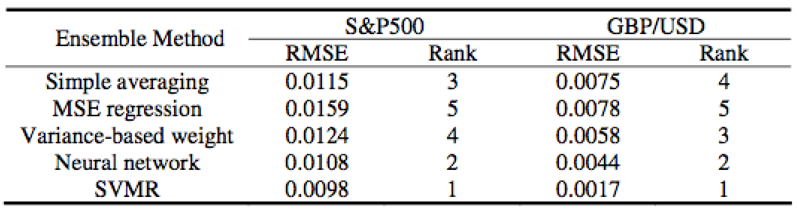
The main focus of this paper written by Gupta and Wang (2011) was to predict future stock index prices, calculate directional efficiency and rate of returns, in order to develop a trading system capable of delivering high yield over a long period of time. Gupta & Wang open by discussing the term Efficient Market Hypothesis (EMH), which asserts how it is impossible to predict the stock market in such a way as to earn greater profits from the stock market in an efficient market. However, in line with Malliaris (1994) reviewed previously, the authors are quick to rebut these claims and draw on literature by Thaler (1985) to illustrate that it is theoretically possible to successfully predict the stock market movements. The author proceed to highlight two specific categories of prediction systems that have had some success but ultimately fail due to certain flaws in the model. Firstly, Fernandez-Rodriguez et al. (2000) relatively simple ANNs to predict and trade in the Madrid Stock Market Index. This process used 9 lagged inputs to predict the prices to base different buy/sell decisions; however this model did not perform well in a bullish market. Secondly, the authors display how the work of Chang et al. (2004) who used a neural network model based on past prices to successfully obtain around 16% of returns per annum using a weekly prediction model; however this model failed for daily predictions.

Lai et al. (2006) open their paper by proposing a triple-phase novel nonlinear ensemble forecasting model based on SVM regression principles. The authors use this proposal as the basis of their study in order to overcome the shortcomings of neural network models found in previous forecasting literature.

Firstly, individual neural network predictors were generated by drawing on Breiman's (1990) literature on bias-variance trade-off. Having established four key methods in generating diverse models, the study focuses on selecting appropriate ensemble members through the principal components analysis. Finally, and dependent on the previous two stages, a collection of appropriate members can be obtained and combined into an aggregated predictor in an appropriate ensemble strategy (there are two ensemble strategies: linear ensemble and non-linear ensemble). The authors outline two main linear approaches in 'the simple averaging approach' and 'the weighted averaging approach' but focuses more on the non-linear approach principally because it has a promising approach for determining the optimal neural ensemble weight of the predictor. However, contrary to previously published work, the study proposes a new nonlinear ensemble method with a SVM regression (SVMR) principle.

Empirical analysis was used in the paper that consisted of two time series data sets: the S&P 500 index series, and the GBP/USD series, shown in Table 2.3 which was reproduced from data in the paper by Lai (2006). In-sample data is collected from January 1991 to December 2000 with out-of-sample data being taken from January 2001 to December 2002.

Table 2.3: A comparison of root mean squared error (RMSE) between different ensemble methods (Lai et al, 2006)



Ensemble Method	S&P500		GBP/USD	
	RMSE	Rank	RMSE	Rank
Simple averaging	0.0115	3	0.0075	4
MSE regression	0.0159	5	0.0078	5
Variance-based weight	0.0124	4	0.0058	3
Neural network	0.0108	2	0.0044	2
SVMR	0.0098	1	0.0017	1

The results presented by the authors demonstrated that of all the ensemble methods, the SVMR model performed the best and therefore confirmed how the non-linear ensemble methods are more suitable for financial time-series forecasting than the linear approaches, due largely to the high volatility of the financial time series.

Charles Yuan (2011) opens his research paper by stressing the importance yet complexity of financial forecasting in providing concrete data for investment decisions. Drawn from the heated historical debate surrounding the predictability of stock returns out-of-sample, the author cites the current ambiguity of stock prediction techniques as the key basis for his study.

Drawing initially on the early work by Alfred Cowles, whose seminal paper, "Can Stock Market Forecasters Forecast?" published in 1933, concluded how forecasters fail to predict the stock market. The paper reviews more contemporary literature, most notably Fama's (1991) work, that presents evidence of numerous variables able to predict, with reasonable success, the U.S. aggregate stock returns. The author categorises these variables into three separate types: 'price multiples predictors' (dividend-price ratio, earnings-price ratio and stock market volatility); 'macroeconomic predictors' (nominal interest rates, interest rate spreads and expected business conditions); and 'corporate actions predictors' (dividend payout ratio and corporate issuing activity). However, whilst accepting that the results of these diverse stock forecasting techniques may prove successful 'in-sample', the work of Goyal and Welch (2003, 2008) suggests that stock return predictability is unreliable due largely to the substantial model uncertainty and parameter instability surrounding data-generating processes for stock returns; this indicates that predictive regression models are unstable and not well suited for out-of-sample forecasting.

Through the use of SVMs combined with conventional predictive regression models that should exhibit substantially improved out-of-sample predictive power, the author aims in this study to build upon and ultimately disprove claims made by Goyal and Welch (2003, 2008) who view out-of-sample stock return predictions as unreliable. Fig 2.3 represents the procedures of the SVM prediction protocol.

The author then introduces the benefits of SVMs in relation to prior studies that have produced limited results using the ANN in stock market returns due to the complex dimensionality of data within stock markets. A two-fold reason why SVMs produce a useful technique for data classification, regression and prediction is provided by the author to present a relatively uncomplicated notion whilst remaining sure that the predictive accuracy of this approach overwhelms many other methods. The author develops his thesis further by explaining the role of linear classifiers and kernels involved in the application of SVMs in financial data prediction techniques.

The second development introduced in this paper is the predictive regression model. The author explains how conventional regression models typically examine stock return predictability through the following predictive regression model:

$$\mathbf{r}_{(t+1)} = \alpha_i + \beta_i \, \mathbf{x}_{i,t} + \varepsilon_{i,(t+1)} \qquad \textbf{(2.2)}$$

where $\mathbf{r}_{(t+1)}$ is the return on a broad marker index in excess of the risk-free interest rate (namely, the equity risk premium) from time **(t+1), x** $_{\mathbf{i,t}}$ is a variable used to predict the equity risk premium, and $\boldsymbol{\varepsilon}_{\mathbf{i,(t+1)}}$ is a zero-mean disturbance term. The author then introduces a sequence that he coined 'the sequence of expanding windows' to generate out-of-sample predictions and produces an example of this sequence in practice: "In order to predict the equity risk premium at time **(t+1)**, we take all historical information from time **1** up to time **t**. Because as **t** grows larger, the information from time **1** up to time **t** also gets larger, I call it a sequence of expanding windows" (Yuan, 2011 p.10). Despite the historical average of the equity risk premium being used as a benchmark forecast corresponding to the constant expected excess return model, this study draws on the work of Campbell and Thompson (2008) in evaluating predictive methods, who use the $\mathbf{R^2_{OS}}$ statistic to measure the proportional reduction in mean squared forecast error (MSFE) for a proposed model relative to the historical average benchmark. Therefore when $\mathbf{R^2_{OS} > 0}$, the proposed model is considered to outperform the historical average benchmark in the sense that it reduces the MSE in the out-of-sampling prediction.

The final theory drawn upon in this paper centres on Bates and Granger's (1969) notion that combining forecasts across models often produces a forecast that performs better than the best individual model. Moreover, Yuan uses Timmermann (2006) to emphasise how forecast combination can be thought of as a diversification strategy that improves forecasting performance, the same way that asset diversification improves portfolio performance. The paper affirms how a combination between SVM and predictive regression models is a move away from the traditional forecast combinations. Therefore, when a SVM predicts that the equity risk premium is negative/non-negative for the current period, this theory seeks to find all historical negative/non-negative equity risk premium data, and use predictive regression models to estimate the period of the current equity risk premium for the forecast combination, the evaluation technique remains the same in this paper. The author continues to use the statistic $\mathbf{R^2_{OS}}$ to evaluate whether or not the combined strategy outperforms the benchmark historical average equity premium.

In order to predict the equity risk premium, the author uses a set of 14 macroeconomic variables (1-14) and two technical indicators (15 and 16) selected from the literature Goyal and Welch (2008) and Neely et al. (2003):

1. *Dividend-price ratio* (log), DP: log of a twelve-month moving sum of dividends paid on the S&P 500 index minus the log of S&P 500 index prices.
2. *Dividend yield* (log), DY: log of a twelve-month moving sum of dividends minus the log of lagged S&P 500 index prices.
3. *Earnings-price ratio* (log), EP: log of a twelve-month moving sum of earnings on the S&P 500 index minus the log of index prices.
4. *Dividend-payout ratio* (log), DE: log of a twelve-month moving sum of dividends minus the log of a twelve-month moving sum of earnings.
5. *Stock variance*, SVAR: monthly sum of squared daily returns on the S&P 500 index.
6. *Book-to-market ratio*, BM: book-to-market value ratio for the Dow Jones Industrial Average index.
7. *Net equity expansion*, NTIS: ratio of a twelve-month moving sum of net equity issues by NYSE-listed firms to the total end-of-year market capitalisation of NYSE stocks.
8. *Treasury bill rate*, TBL: interest rate on a three-month Treasury bill (secondary market).
9. *Long-term yield*, LTY: long-term government bond yield.
10. *Long-term return*, LTR: return on long-term government bonds.
11. *Term spread*, TMS: long-term yield minus the Treasury bill rate.
12. *Default yield spread*, DFY: difference between BAA- and AAA-rated corporate bond yields.
13. *Default return spread*, DFR: long-term corporate bond return minus the long-term government bond return.
14. *Inflation*, INFL: calculated from the CPI (all urban consumers).
15. %MA(2,12): calculated as the percentage change from MA(12) to MA(2).
16. MOM(12): calculated as the percentage change from $\mathbf{P_{t-12}}$ **to** $\mathbf{P_t}$.

(In-sample data range from 1938:06 to 1974:12, and out-of-sample data range from 1975:01 to 2010:12).

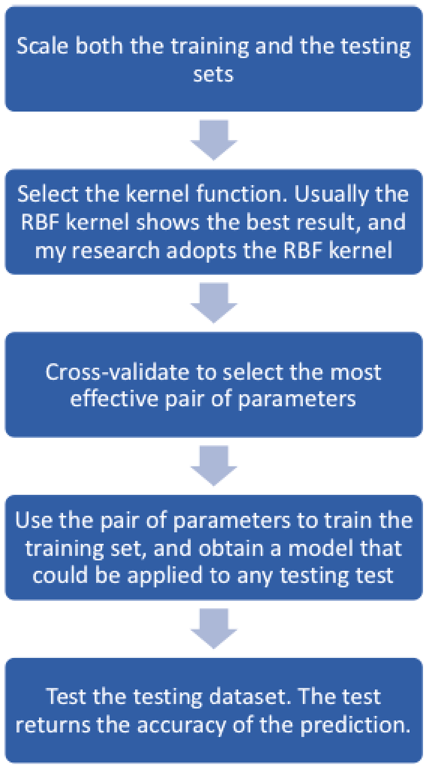

Scale both the training and the testing sets
Select the kernel function. Usually the RBF kernel shows the best result, and my research adopts the RBF kernel
Cross-validate to select the most effective pair of parameters
Use the pair of parameters to train the training set, and obtain a model that could be applied to any testing test
Test the testing dataset. The test returns the accuracy of the prediction.

Figure 2.5: Procedures of the SVM prediction protocol (Yuan, 2011)

When analysing which types of data worked well with SVMs in this study, the author stresses the danger of using a naïve application of SVMs to a large set of explanatory variables; this would return a low SVM prediction accuracy and would produce results little better than guess work (a little above 50%).

In order for SVMs to work well, this study proposed that the set of explanatory variables should have three key characteristics. Firstly, because the response variable, which is the equity risk premium, measure the change in stock market returns, correspondingly the explanatory variables need to measure changes as well. Secondly, the selected set of explanatory variables should have relatively low correlations with each other. Explanatory variables that are highly correlated do not provide much extra information that helps with prediction; indeed they introduce noise that disturbs prediction accuracy. Thirdly, the explanatory variables need to have relatively high correlations with the response variable, in this case the sign (positive or negative) of the equity risk premium.

Based on these proposed sets of explanatory variables, data results displayed out-of-sample SVM had a 86.1111% prediction accuracy. When combining SVM with predictive regression, results graphs display out-of-sampling actual equity risk premiums, as well as out-of-sample equity risk premium predictions based on the combined strategy and the benchmark historical average. It is clear from the graph that the greater predictive power of the combined strategy is observed, only with smaller magnitude compared with the actual out-of-sample realisations of the equity risk premiums.

Overall, this paper has made clear to me that despite recent studies which have developed strategies to accommodate many of the problems challenging traditional predictive regression models, out-of-sample prediction tests remain unsuccessful owing largely to the model uncertainty and parameter instability. In contrast, however, this article underlined the value of SVMs for successful stock market return predictions in attaining over 87% prediction accuracy out-of-sample. The significance of SVMs as a powerful predictive tool is further underlined when using DP (log of dividend-price ratio) as a predictive variable with the combined strategy shown to outperform the benchmark historical average by 33.44% in the MSFE sense.

2.5 Observations and conclusions based on literature review

Table 2.4 displays the key advantages and disadvantages of the two approaches based on the literary review.

Table 2.4: Summary Table – advantages and disadvantages of SVM and ANN

Approach	Advantages	Disadvantages
SVM	Can handle non-linear data Good classifications capabilities Not relying on model so requires fewer data assumptions	Processing speed highly dependent on the quantity of inputs Hard to parallel process Out-of-sample results have not been successfully demonstrated
ANN	Can handle non-linear data Good prediction capability Ability to extract patterns and detect trends that are too complex to be noticed either by humans or computer techniques. Inherently parallel nature	May have local minimum convergence problem when trained with certain algorithms. Human intervention needed to specify when a MSE becomes acceptable Out-of-sample results have not been successfully demonstrated

Both ANNs and SVMs are predicated on the fact that history informs the future, which in the case of financial forecasting is incorrect. Whilst history does not necessarily guarantee successful prediction of future market movement, it does contain relevant, though not sufficient, information for financial forecasting. Having reviewed the literature on ANNs and SVMs, these methodologies cannot be developed further to successfully predict the directional trend of the S&P 500.

Reflecting on previously published work, it became clear to me that what's required is a new approach that takes into account associated key drivers that influence the financial environment as well as the historical movements of the S&P 500. These inputs essentially capture expert knowledge and must be incorporated into the algorithm as they are significant in influencing the directional trend of the S&P 500, as well as taking into account historical data.

My novel approach is likely to require greater computational power as the pure volume of data to be processed will significantly increase above previously published algorithms, assuming that logic dictates that more inputs will provide greater accuracy. However, a key aspect will be establishing what the optimum number of inputs and optimum number of historical data points are needed to provide the most accurate financial prediction given time and computational limitations.

In recent years there has been a considerable number of revolutionary technological developments based on bio-inspiration, where biology has been studied closely and used to inspire an alternative and often more efficient technique, particularly in the area of pattern recognition. These new approaches diverge from ANN and SVM techniques, offering a powerful novel way forward to address the development of this radically different approach and will be described in Chapter 3.

Chapter 3

The Algorithm (RPA) and Proof of Concept

In this chapter a radically different algorithm for financial forecasting is presented following the findings of the literature review discussed in Chapter 2. All companies included within the S&P 500 index operate within the same financial market conditions (Gross, 2013) which is an imperative to be taken into account when forming the basis of a predicted algorithm, such as the Regan Predictive Algorithm (RPA). Therefore, an appropriate algorithm, the RPA, has been developed which includes both historical S&P 500 closing prices in tandem with a carefully crafted set of significant financial data deemed to hold the highest success in predicting the future trends and value of the S&P 500.

My approach in establishing a prediction result for the S&P 500 index is based on a bio-inspired methodology. The technique used by the RPA to attain a prediction result is similar to the function of the human eye. In assessing the human eye, the retina has clear inputs (light as a stream of images projected onto the retina) and outputs (optic nerve impulses), which hence leads to a well defined, unidirectional information flow. Similarly, with this concept in mind, the RPA inputs vast amounts of data into a regression core, which then selects and sorts the most relevant data based on its mean squared error (MSE) and forms the basis for the prediction result. The RPA will focus on 'hot spots, which are particular areas of interest that are likely to influence or are influenced by the S&P 500 in the same way a human eye will react to, and process a dominant area of activity, even if it is in the peripheral vision.

In this chapter I will introduce the concept of the RPA and outline the 51 chosen FMD inputs that effectively capture the financial market conditions and will form the basis of my algorithm. The chapter then moves on to testing the concept of the RPA using Matrix Laboratory (MatLab) and provides key results that will aid the progression of the model. Chapter 3 closes with an attempt at the practical implementation of the RPA using FPGA hardware.

3.1 Introducing a novel bio-inspired approach to financial forecasting

Predicting the S&P 500 is no easy task because the index is an aggregation of 500 individual companies operating across all business sectors. Clearly previous values of the index have some relevance in predicting tomorrow's value, though historical data alone, as the review of published work in this area covered in Chapter 2 showed, is not sufficient. Also of relevance to the performance of each of the 500 companies in the S&P 500 is the financial climate that they are all operating within. Clearly the task of predicting the future value of the index must involve 'viewing' this extensive data set systematically to determine the most likely figure for the S&P 500 tomorrow, that is at (t+1).

In the early days of my research, Professor Toumazou and I were exchanging ideas of how best to devise a new predictive algorithm. During our conversation he told me about some novel research that had recently been undertaken on the design of an artificial retina (Nikolic, 2007). It was during this discussion that we both realised the parallels between the way the retina of the human eye functions and the way my algorithm to predict the S&P 500 needed to operate. The human vision system is capable of processing an enormous amount of visual data that it receives from the retina. The brain reviews all of this data but can still rapidly home in on a particular area of interest; this ability being called foveation. Turning to my task of predicting tomorrow's value of the S&P 500, the similarity is that the prediction needs to review a vast amount of data and rapidly home in on the area of interest that is most likely to cause the value of the index to alter with time. It was these functional similarities that inspired me to develop my algorithm, hence the use of the term 'bio-inspired algorithm'.

3.2 Initial concept for the proposed RPA

All financial prediction algorithms take historical financial data as inputs, evident from Chapter 2. The approach I have chosen is to take the S&P 500 historical closing prices together with a carefully selected number of historical financial market data (FMD) inputs, which will be outlined in detail below, that dictate the financial conditions within which all 500 companies operate. As the number of FMD inputs included will directly increase the computational time, it is likely that there is a trade-off between accuracy of prediction and computation time, hence the optimum number needs to be established. The time period of historical data will have a similar effect on computational time and accuracy; that is, an excessive historical window will absorb more computational time and the prediction accuracy is likely to plateau as the relevance to (***t+1***) diminishes when including historical data. Again, it is important to establish the optimum time frame.

With this architecture, the algorithm takes account of the key drivers that influence a change in the S&P 500. The result for the S&P 500 predicted closing price for (***t+1***) must be generated in less than 24 hours. This is likely to require a high speed computational architecture. Historically, human intervention has a significant role in forecasting the S&P 500 (Gilbert & Karahalios, 2010). This is undesirable when predicting the S&P 500 because the component companies are subject to change. For this reason, the goal for the RPA would be to have no external expert intervention.

3.2.1 Selection criteria for the chosen FMD inputs

Through my years of experience in the financial markets, it is my opinion that the performance of the financial markets is influenced by and fall into a number of categories including:

- Key stock indices
- Interest rate futures
- Exchange rates
- Commodity indexes

Initially, I formulated a lengthy list of contributing FMD inputs and through extensive economic research and my personal knowledge of financial markets, was able to refine the number of chosen FMD inputs to an optimum figure that will best position the model in achieving successful financial prediction of the S&P 500.

A comprehensive list of FMD inputs that may have an impact on the performance of the S&P 500 extended initially to 60. I decided that due to the very significant increase in possible combinations each time an additional FMD input is added to the model (bearing in mind the inevitable increase in computation time), I would need to reduce the total number of FMD inputs. Therefore, having considered the most significant FMD inputs, the final set was reduced to 51 by removing 9 FMD inputs considered to be least influential in affecting the S&P 500.

3.2.2 Chosen financial market data set

The 51 FMD inputs are defined below, together with their justification for inclusion in the RPA. A full list of all 51 FMD inputs with their corresponding values on a given day can be seen in Appendix 1.

NASDAQ100 (1)

The NASDAQ100 is a stock market index of 100 companies comprising the largest non-financial companies listed on the NASDAQ (National Association of Securities Dealers Automated Quotations). The companies' weights in the index are based on their market capitalisations, with certain rules capping the influence of the largest components. The index does not contain financial companies (banks or investment firms), but includes companies incorporated outside the United States (NASDAQ, 2014).

EURO STOXX 50 (2)

EURO STOXX 50 is a market capitalisation-weighted stock index of 50 large, blue-chip European companies operating within certain Eurozone nations. The index covers 50 stocks from 12 Eurozone countries, excluding the UK. The EURO STOXX 50 Index is licensed to financial institutions to serve as a barometer for a wide range of investment products such as Exchange Traded Funds (ETF), Futures and Options, and structured products worldwide (STOXX, 2014).

DAX (3)

The Deutscher Aktien Index (DAX) is a stock index created in 1988 that represents 30 of the largest and most liquid German companies which trade on the Frankfurt Exchange. The value of the index is based on a free-float weighted system and average daily volume (Bloomberg, 2014a).

CAC 40 (4)

The CAC 40, which takes its name from the system "Continuous Assisted Quotation", is the main stock index on the Paris market. The index represents a capitalisation-weighted measure of the 40 most significant values amongst the 100 highest market caps on the EuroNext Paris (EuroNext, 2014).

FTSE100 (5)

The Financial Times Stock Exchange 100 share index is an average of share prices in the 100 largest (based on market capitalisation) and most actively traded companies on the London Stock Exchange. FTSE 100 companies represent about 81% of the entire market capitalisation of the London Stock Exchange (London Stock Exchange, 2014).

Euro 6m LIBOR (6)

The 6-month Euro LIBOR interest rate is the average interest rate that a selection of banks in London are prepared to lend to one another in Euros with a maturity of 6-months. Alongside the 6-month euro LIBOR interest rate there is a large number of other LIBOR interest rates for different length maturities as well as in other currencies (Global Rates, 2014).

Euro 2yr Swap (7), Euro 5yr Swap (8), Euro 10yr Swap (9), Euro 20yr Swap (10), Euro 30yr Swap (11)

In an interest rate swap agreement, one party undertakes payments linked to a floating interest rate index and receives a stream of fixed interest payments with the second party undertaking the reverse arrangement. The interest rate swap rate denotes the fixed rate paid on a rate swap to receive payments based on a floating rate (PIMCO, 2014).

US$ 6m LIBOR (12)

The London Interbank Offered Rate is the average interest rate estimated by leading London banks that they would be charged if borrowing from other banks. The 6-month $US dollar LIBOR interest rate is the average interest rate at which a selection of banks in London are prepared to lend to one another in $US dollars with a maturity of 6 months (Global Rates, 2014).

US$ 2yr Swap A (13), US$ 5yr Swap SA (14), US$ 10yr Swap S/A (15), US$ 20yr Swap A (16), US$ 30yr Swap A (17)

In an interest rate swap agreement, one party undertakes payments linked to a floating interest rate index and receives a stream of fixed interest payments with the second party undertaking the reverse arrangement. The interest rate swap rate denotes the fixed rate paid on a rate swap to receive payments based on a floating rate (Swap-Rates, 2014).

JPY 6m LIBOR (18)

The London Interbank Offered Rate is the average interest rate estimated by leading London banks that they would be charged if borrowing from other banks. The 6-month Japanese yen LIBOR interest rate is the average interest rate at which a selection of banks in London are prepared to lend to one another in Japanese yen with a maturity of 6 months (Global Rates, 2014).

JPY 2yr Swap (19), JPY 5yr Swap (20), JPY 10yr Swap (21), JPY 20yr Swap (22), JPY 30yr Swap (23)

In an interest rate swap agreement, one party undertakes payments linked to a floating interest rate index and receives a stream of fixed interest payments with the second party undertaking the reverse arrangement. The interest rate swap rate denotes the fixed rate paid on a rate swap to receive payments based on a floating rate (Swap-Rates, 2014).

GBP 6m LIBOR (24)

The London Interbank Offered Rate is the average interest rate estimated by leading London banks that they would be charged if borrowing from other banks. The 6-month GBP LIBOR interest rate is the average interest rate at which a selection of banks in London are prepared to lend to one another in GBP with a maturity of 6 months (Global Rates, 2014).

GBP 2yr Swap (25), GBP 5yr Swap (26), GBP 10yr Swap (27), GBP 20yr Swap (28), GBP 30yr Swap (29)

In an interest rate swap agreement, one party undertakes payments linked to a floating interest rate index and receives a stream of fixed interest payments with the second party undertaking the reverse arrangement. The interest rate swap rate denotes the fixed rate paid on a rate swap to receive payments based on a floating rate (Swap-Rates, 2014).

Swiss 6m LIBOR (30)

The London Interbank Offered Rate is the average interest rate estimated by leading London banks that they would be charged if borrowing from other banks. The 6-month Swiss franc LIBOR interest rate is the average interest rate at which a selection of banks in London are prepared to lend to one another in Swiss frans with a maturity of 6 months (Global Rates, 2014).

Swiss 2yr Swap (31), Swiss 5yr Swap (32), Swiss 10yr Swap (33), Swiss 20yr Swap (34), Swiss 30yr Swap (35)

In an interest rate swap agreement, one party undertakes payments linked to a floating interest rate index and receives a stream of fixed interest payments with the second party undertaking the reverse arrangement. The interest rate swap rate denotes the fixed rate paid on a rate swap to receive payments based on a floating rate (Swap-Rates, 2014).

EUR/USD 1m forward (36), EUR/USD 3m forward (37), USD/CHF 1m forward (38), USD/CHF 3m forward (39), USD/JPY 1m forward (40), USD/JPY 3m forward (41), GBP/USD 1m forward (42), GBP/USD 3m forward (43)

A currency forward is a binding contract in the foreign exchange market that locks in the exchange rate for the purchase or sale of a currency on a future date, in this case, either 1 month or 2 months. It is essentially used as a hedging tool that does not involve any upfront payment. A major benefit of a currency forward is that it can be tailored to a particular amount and delivery period, unlike standardised currency futures (Currencies Direct, 2014).

WTI NYMEX (Oil 1m) (44), WTI NYMEX (Oil 3m) (45)

Light Sweet Crude Oil (WTI) futures and options are the world's most actively traded energy product. WTI is the deepest and most liquid global energy benchmark, trading nearly 850,000 futures and options contracts daily (CME Group, 2014a).

NYMEX (Gas 1m) (46), NYMEX (Gas 3m) (47)

Henry Hub Natural Gas futures allow market participants significant hedging activity to manage risk in the highly volatile natural gas price, which is driven by weather-related demand. They also provide efficient transactions in and out of positions (CME Group, 2014b).

Gold (48)

The bullion market is measured in $US per troy ounce. Investors generally buy gold as a hedge or harbour against economic, political or social fiat currency crises. The gold market is subject to speculation as are other markets, especially through the use of futures contracts and derivatives (BullionVault, 2014).

Goldman Sachs Commodity Index (49)

The S&P Goldman Sachs Commodity Index (GSCI) is a composite index of commodity sector returns demonstrating an unleveraged, long-only investment in commodity futures. The returns are calculated on a fully collateralised basis with full reinvestment. These attributes provide investors with an index representing a realistic picture of realisable returns attainable in the commodities markets (Goldman Sachs, 2014).

Long U.S. bond (50)

The 30-year U.S. Treasury Bond is a U.S. bond with the longest maturity; it has a coupon payment every 6-months as with treasury notes and are commonly issued with maturity of 30 years. The secondary market is highly liquid; therefore the yield on the most recent Treasury Bond offering was commonly used as a proxy for long-term interest rates in general (Investopedia, 2014).

VIX (51)

VIX is a trademarked ticker symbol for the Chicago Board Options Exchange Market Volatility Index. The VIX is a common measure of the implied volatility of S&P 500 index options. Often referred to as the fear index or the fear gauge, the VIX represents the most popular measure of the market's expectation of stock market volatility over the next 30-day period (Bloomberg, 2014b).

3.2.3 Historical data

Having chosen the 51 FDM points that most relevant in determining the S&P 500 at (***t+1***) and subsequently shown to be well-founded, the closing prices of all 51 FMD inputs over a 160-day period formulated the data set to test the concept of the RPA. I chose 160 days of historical data because it provides a data set large enough to establish a proof of concept of this model whilst maintaining a manageable set of data for processing.

3.3 Architecture of the proposed RPA

I propose a unique concept of a method, which compares the movement of the 51 FMD inputs over a chosen period of days, with the change in the movement between (***t***) (today) and (***t – 1***) (yesterday). The FMD inputs within the data sets that have the closest mean squared error (MSE) to the difference in the movement of (***t***) to (***t – 1***), are chosen and weighted based upon their movement to the S&P 500 at (***t***) to (***t – 1***), closest to the MSE. The results are then combined using the weightings to produce a number for the closing price of the S&P 500 at (***t+1***). This is then compared with (***t***); if the basic prediction is greater than (***t***), the S&P 500 will be forecast to increase compared with (***t– 1***); equally, if the number is lower, this signifies that the S&P 500 will fall in price. At this point there is no significance given to the size of the difference between the result and the closing price because I am initially only interested in forecasting the directional trend.

3.3.1 Potential implementation challenges

The scale of the proposed model could produce a number of complications that would need to be answered in order to successfully fulfil its intrinsic objective of predicting the S&P 500 at (***t+1***). As the number of FMD inputs used increases, the number of potential combinations would also increase – the same will also happen if I increase the number of historical days of data. Calculations suggest that, on average, for each added FMD input, the number of potential data combinations would increase exponentially. Table 3.1 displays the number of possible combinations against number of FMD inputs.

Table 3.1: Number of combinations against number of FMD inputs

Number of FMD Inputs	Number of Combinations
1	51
2	1,275
3	20,825
4	249,900
5	2,349,060
6	18,009,460
7	115,775,100
8	636,763,050
9	3,042,312,350
10	12,777,711,870
11	47,626,016,970
12	158,753,389,900
13	476,260,169,700
14	1,292,706,174,900
15	3,188,675,231,420
16	7,174,519,270,695
17	14,771,069,086,725

18	27,900,908,274,925
19	48,459,472,266,975
20	77,535,155,627,160
21	114,456,658,306,760
22	156,077,261,327,400
23	196,793,068,630,200
24	229,591,913,401,900
25	247,959,266,474,052
26	247,959,266,474,052
27	229,591,913,401,900
28	196,793,068,630,200
29	156,077,261,327,400
30	114,456,658,306,760
31	77,535,155,627,160
32	48,459,472,266,975
33	27,900,908,274,925
34	14,771,069,086,725
35	7,174,519,270,695
36	3,188,675,231,420
37	1,292,706,174,900
38	476,260,169,700
39	158,753,389,900
40	47,626,016,970
41	12,777,711,870
42	3,042,312,350
43	636,763,050
44	115,775,100

45	18,009,460
46	2,349,060
47	249,900
48	20,825
49	1,275
50	51
51	1

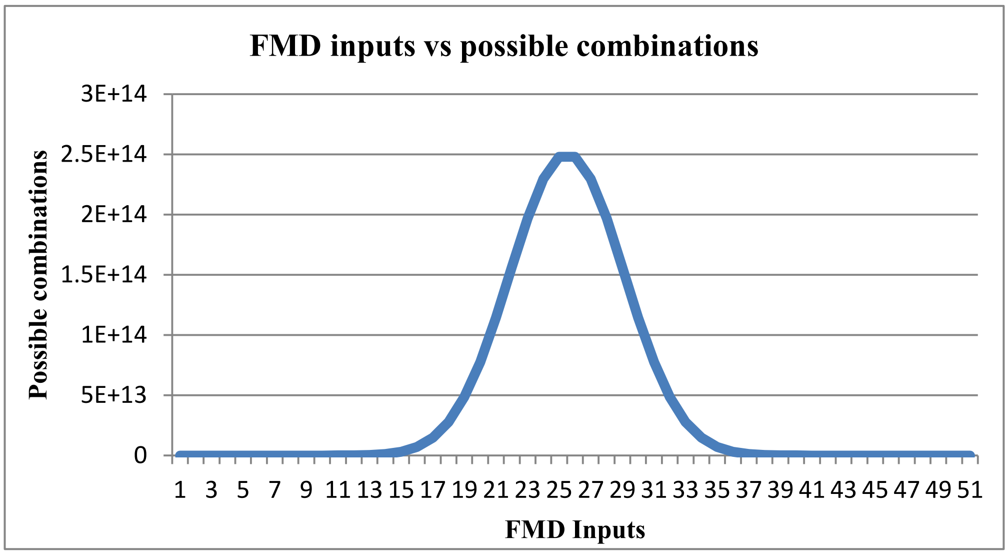


Figure 3.1: FMD inputs vs. possible combinations

Figure 3.1 is effectively the equation for the total number of combinations vs. FMD inputs:

(3.1)
$$^{n}C_{v} = NC = \frac{n!}{n!(n-v)!}$$

where…

NC = number of combinations (this is column 2 of table 3.1)

r = number of FMD inputs (this is column 1 of table 3.1)

n = 51

As a result of these growing numbers of data combinations, a significant concern exists over how such large sums of data will be processed in a time less than 24 hours, so that a buy/sell position is obtained at (***t+1***).

3.4 Concept – Explanation of RPA architecture

This section will provide an explanation of the RPA's architecture and how a prediction result is obtained for tomorrow's (***t+1***) S&P 500 closing price. Tests were calculated using the RPA running 6 FMD inputs and 30 days historical data.

The initial stages of the RPA process uses a data set of the daily closing prices of the chosen FMD inputs over the 30-day test period (the exact number of days historical data will be confirmed when an optimum number is established through testing).

Live data of the chosen 51 FMD inputs is attained from various sources including Reuters, Bloomberg and many more.

The core part of the RPA takes 7 sequences of 30 numbers: 6 of these are for the chosen FMD inputs from the total set of 51 (see Fig 3.3), and are the values of each chosen FMD input over days (***t+1*** to ***t+30***). The 7th sequence is the S&P 500 price over days ***(t+2*** to ***t+31***) (one day in advance of my data set because the RPA is predicting tomorrow).

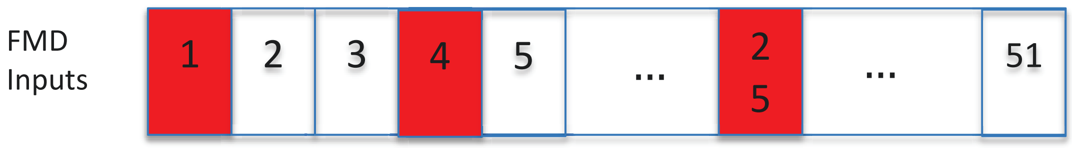


Figure 3.2: Layer of FMD input indexes

The RPA then runs a standard mathematical operation called Linear Regression which establishes the 'best fit' parameters between the 6th input sequence (inputs from time (***t***)), and the 7th (S&P 500 price prediction at time (***t+1***)) using a method called Ordinary Least Squares (OLS). OLS is a method for estimating the unknown parameters in a linear regression model. This method minimises the sum of squared vertical distances between the observed responses in the dataset and the responses predicted by the linear approximation.

The result of the linear regression is a series of coefficients that minimises the error value itself. These are the coefficients to the Linear Combination, an expression constructed from a set of terms by multiplying each term by a constant and adding the results, of the 6th input sequences with the minimum error when compared with input 7.

The error term is across the entire 30 day sequence. It is calculated for each day and added together, so for coefficients (c_0-c_6), and inputs (i_1-i_7) (i_7 is the predicted S&P 500 price tomorrow):

$$\text{Error} = \text{sum over all 30 days of } ((c_0 + c_1*i_1 + c_2*i_2 + \ldots + c_6*i_6) - i_7)^2$$

In words, multiply each input by its coefficient and add them together to attain the 'predicted' value, including the constant (c_0). The next stage would be to subtract the actual value (i_7) (S&P 500 tomorrow) and square it. This process is carried out for each day in the sequence. These figures are added together to ensure that there is a single error term for a particular set of 6 'best fit' inputs over 30 days. It is important that there is no concept of an error associated with any *individual* input, the calculation is performed purely over the set as a whole.

The framework therefore ensures that the RPA operates by testing different sets of 6 FMD inputs from the total set of 51. It performs the linear regression with every combination of 6 inputs (~18 million possible combinations using 6 FMD inputs) and selects the combination with the lowest error. It is a brute-force approach of trying every possible combination and due to the nature of this framework, computation takes a long time.

The assumption of the RPA is that the set of inputs with the minimum error over the training period (in this example, the 30 days used to generate the coefficients) will best predict the next day's price of the S&P 500.

At this stage I am able to use the lowest-error coefficients chosen above to predict the next out-of-sequence S&P 500 value. The same equation is used to attain the predicted value, with input values for (***t+30***):

(3.2) $$\text{prediction} = (c_0 + c_1 * i_1 + c_2 * i_2 + \ldots + c_6 * i_6)$$

I then compared the prediction with the S&P 500 for (**t+31**), which is effectively tomorrow ***(t+1***) and could then establish whether the result attained the correct directional trend.

Fig 3.3 displays a graphical representation of how each individual set of 6 FMD inputs will look when plotted onto a graph (which in the case of the RPA would include many more results) that displays their individual error term. The group with the lowest error and hence most likely to produce a correct directional trend result is highlighted in red.

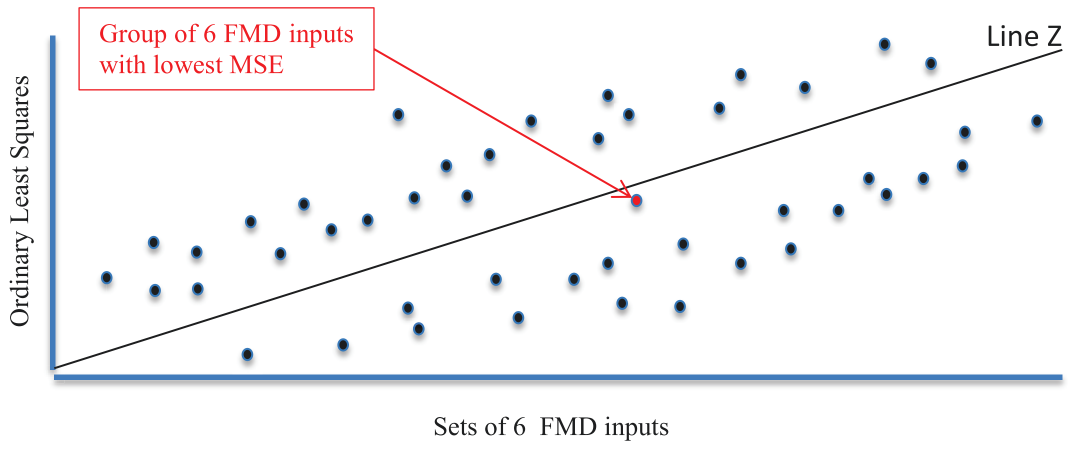


Figure 3.3: Error term of the sets of 6 FMD inputs to the price of the S&P 500.

This entire process is completed over the 98 days of data available to complete one full cycle.

RPA explanation Flow chart:

Establish a difference between the absolute value of the S&P 500 from yesterday to today (t-1 to t). The change shown as a percentage and a number.

↓

Algorithm selects which set of 6 FMD inputs (from the total set of 8 billion combinations) has the closest match (lowest error) to the price of the S&P 500 from (t-1 to t) (Line Z on Fig 3.3) over the 60 day period.

↓

Using circa 20 million combinations (calculating 51 FMD inputs & 60 days historical data).

Checking all possible combinations (e.g. on graph below) compared with the percentage movement of the S&P 500 from (t-1 to t) - which is represented as a zero error value (line Z).

↓

Fig 3.2 indicates the set of 6 FMD inputs with the least collective error relative the S&P 500 (t-1 to t). (This error value is formed of 6 non-linear lines representing the movement of each selected input over the 60 day period.

Therefore each selected FMD input has its own coefficient (error) over the 60 day period based on its closeness to the movement of the S&P 500 from (t-1 to t).

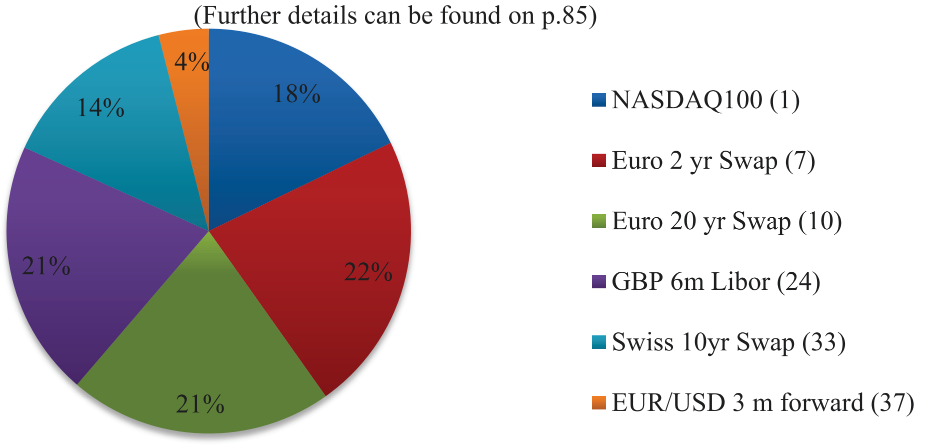
A pie chart can be formed to best represent this visually, based on the weighting of each FMD input made from the set of 6 inputs as a whole. This is formed through multiplying the error coefficient by the mean absolute value of the selected inputs price over the 60 days.

See the table below. For each selected input the error coefficient is multiplied by the average value of the input over 60 days. This forms that particular inputs segment within the pie chart.

FoM: 24/11/2008			
Chosen six FMD Inputs (No. corresponds to assigned FMD input number)	**Average value of FMD input over 60 days**	**Coefficient**	**Total**
NASDAQ100 (1)	1,848.25	0.405799	**750.02**
Euro 2 yr Swap (7)	4.15	-227.077882	**-942.06**
Euro 20 yr Swap (10)	4.60	193.35667	**888.95**
GBP 6m Libor (24)	5.76	149.769347	**862.44**
Swiss 10yr Swap (33)	3.29	-181.951768	**-599.49**
EUR/USD 3 m forward (37)	16.73	10.053887	**168.23**

FoM (Figure of Merit): one 60-trading day cycle ending 24/11/2008

(Further details can be found on p.85)



To predict the result for (t+1), multiply each of the 6 inputs average 60 day price by its given coefficient to again form a weighted absolute number. Add all 6 of these weighted absolute numbers (red totals column on table above) to form a single absolute number and compare it with the RPA calculation result at the previous trading day (t-1)

If the prediction result is greater than RPA prediction result at (t-1), the algorithm would suggest the S&P 500 will increase at (t+1) close, therefore a long position would be placed.

Conversely, if the RPA prediction result at (t) is lower than the RPA prediction result at (t-1), then a short position would be placed for the S&P 500 at (t+1). In this case I would be predicting the S&P 500 to fall in value.

Example RPA S&P 500 prediction for 24/11/2008

RPA calculation for today (t): 750.02 + -942.06 + 888.95 + 862.44 + -599.49 + 168.23 = **1128.09**

Therefore, this figure is compared with the RPA calculation from the previous trading day (Friday 21st Nov 2008).

1128.09 (prediction result for t) **- 1126.34** (prediction result for t-1) = **1.75**

1.75 is a positive number. Therefore in this instance I would place a long trade because the calculation infers the S&P 500 will increase at (t+1) relative to the movement from (t-1 to t).

Having formed a conceptual model for the RPA, this section will focus on proving its functionality. The initial testing was conducted using the numerical computing program MatLab. Whilst I recognise MatLab as being a vital program in proving the concept of this model, I am aware that it does not have the computational ability to handle large data sets, as would be required when processing the RPA to its fullest extent (see Fig 3.1). For this reason, MatLab will be used purely for validation purposes.

3.5.1 MatLab explanation

MatLab is a numerical computing environment and fourth-generation programming language (MathWorks, 2014). The program facilitates a wide array of numerical functions including plotting of functions and data, matrix manipulations (*ibid.*), and in the case of the RPA, MatLab will be utilised for the implementation of the algorithm.

With the MatLab language, algorithms can be developed faster than with traditional languages as it does not need to perform low-level administrative tasks including declaring variables, specifying data types, and allocating memory (*ibid.*). Using MatLab for algorithm development also has significant coding benefits; it is commonly understood that one line of MatLab code can often replace several lines of C or C++ code (*ibid.*).

Drawing on the paper by Houck et al. (1995) who successfully implemented their genetic algorithm using MatLab, I decided to use the program to prove the concept of my model, the RPA. MatLab provides an integrated capacity for deep and broad exploration of algorithm design options, but also caters for efficient deployment to desktop and embedded software environments (Klamt et al., 2007).

3.5.2 Implementation of RPA using MatLab

This section describes the MatLab implementation of the RPA algorithm proving the concept as well as its capability. In order to test the feasibility of the algorithm it was demonstrated on a 12-core windows PC using historical data for the factors chosen that influence the S&P 500. The program written allows me to define the number of FMD input parameters (that is 'parameters that affect S&P 500'), in addition to the number of parameters from these which will be used for the forecast of the next day (***t+1***).

The core program functionality is first to rotate the data array, using loops, capturing all possible combinations as in hardware, and then to run the parameters in the regression analysis module, which will come up with the forecasting weights for the given FMD inputs. For each rotation, or shifting, it therefore calculates the weights which will give the smallest error (MSE). In MatLab the results from the regression analysis of each pixel are received and then compared with the sum of residual error with the previous lowest. If the new result is lower, it stores the new error value and the new associated weights. Note that conventional regression systems would compute one regression result each time and compare error values one after the other (similar to raster scanning), while the RPA processes a batch of regression parameters in parallel and selects the lowest error amongst all. This effectively speeds up the process without losing any accuracy, which provides a significant advantage against other methods such as ANNs. Fig 3.4 depicts this process of finding the optimal FMD inputs of the RPA:

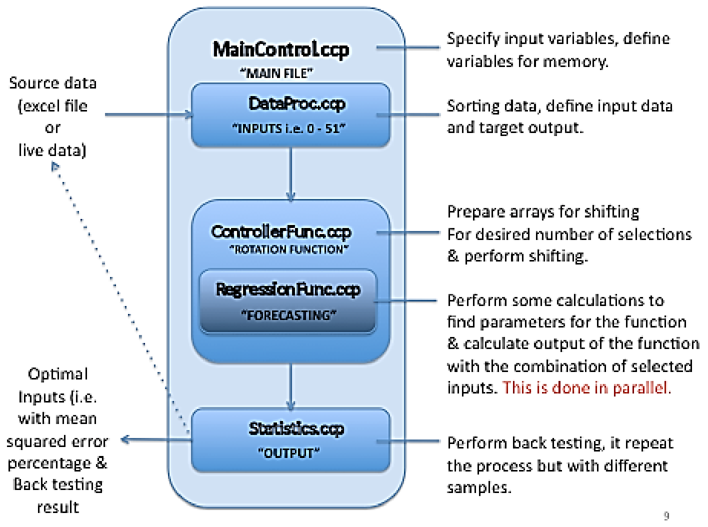

MainControl.ccp
"MAIN FILE"
Specify input variables, define variables for memory.
Source data (excel file or live data)
DataProc.ccp
"INPUTS i.e. 0 - 51"
Sorting data, define input data and target output.
ControllerFunc.ccp
"ROTATION FUNCTION"
Prepare arrays for shifting For desired number of selections & perform shifting.
RegressionFunc.ccp
"FORECASTING"
Perform some calculations to find parameters for the function & calculate output of the function with the combination of selected inputs. This is done in parallel.
Optimal Inputs (i.e. with mean squared error percentage & Back testing result
Statistics.ccp
"OUTPUT"
Perform back testing, it repeat the process but with different samples.
9

Figure 3.4: Flow diagram of model implemented on MatLab

3.5.3 Initial implementation and 'proof of concept' of the RPA using MatLab

I conducted a series of tests that demonstrated the concept of my algorithm using 19 parameters which affect the S&P 500, from which the best three (ones with the lowest error) were chosen to give a forecast of the next day's percentage movement (***t+1***). I chose to run my 'proof of concept' testing using the 19 best fit FMD inputs (instead of the 51 FMD inputs) reduced because of the computational time limitations of MatLab.

As displayed in Fig 3.1 the number of FMD inputs is positively correlated with the number of possible combinations, which severely increases the computational time. Using 19 FMD inputs presents a manageable computational time whilst also ensuring a proof of concept can be attained. This test is computed using 30 days historical data, as it represents one month and hence a manageable time period both to compute, and use as a basic comparison when analysing and comparing results. Fig 3.5 displays the output of the RPA running in MatLab.

What can be seen in the top window is the processing of the 19 parameters, whereby the Y-axis shows which of the 3 parameters are selected. Each vertical combination of these dots is a combination of FMD inputs, which would feed to the regression analysis model calculating in parallel.

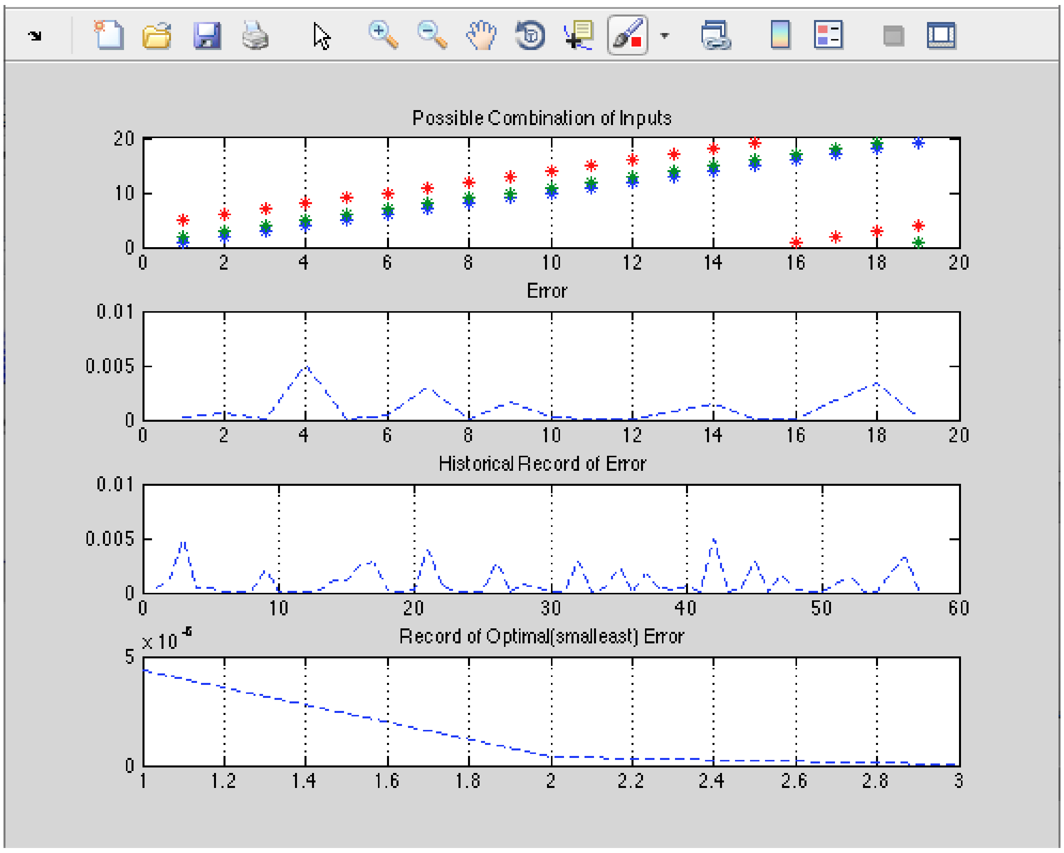


Figure 3.5: The current state of shift registers (top); the regression errors of the current state (2nd); the historical record of error (3rd); the record of the optimal solution shown by the smallest error (lowest)

The simulation progress of my RPA as it is running is then shown in the rest of the figures, with the lowest showing the minimum error which has been achieved.

Fig 3.6 shows the final results using my RPA, whereby the top graph indicates the error calculated for each combination and the lowest the smallest error achievable. Based on the smallest error calculated, shown in the bottom graph, the output then shows the three parameters which achieved this.

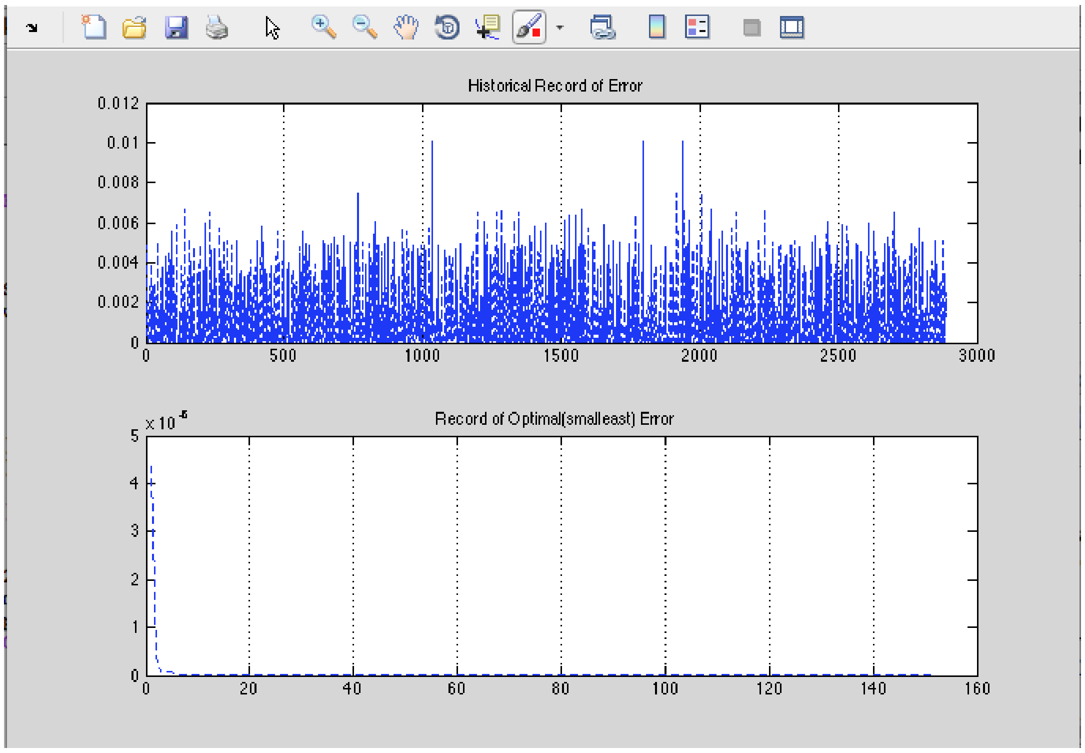


Figure 3.6: Final calculation of minimum error using the RPA

For instance in the example shown in Fig 3.5 and Fig 3.6, the output of the RPA provided the following:

- The optimum weights are: −4.0238 0.06805 9.6072
- The FMD input columns: (8 [Euro 5 yr swap], 13 [US$ 2m Swap], 6 [Euro 6m LIBOR]) are selected in linear regression model with minimised MSE: 9.3302e−10
- What we can see clearly is that the FMD input columns 8, 13 and 6 with weights −4.0238, 0.06805, 9.6072 gave the lowest error with 9.33e−10.

To use this to forecast 'tomorrow' (***t+1***) I would now compute y_{t+1} using the following equation:

(3.3) $$y_{t+1} = a_1x_1 + a_2x_2 + a_3x_3$$

(3.4) ***S&P 500(t+1)= −4.0238*x1(t)+0.06805*x2(t)+9.6072*x3(t)***

where $\boldsymbol{x_1}$ = FMD input 8, $\boldsymbol{x_2}$ = FMD input 13 and $\boldsymbol{x_3}$ = FMD input 6.

S&P 500 prediction (***t+1***) = if number is higher than yesterday, the RPA will predict an upward trend. Lower = downward trend.

3.5.4 Further implementation and feasibility of RPA (using MatLab)

Having proved concept of the RPA in Section 3.3.4, I will now further test the RPA using all 51 FMD inputs and also extend the testing results to include the top parameters over 3 separate 30-day periods. These results provide a full test for the initial proof of concept. The feasibility of the algorithm is displayed in Table 3.2 and confirms the functionality of the algorithm in achieving a directional efficiency of greater than 50% for a minimum of 6 FMD inputs based on 30 days of data from October 2005.

Table 3.2 Directional results, accuracy and speed of algorithm on a PC- platform in October 2005

FMD input data	1	2	4	6
Directional Result	11/30	14/30	15/30	17/30
RMSE	28.57	33.02	24.784	19.446
3Ave. Time Elapsed (minutes)	0.02760	.66196	16.3922	36,096.65

Having found the optimum number of FMD inputs, 6 FMD inputs is the minimum to give me greater than 50% directional trend; therefore I will use 6 FMD inputs as my testbench. One of the massive limiting factors of these results is the time it took to compute; on a 12 core PC this took over 25 days. In order to verify the validity of 6 parameters statistically, I have run this testbench in MatLab on 3 separate sets of data:

Table 3.3: January 2000

Selected FMD Inputs					
1	12	18	21	23	51
1	12	18	21	23	51
1	9	18	26	33	51
1	9	23	29	31	33
1	9	23	29	31	33
1	13	15	18	31	33
1	13	15	18	31	33
1	9	19	21	23	51
1	9	21	23	26	33
1	9	21	23	26	33
1	11	12	26	32	33
1	9	21	23	27	33
1	9	21	23	27	33
1	9	21	23	25	33
1	9	21	23	25	33
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	18	21	25	33
1	11	18	21	25	33
1	11	18	24	33	34
1	11	18	24	33	34
1	11	19	21	23	25
1	11	19	21	23	25
1	11	19	21	23	25
1	11	18	24	33	34

Directional trend accuracy	MSE	Time to compute (mins)
16	32.50965642	36,100.12904

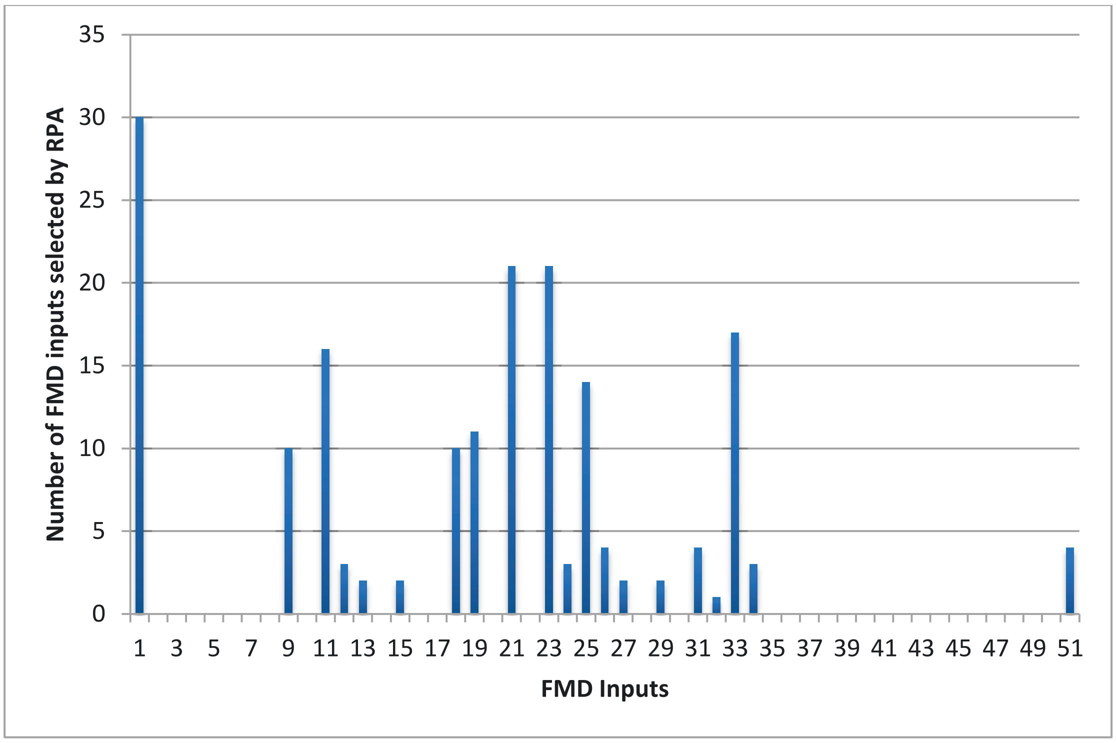


Figure 3.7: January 2000 – Individual FMD inputs selected by RPA

Top three parameters:

1. NASDAQ100

2. JPY 30yr Swap

3. JPY 10yr Swap

Results in Table 3.3 for January 2000 display a directional accuracy of 16 out of 30 with a MSE of 32.5. Therefore, because the directional accuracy was greater than 50% out-of-sample (in this case 53%), the RPA proved successful over this data set. Fig 3.7 indicates the number of times each parameter was selected over the 30 days testing period. For January 2000 the top parameter was again the NASDAQ100, which was selected for all 30 days. The graph shows that JPY 30yr Swap and JPY 10yr Swap were the other two top selected parameters.

Table 3.4: October 2005

Selected FMD Inputs					
1	7	25	33	36	37
1	3	4	7	33	36
1	3	4	7	33	36
1	2	3	26	33	38
1	2	3	26	33	38
1	2	3	25	33	38
1	2	3	25	33	38
1	2	3	26	33	38
1	2	3	26	33	38
1	2	3	26	33	38
1	2	3	26	33	38
1	5	7	33	36	37
1	2	3	26	33	38
1	7	20	33	36	37
1	4	20	30	36	51
1	4	20	33	36	51
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	4	18	25	33	36
1	18	30	31	42	48
1	18	30	31	42	48
1	26	36	41	48	50
1	27	36	39	41	48
1	27	36	39	41	48

Directional trend accuracy	MSE	Time to compute (mins)
17/30	19.44636487	36,096.65125

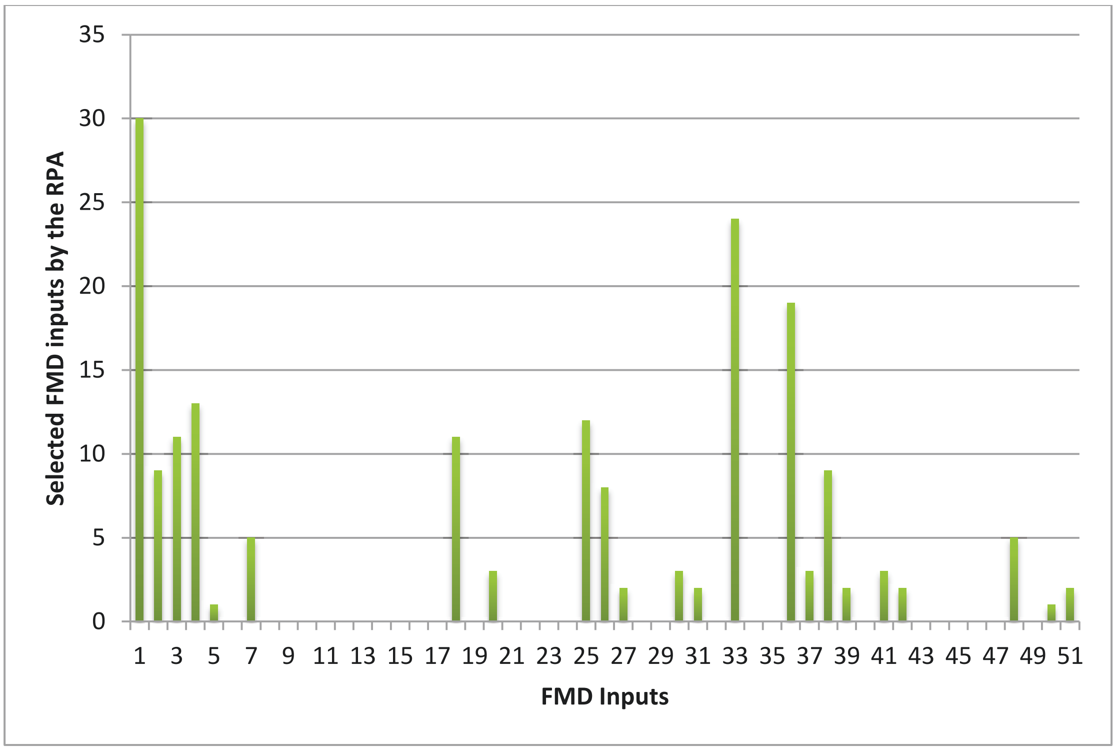


Figure 3.8: October 2005 – Individual FMD inputs selected by RPA

Top three parameters:

1. NASDAQ100

2. EUR/USD 1 m forward

3. Swiss 10yr Swap

Results in Table 3.4 for October 2005 display a directional accuracy of 17 out of 30 with a MSE of 19.4. Therefore, because the directional accuracy was greater than 50% out-of-sample (in this case 63%), the RPA proved successful over this data set. Fig 3.8 indicates the number of times each parameter was selected over the 30 days testing period. For October 2005 the top parameter was again the NASDAQ100, which was selected for all 30 days. The graph shows that EUR/USD 1 m forward and Swiss 10yr Swap were the other two top selected parameters.

Table 3.5: September 2008

Selected FMD Inputs					
1	8	11	25	34	38
1	9	11	25	30	38
1	9	11	25	30	38
1	8	11	25	33	38
1	9	11	25	30	38
1	9	11	25	30	38
1	9	11	25	30	38
1	9	11	25	30	38
1	6	25	28	29	38
1	6	25	28	29	38
1	6	25	28	29	38
1	6	25	28	29	38
1	6	25	28	29	38
1	6	16	25	26	38
1	6	16	25	26	38
1	6	16	25	26	38
1	6	17	25	34	38
1	21	25	30	36	51
1	21	25	31	36	51
1	21	25	31	36	51
1	21	25	31	32	41
1	21	25	31	32	41
1	4	5	20	38	51
1	12	22	31	32	36
1	6	31	32	41	49
1	6	31	32	41	49
1	6	31	32	41	49
1	6	31	32	41	49
1	6	31	32	41	49
1	14	24	31	32	49

Directional trend accuracy	MSE	Time to compute (mins)
16/30	37.29831536	36251.22677

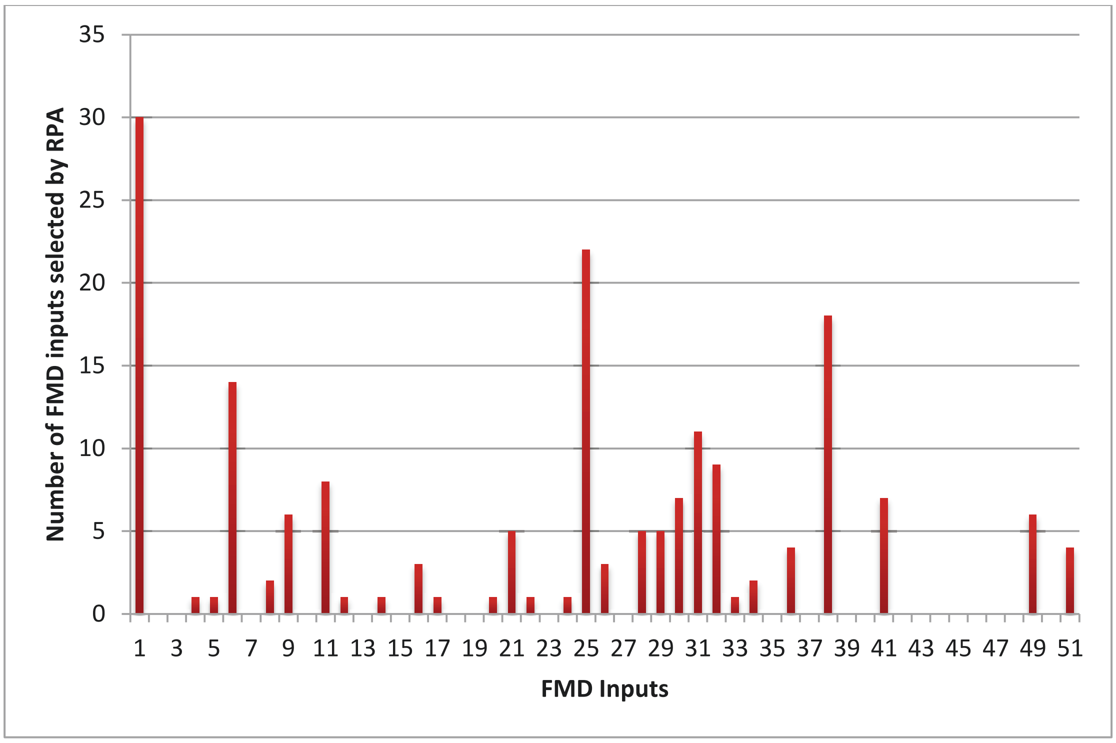


Figure 3.9: September 2008 – Individual FMD inputs selected by RPA

Top three parameters:

1. NASDAQ100

2. GBP 2yr Swap

3. USD/CHF 1 m forward

Results in Table 3.5 for September 2008 display a directional accuracy of 16 out of 30 with a MSE of 37.3. Therefore, because the directional accuracy was greater than 50% out-of-sample (in this case 53%), the RPA proved successful over this data set. Fig 3.9 indicates the number of times each parameter was selected over the 30 days testing period. For September 2008 the top parameter was the NASDAQ100, which was selected for all 30 days. The graph shows that GBP 2yr Swap and USD/CHF 1 m forward were the other two top selected parameters.

For all 3 tests, the time taken to find the 6 best-fit FMD inputs (with lowest closest MSE) was extremely long given the limitations of processing in MatLab. In order to fully exploit the potential of the algorithm and to conduct a more exhaustive study of the optimum number of FMD input parameters, it was decided that dedicated hardware such as an FPGA platform was needed.

3.6 Implementation of model using FPGA

In order to increase the number of FMD inputs above 6, it was necessary to input the algorithm onto a suitable hardware platform and a field-programmable gate array (FPGA) was chosen. However using the FPGA platform created a number of problems, the biggest was a limitation on the number of resources inside the FPGA. Rather than continue with the FPGA it was decided to move to a different hardware platform capable of handling the large data sets (more than 8 billion possible combinations) and successfully computing the next day's S&P 500 results within 24 hours. Although not specifically designed for, nor to my knowledge used for such a task, the Graphics Processing Unit (GPU) offers a hardware architecture that appeared potentially ideal for the RPA.

Chapter 4

RPA Hardware Implementation

4.1 Developments in parallel computing

Since the invention of the transistor by William Shockley (1949) in 1947 the developments in semiconductor technology, particularly the ever reducing dimensions of the transistor, have given rise to a rapidly expanding computer industry (Saxenian, 1991). The reducing dimensions of a single transistor together with developments of monolithic (single-chip) integrated circuits (ICs) has meant that the functional circuit blocks using several interconnected transistors, such as a memory element, has in turn decreased in size allowing greater and greater memory to be built per unit area of semiconductor; still almost exclusively silicon. Thus we have seen, and continue to see, rapid growth in computer power.

In 1965 Gordon Moore, a founder of the computer company Intel, stated that the number of micro-electronic devices, principally transistors, that could be placed on an integrated circuit was doubling roughly each year and he predicted that this trend would continue into the foreseeable future (Moore, 1965). This prediction became known as Moore’s Law and time has shown that Moore’s Law has been correct for many years, although the rate of change has slowed in recent years to a doubling approximately every 18 months (*ibid.*). Doubling the number of transistors on a silicon wafer means that the transistor size must halve, through extensive semiconductor research and development. Indeed, the transistor size has progressively reduced and it is now at the point where the feature size of the device is down to nano meters (Fang et al, 2013). This presents semiconductor engineers with several practical limitations. Firstly, the transistors can only operate at very low current levels due to the very large number of transistors per chip; secondly, leakage currents between transistors becomes so significant that neighbouring devices interfere with each other causing unreliable operation; and thirdly, production reliability means that the yield of perfectly functioning integrated circuits falls rapidly. It is therefore likely that Moore’s Law will not be true for much longer (Kaku, 1999).

The standard computer architecture has for many years been based on the original ideas of Alan Turing (Turing, 1936) sometimes referred to as 'the father of the computer'. The conventional computer architecture is based on a Central Processing Unit (CPU) with several associated memory blocks. This type of computer architecture is generally referred to as a CPU machine. The power of a given architecture computing machine is proportional to the speed that data can be handled. However, with the approaching limit of Moore's Law, recent computer system developments, where vast data is required to be processed rapidly, have necessitated different approaches to be explored (Keckler et al., 2011). A major driver of this has been the development of real-time graphics which is needed for high quality image presentations, required in the games industry and video graphics in general. Dedicated GPUs have been developed which offer significant advantages over and above the conventional CPU machine.

The GPU has a significantly different architecture enabling it to handling extensive parallel data sets easily at the high speed needed in advanced computer graphics applications. In my case the data handling capability of the GPU appeared to offer an ideal way forward for implementation of the RPA as the task in hand requires a fast machine able to compute a vast amount of financial data within 24 hours (or less) to obtain a prediction of tomorrow's S&P 500 value, today. The remainder of this chapter covers the GPU and its use for the implementation of the RPA, concluding with the presentation of early results.

4.2 Advancing the implementation method for the RPA

The concept of the RPA was successfully demonstrated using MatLab, as described in Chapter 3. However, we found that MatLab was limited in speed and data size, so unable to handle the extensive data that the RPA uses in predicting the S&P 500. Initially a dedicated hardware solution based on an FPGA implementation was considered. However, practical implementation of the FPGA proved to be overly complicated so further work using this hardware was halted and I decided to look for an alternative platform. The GPU appeared an attractive solution as it offered all the features that the RPA required. The key driver of GPU development has been the growing need for three dimensional, colour and moving images (QNX, 2014). All three require rapid parallel data handling, as does the RPA.

4.2.1 A solution to test a greater number of FMD inputs to establish an optimum

The original MatLab results using the September 2008 dataset ran 30 trials only and gave a directional test result of 16/30 (53.33%). The total runtime was 36,251 mins (~25 days). As will be discussed in full later, the GPU-optimised version of the algorithm, run with the same dataset and the same parameters, gives exactly the same results in only 359 secs (~0.1hrs). This is a speedup of over 100-times compared with the original proof of concept MatLab implementation. It should be noted that MatLab testing was kept to 30 trials because of the excessive computational time required.

Given the massive speed advantage of using a GPU, all subsequent work will allow for the maximum possible number of trials given the available dataset. Running the RPA with a GPU engine will overcome the computational limitations experienced using MatLab and allow the RPA to function fully, the goal being to obtain a solution within a 24-hour period. This development in GPU hardware implementation will allow a greater number of FMD inputs to be tested that will provide an optimum prediction result.

4.3 Graphics Processing Unit (GPU)

GPU architecture is highly parallel, with multi-thread co-processors resulting in a powerful computer which in handling data inputs in parallel results in a greatly reduced computational time (Nickolls, 2007). The significant difference between today's GPUs and traditional CPUs is that the GPU is designed for high throughput processing of many parallel operations over the lower latency execution of a single task within a CPU (Lee et al., 2010). Not only does the GPU perform better than the

CPU in terms of computational power, it also has a far greater memory bandwidth (*ibid.*).

GPUs comprise a large collection of fixed function and software programmable resources (Fatahalian & Houston, 2008). To enable programmers to develop applications with the GPU a software platform, such as NVIDA's CUDA (Compute Unified Device Architecture) (NVIDA, 2014) has been included. This platform is for massively parallel high performance computing on the NVIDA-based GPU and enables programmers to easily implement a highly parallel program (*ibid.*). Despite the origins of the GPU being used in graphics applications, particularly in video games, the GPU has been identified as an ideal computer engine for other applications which require greater parallel data handling with very high operational speed (Owens et al., 2008).

Dedicated graphics cards have been in use for many years, particularly for large data sets required in video signal processing, which in recent years have been essential for games programming (Fung & Mann, 2004). The graphics card processor has evolved into a dedicated GPU which is similar to a computer's CPU but with a dedicated role of handling graphical information (Owens et al., 2008). A GPU, however, is designed specifically for performing the complex mathematical and geometric calculations that are necessary for graphics rendering (Liu, 2014). A GPU is a specific electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display (NASA, 2014).

Today's generation of GPUs comprises a dedicated graphics card connected to its own CPU and is completely separate from the motherboard of the main computer (Che et al., 2008). The random access memory (RAM) is connected through the accelerated graphics port (AGP) or the peripheral component interconnect (PCI) express bus. Some GPUs are integrated into the northbridge on the motherboard and use the main memory as a digital storage area, but these GPUs are slower and have poorer performance (*ibid.*).

4.3.1 How a GPU differs from a CPU

The GPU is an electronic circuit unit that is designed to rapidly manipulate and modify memory to substantially increase the rate at which the system builds images in a frame (Coates et al., 2009). Its core purpose is to speed up the image building process that is intended for output to a display (*ibid.*). Modern GPUs are very efficient in manipulating computer graphics and their highly parallel structure makes it more effective than a general purpose CPU (as used when implementing the RPA in MatLab for testing) for processing large blocks of data that are handled in parallel (Lee et al., 2010). GPUs are more effective when it comes to manipulating computer graphics, because they handle data in parallel; a CPU commonly has 4 to 8 fast, flexible cores clocked at typically 3GHz, whereas a GPU has hundreds of relatively simple cores clocked at about 1GHz (Thiesen, 2010).

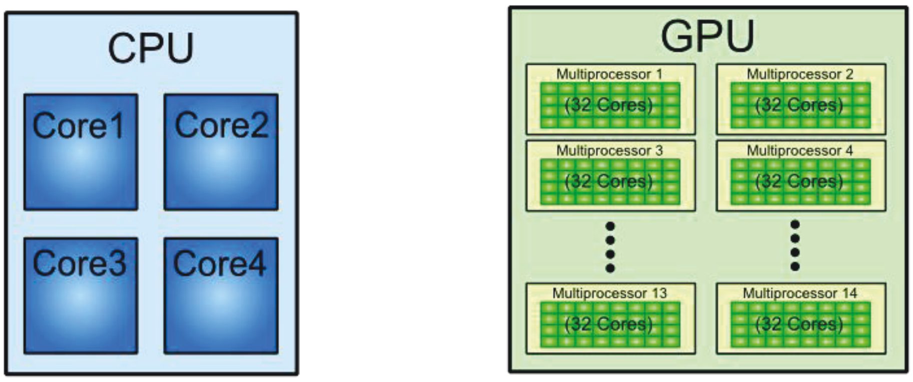


Figure 4.1 – CPU versus GPU architecture comparison (Thiesen, 2010)

Tasks that can be efficiently divided across many threads will see enormous benefits when running on a GPU. This highly parallel architecture is represented in Fig 4.1 and is the reason why a GPU can process such large batches of copy number data so quickly. Table 4.1 displays the most important differences between a CPU and a GPU.

Table 4.1 – Key differences between a CPU and GPU (Thiesen, 2010)

CPU	GPU
2 to 4 cores	Up to 1000 cores
Each core runs 1 to 2 independent threads in parallel	Each core runs around 1000 threads in parallel. All threads on a core must execute the same instruction at any time
Automatically managed hierarchy of caches	Each core has up 16-64kB of cache, explicitly managed by the programmer
0.1 billion floating-point operations / sec (0.1 TFLOP)	1 billion floating-point operations / sec (1 TFLOP)
Main memory throughput: 10GB/sec	GPU memory throughput: 100GB/sec

4.3.2 Unique approach: benefits of using GPUs in financial prediction

Having conducted extensive research on the demands required of financial prediction in Chapter 2, one specific correlation has stood out as a key problem when attempting to produce a successful algorithm that predicts the stock market. This correlation is amount of data input into the algorithm and time taken to compute the algorithm. Due to a growing data set to be included in the RPA, based upon increasing the number of FMD inputs and raising the number of days of historical data, the computational complexity of processing the RPA is also increasing. Fig 3.1 displays the number of possible combinations as each FMD input is added to the RPA; therefore it is imperative that a specifically tailored hardware is implemented that is able to handle such a large data set in a given time period. In the case of the RPA, this time period is tomorrow (***t+1***); the algorithm will produce a binary position (either a buy or sell position) that can be placed before the market opens tomorrow. Therefore, a mandatory function of the RPA is that it *must* be processed in less than 24 hours, i.e. before the markets re-open at (***t+1***).

Having explored a variety of implementation options with limited success, initially MatLab which successfully proved the concept and subsequently realised using an FPGA, operating the RPA using a specifically optimised GPU should eradicate the shortcomings of previous implementation attempts by providing the technological architecture that supports the processing of large data sets within the desired time frame.

Through extensive research in the development of GPU application, it became clear that the high technological demands of financial prediction could be met by implementing the RPA onto a specifically optimised GPU platform.

Modern GPUs use most of their transistors to compute calculations related to 3D computer graphics (Owens et al., 2008). Initially, GPUs were used to accelerate the memory-intensive work of texture mapping and rendering polygons, but later added units to accelerate geometric calculations including the rotation and translation of vertices into different coordinate systems. More recently, however, GPUs have developed to include support for programmable shaders (Blythe, 2006) that can manipulate vertices and textures with many of the same operations supported by CPUs, oversampling and interpolation techniques to reduce aliasing, and high precision colour spaces. Due to the fact most of these computations involve matrix and vector operations, engineers and scientists alike have increasingly studied the use of GPUs for non-graphical calculations (Crookes et al., 2009) and it is for this reason that I proposed to develop a novel architecture that would accommodate the RPA onto a GPU system.

4.3.3 Specifications of GPU system used to implement the RPA

Table 4.2 shows the full specifications of the Custom X79 i7 3930K System used to implement the RPA.

Table 4.2 – Specifications of hardware used to implement the RPA

Motherboard	Asus Rampage IV Formula Intel X79 (Socket 2011) DDR3 Motherboard
Memory	Corsair Dominator 32GB (4x8GB) PC3-12800C10 1600MHz Dual/Quad Channel Kit (CMP32GX3M4X1600C10)
Computer chassis	Silverstone Fortress FT02 USB3.0 Case - Black (SST-FT02B USB 3.0)
Hard drive	Intel 330 Series 120GB 2.5" SATA 6Gb/s Solid State Hard Drive – Retail
Cooling system	Cooler Master Seidon 120XL Watercooling System (Socket 775 / 1155 / 1156 / 1366 / 2011 / AM2 / AM2+ / AM3 / FM1 / FM2) OcUK 24x DVD±RW SATA ReWriter (Black) – OEM
Processor	Intel Core i7-3930K 3.20GHz (Sandybridge-E) Socket LGA2011 Processor – OEM
Power supply	Enermax Platimax 1500w '80 Plus Platinum' Modular Power Supply (EPM1500AWT)
Graphics cards	Three EVGA GeForce GTX TITAN Superclock Signature Edition 6144MB GDDR5 PCI-Express Graphics Card (06G-P4-2793-KR)

For the purposes of this RPA implementation, the key hardware components are the three top-of-the-range EVGA GeForce GTX TITAN graphics cards. Each card has 2688 cores, meaning that in total 3 x 2688 = 8064 calculations can be computed in parallel, compared with 12 on the 12-thread CPU (evga.com, 2014). Each GPU calculation takes longer than the CPU, but the massive parallelism more than makes up for this (Owens et al., 2008), leading to significant advantages overall.

4.4 Computational implementation and optimisation of the RPA onto GPU

In this section the concept of the RPA, as established and proved in MatLab (see section 3.3), is outlined stage by stage to underpin *how* the algorithm is processed and optimised onto a GPU, clearly displaying the speed of calculation for each step of optimisation, starting with pre-optimisation.

Pre-optimisation:

Initial speed of calculation cycles considering 4 FMD inputs = 5,500,000 ms = 5,500s

4.4.1 Step 1: Optimise existing C++ code on CPU (No GPU acceleration)

Objective:

Quick pass of the codebase focussing on ‘low hanging fruit’ to ascertain cheap speed-ups.

Description:

Main gains were achieved by reworking sections of the code-base to minimise duplication of data. Rather than having copies of the data being created and passed through the algorithm, the same data is used and referenced throughout the algorithm.

No changes were made to the algorithm order of execution. It was apparent that the algorithm could be made to execute significantly faster from parallelisation.

Results:

Code speeds up in line with expectations (not a fraction of our ultimate target performance) Step 1 optimised speed of calculation considering 4 FMD inputs = 250,380ms = ~250s

4.4.2 Step 2: Initial C++ AMP implementation to run on a GPU

Objective:

Convert the existing code to take advantage of C++ AMP (Accelerated Massive Parallelism). In doing so, this should allow me to run many blocks of computation in parallel on the GPU.

Description:

Converted the code-base to C++ AMP, running on a domestic GPU.

Results:

Firstly, I encountered a very unusual error. Once the code was moved to fully utilise the GPU, it would consistently crash after approximately 2 seconds. After several hours of attempting to find the cause of the crash it became apparent that the reason was deeper than just my GPU code.

Simply, the GPU was being stressed to such a degree by the relative onslaught of computation that there was no available runtime capacity for the GPU to actually update the screen on the PC – a function that a GPU is typically entirely concerned with.

I learnt that when this happens, there is a failsafe mechanism embedded deep in the operating system that resets the machine, once it senses that the screen cannot be refreshed. This was most unusual, but it did confirm that the use of a GPU for this type of significant algorithmic calculation was an unconventional use.

To resolve this and ensure I had 100% of the available power of the GPU(s) a cheap additional GPU card was purchased and set as the unit tasked to update the screen. This then resolved the crash and freed the main GPU to run computational cycles only.

This initial GPU implementation provided a reasonable speed up, but not to the level expected. The reason for this was the way the algorithm was being executed. Basically the GPU was being fed tasks too small for its architecture, and the majority of the time (95% +) was being spent waiting for the memory to feed it more data and catch up.

The traditional thinking around running efficient GPU compute processes is to give as much data up-front as possible and let the many compute cores on the GPU chew through the data. Given the CPU has fewer - but bigger - cores, along with a much bigger cache, this indicates that memory coherency is an issue. The CPU is much better at dealing with inefficient memory layouts due to the redundancy hardware built in to the chip.

Step 2 optimised speed of calculation considering 4 FMD inputs = 122,750 ms = ~123s

4.4.3 Step 3: Unroll the algorithm to feed the data sympathetically to the GPU

Objective:

Rework the algorithmic execution to unroll the main internal loop from a vertical execution to a horizontal execution by pre-loading all of the work for the GPU into memory. This approach will allow the entire workload to be as parallel as possible.

Results:

The actual time spent executing the algorithm has been reduced dramatically, but the memory usage is extremely high. Several gigabytes were used in a test run considering 5 FMD inputs or ‘characteristics’.

Step 3 optimised speed of calculation considering 4 FMD inputs = 57,900 ms = ~58s

4.4.4 Step 4: Re-engineer the algorithm to use the smallest dataset possible

Objective:

The best potential optimisation for the GPU implementation is to move towards using tiled memory (A C++ AMP term). What has become very apparent is the need to carefully organise the data for GPU efficiency. A traditional CPU is efficient at dealing with out-of-order data, and non-coherent data. A GPU is not good at dealing with this and cache misses are extremely expensive. However, when the data is organised, ordered and fed as efficiently as possible, the speed advance of the GPU is dramatic.

Estimate for speed of calculation considering 4 FMD inputs on GPU = 3,100 ms.

Additionally, the re-engineered algorithm is now considerably more predictable when observing increasing the number of FMD inputs. It is estimated that with a specifically purchased piece of hardware, budget £5000, that calculations considering 6 FMD inputs would take less than 4 minutes to compute.

This would be a *very significant* increase from the initial position where calculating 6 FMD inputs was taking over 3 weeks of computational time.

With the reasonable prediction that each additional FMD input will take around 10.5x longer, it would be possible to calculate 8 FMD inputs on a single computer in approximately 7 hours.

4.5 Effectiveness in implementing the RPA onto the GPU system

The use of a GPU, and the potential for spreading computation across multiple GPU boards simultaneously, allows for a significant speed increase over standard C++

CPU implementations. This is providing that the GPU code is extremely carefully written and the data is extremely carefully organised.

The reason for the entire multi-core GPU approach opposed to the bigger/faster CPU approach is depicted when drawing upon ‘Moore’s Law’ (Moore, 1965) which states that the number of transistors per square inch of integrated circuits had doubled every year since the integrated circuit was invented. This demonstrates the effective GPU approach that parallelising repetitive computation with many small cores (GPU) will outperform a single large processor (CPU) as scale increases.

4.6 Early results

The following results display the successful application of the GPU using the same data set as used when testing the concept of the RPA in Chapter 3. The same data set was utilised firstly, to test the hardware by ensuring the GPU produced the same results as the MatLab results; secondly, this data set was used to provide a time comparison. The reason MatLab could not be used to implement the RPA was due to the huge processing times, something that implementing the RPA using a GPU has addressed.

The original MatLab results with the September 2008 dataset ran 30 trials only, and gave a directional test result of 16. The total runtime was 36251 mins (~25 days).

The GPU optimised version of the algorithm, run with the same dataset and the same parameters, gives exactly the same results and takes 359 secs. This is a speedup of 101 times over the original implementation and produces the same directional accuracy.

For GPU testing purposes, I ran the RPA under the same paramaters as used on MatLab in order to directly compare results and test the functionality of the GPU. Results using the GPU were identical to MatLab but with a significantly faster processing time, as displayed in Fig 4.2. It is also interesting to compare the results on Fig 4.3 of the GPU with the dedicated GPU that has been optimised to produce significant time reductions whilst maintaining the same directional result.

PC running MatLab					
51 FMD input data	**1**	**2**	**4**	**6**	**8**
Directional Result	11/30	14/30	15/30	17/30	n/a
RMSE	28.57	33.02	24.784	19.446	
Ave. Time Elapsed (mins)	0.03	0.66	16.39	36,096.65	-

GPU Card					
51 FMD input data	**1**	**2**	**4**	**6**	**8**
Directional Result	11/30	14/30	15/30	17/30	n/a
Ave. Time Elapsed (secs)	-	-	17.00	1,860.00	-

Dedicated (optimised) GPU Card					
51 FMD input data	**1**	**2**	**4**	**6**	**8**
Directional Result	11/30	14/30	15/30	17/30	n/a
Ave. Time Elapsed (secs)	-	-	3.10	240.00	25,200.00

Figure 4.2 – Directional results, accuracy and speed of RPA when processed using MatLab, GPU card and Dedicated GPU card

These results hence proved the success of the GPU as the correct hardware to run the algorithm and allowed me to explore implementing the RPA with a larger number of FMD inputs. The results of running the RPA using up to 9 different FMD inputs to achieve a buy/sell position on the S&P 500 is outlined in Chapter 5.

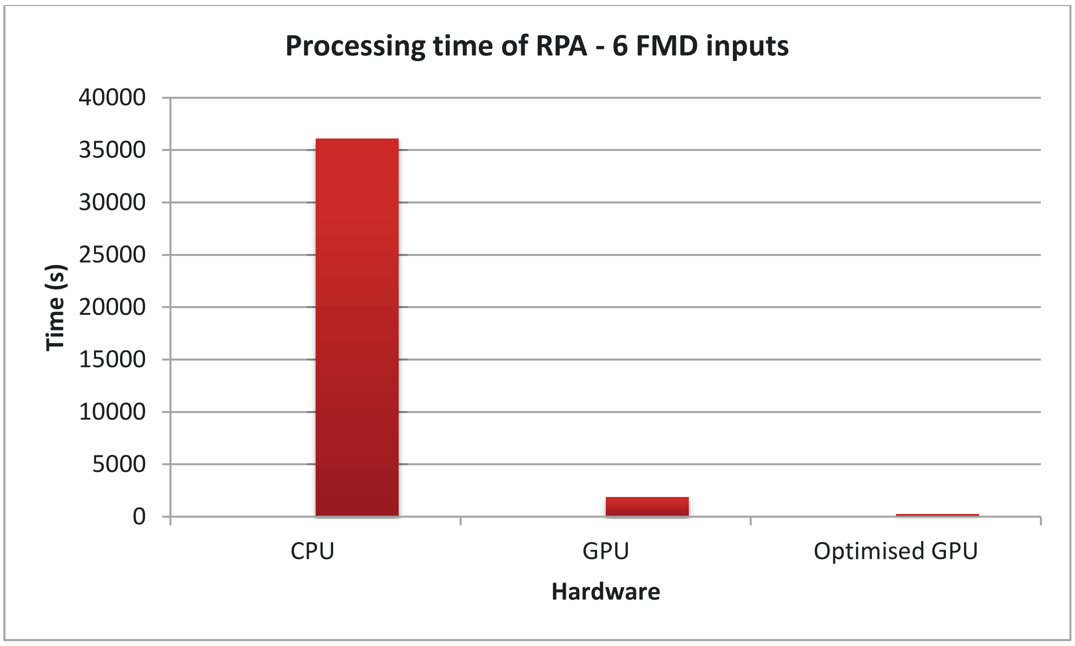


Figure 4.3 – Processing time of the RPA when implemented in three separate hardwares

Chapter 5

Initial Results: Testing Optimum Number of FMD Inputs

5.1 Importance in establishing the optimum number of FMD inputs to successfully predict the S&P 500 at (t+1)

In this chapter, I intend to use the newly implemented and optimised GPU architecture to confirm that increasing the number of FMD inputs should have a positive correlation in forecasting the directional trend of the S&P 500 for (***t+1***). The focus of this chapter is to establish at what point increasing one extra FMD input used to produce a buy/sell position for predicting the S&P 500 will lead to a detrimental effect in the directional trend forecasting.

A further aim of this chapter is to establish which FMD inputs are most regularly selected when the RPA is computed through the optimised GPU and are hence most closely correlated to the price of the S&P 500. Attaining this figure of merit is vital in understanding the relationship between my chosen 51 FMD inputs and the S&P 500 index and allow me to produce a ratio of each selected FMD input for every cycle computed through the GPU .

I have chosen to use a 120-day data set because it represents a manageable time period that can be computed within the time constraints. This data range also displays a fair representation of the financial markets as it would produce 98 individual cycles which together consist of ***X*** possible combinations increasing rapidly as the number of chosen FMD inputs rises – as displayed in Fig 3.1.

The final goal is to predict correctly the trend of the S&P 500 at (***t+1***), so the processing time for each set of cycles must be completed within 24 hours so that an open position (buy or sell) can be placed in time for the market open.

‘In-sample’ and ‘out-of-sample’ are terms frequently used in financial prediction systems to describe (i) historical/past (known) data (in-sample) and (ii) future (unknown) data values. Development of financial forecasting algorithms that depend entirely on historical data is referred to as ‘data mining’, or ‘in-sample modelling’. Data mining is said to occur when a researcher reviews alternative forecast models, but only reports results for the specification with the highest predictive content (Inoue & Kilian, 2002). In the context of predictive inference, the underlying concern is that in-sample tests of predictability may spuriously indicate predictability when there is none. In this context, a predictability test would be considered unreliable if it has a tendency to reject the no predictability null hypothesis more often than it should at the chosen significance level. Algorithms that rely entirely on ‘in-sample modelling’ are effectively trend based; Granger (1990: 3) writes, “One of the main worries about the present methods of model formulation is that the specification search procedure produces models that fit the data spuriously well, and also makes standard techniques of inference unreliable.” Many authors (see Chapter 2) have attempted to use an in-sample approach with moderate success. However, the fundamental assumption that the future is entirely determined by the past misses the fact that an index such as the S&P 500 is an aggregation of the performance of 500 individual companies. In recognising this limitation, I decided to develop an algorithm based on an alternative methodology that exploits a technique that I shall refer to as an out-of-sample technique, where the term ‘out-of-sample’ now refers to additional known financial data that strongly influence the S&P 500 companies.

5.1.1 Results: Optimum number of FMD inputs for the RPA

In order to test the number of FMD inputs that represent the optimum performance of the RPA in predicting the S&P 500, it was necessary to establish the best possible range of historical data to use. Therefore, I programmed the GPU to run the RPA to predict the price of the S&P 500 using 4 and 6 FMD inputs. This provided a manageable data set that can be computed under a given timescale whilst also large enough to ensure that results are statistically significant. This pre-testing is useful in strengthening the framework in establishing the RPA’s optimum number of FMD inputs. Table 5.1 displays these test results. Tests were carried out 9 times, using a different number of days historical data (2, 5, 10, 20, 30, 40, 50, 60 and 70 days) for each test set and thus establishing the optimum number of days historical data to run the RPA on.

Tests were carried out using a total data set of 120 days which allowed for 99 trials when using 2 to 60 days historical data. To keep the number of trials as an even number and hence more manageable for comparison, I decided to reduce the number of trials to 98.

Due to the larger number of days when testing 70 days historical data, the maximum number of trials possible (within the 120-day data set) was 89 but again reduced to 88 trials to create a figure more manageable for comparison.

The results in Table 5.1 clearly display that using 60 days of historical data is the optimum. Using both using 4 FMD inputs (51% directional trend) and 6 FMD inputs (53.1% directional trend), the 60-day historical data represented the most accurate set and will therefore be used as the fixed data set to test the optimum number of FMD inputs. Tests calculating the optimum number of historical days data will be re-established in Chapter 6, after an optimum number of FMD inputs is attained for the RPA.

Table 5.1: Establishing optimum number of days historical data to test FMD inputs – 4 and 6 FMD inputs

4 FMD inputs									
Days of data	2	5	10	20	30	40	50	60	70
Directional test correct	50/98	50/98	47/98	49/98	50/98	48/98	43/98	50/98	44/88
Correct %	51.0	51.0	48.0	50.0	51.0	49.0	43.9	51.0	50.0

6 FMD inputs									
Days of data	2	5	10	20	30	40	50	60	70
Directional test correct	52/98	45/98	44/98	46/98	51/98	49/98	45/98	52/98	44/88
Correct %	53.1	45.9	44.9	46.9	52.0	50.0	45.9	53.1	50.0

In establishing the optimum number of days historical data to use in testing, I was then able to use this data to ascertain the optimum number of FMD inputs used to compute the RPA. The optimum figure being that which gives the highest directional trend accuracy in predicting the S&P 500.

Testing the optimum number of FMD inputs was carried out using the same architecture as testing the optimum number of days. Again, I used a 120-day data set, using the optimal 60 days historical data for each individual cycle – hence allowing for a maximum of 99 cycles per FMD input test. The 99 cycles were reduced to 98 for ease of comparison when analysing the data results.

My initial hypothesis, built through my acquired knowledge of the financial markets, was that as the number of FMD inputs used to produce a prediction for the S&P 500 increases, so will the directional trend. I also felt that there would be a point where increasing the number of FMD inputs, and hence number of possible combinations, would break from its positive correlation and start to create ‘financial noise’ as the number of combinations being computed into the GPU soared (see Fig 3.1).

Looking at Fig 5.1, it clearly shows that my initial hypothesis was correct to a certain extent. In this series of testing there was a positive correlation between the number of FMD inputs and the directional trend except for when using 8 FMD inputs, which I have put down as an anomaly.

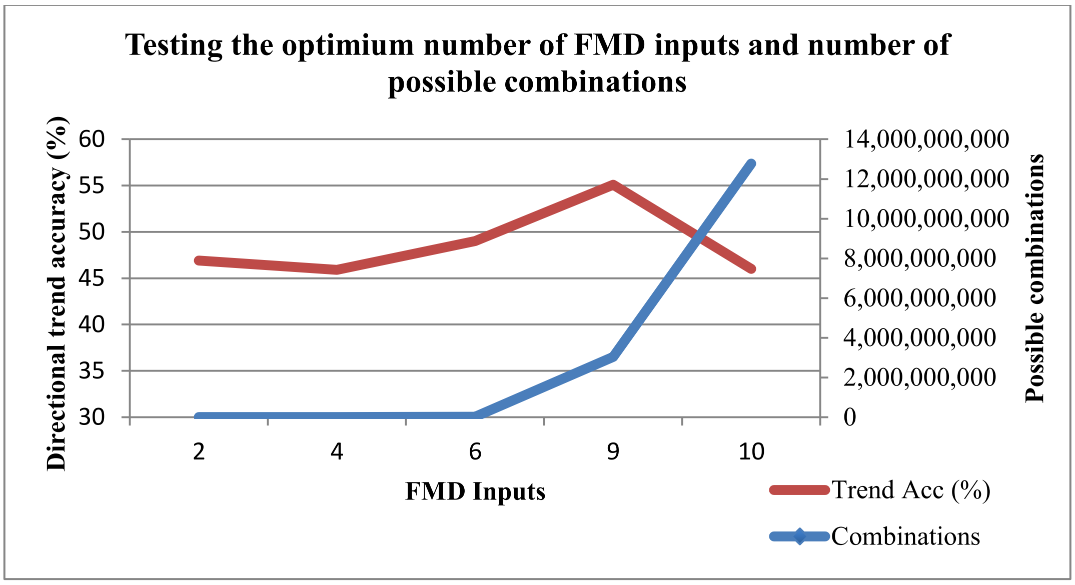


Figure 5.1: Optimum number of FMD inputs vs. number of possible combinations

Table 5.2 displays the key information when programming the RPA with 9 FMD inputs which has been established as the optimum number.

Table 5.2: Optimum number of FMD inputs

Optimum FMD inputs	**9**
Num trials	**98**
Directional test correct	**54**
Correct %	**55.1**
Binomial probability	**.182**
MSE	**34.77**
MSE%	**4.405**

A Figure of Merit (FoM) is a quantity used to characterise the performance of a device, system or method, relative to its alternatives. Applying this theory to the RPA, finding a FoM for each individual cycle will enable me not only to establish which FMD inputs were chosen in that particular cycle, hence representing the FMD inputs with the lowest MSE against (***t***), but also to understand the weightings of this error based on the group of selected FMD inputs as one whole cycle. This proportion allows me to understand the contribution of each individual chosen FMD input against the chosen set as a whole.

The FoM is achieved by attaining the average price of each chosen FMD input over the prior 60 days (optimum number of days historical data to use, as will be established in chapter 6). This figure is then multiplied by each relevant 'coefficient' (the coefficient acts to normalise the data so a comparison can be attained). This 'total' acts as an overall weight of each 'chosen FMD input' in relation to the 'chosen FMD input set' as a whole. The sum of the 'total' figure represents 100% of the data set, hence the pie chart acts as a proportional representation of each chosen FMD input for a selected day. Table 5.3 displays a data set of the RPA ending on 24th November 2008 and shall be used to explain how a FoM works.

Table 5.3: FoM for the 60 days ending 24th November 2008 using 6 FMD inputs

FoM: 24/11/2008			
Chosen six FMD Inputs (No. corresponds to assigned FMD input number)	**Average value of FMD input over 60 days**	**Coefficient**	**Total**
NASDAQ100 (1)	1,848.25	0.405799	**750.02**
Euro 2 yr Swap (7)	4.15	-227.077882	**942.06**
Euro 20 yr Swap (10)	4.60	193.35667	**888.95**
GBP 6m Libor (24)	5.76	149.769347	**862.44**
Swiss 10yr Swap (33)	3.29	-181.951768	**599.49**
EUR/USD 3 m forward (37)	16.73	10.053887	**168.23**

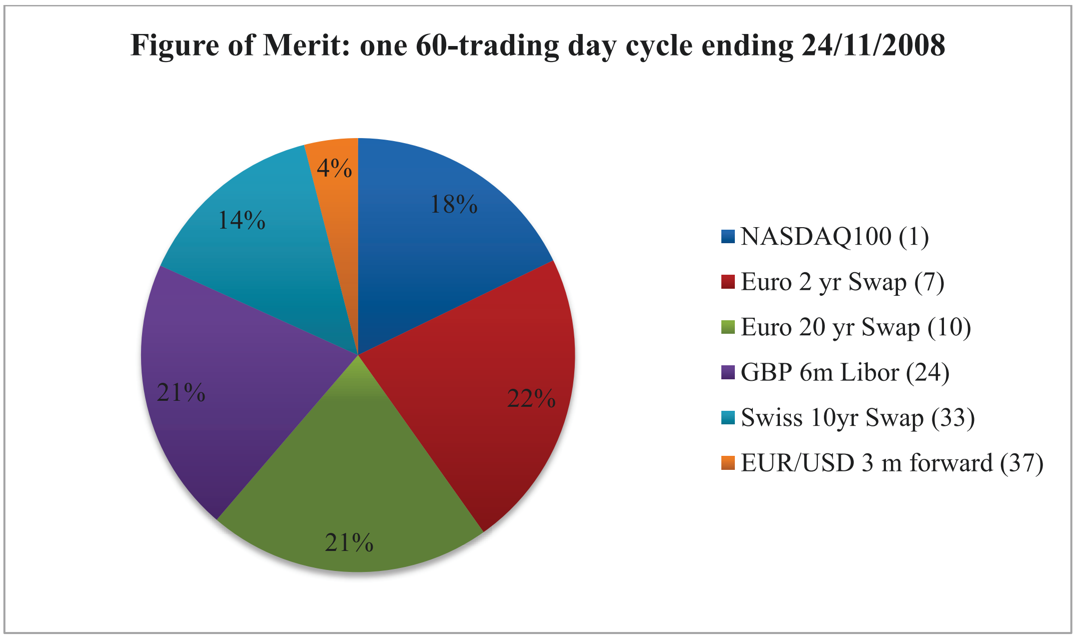


Figure 5.2: Pie chart displaying figure of merit for one cycle using 60 days historical data

Fig 5.2 displays the weightings of the 6 FMD inputs (out of the total set of 51) that produced the lowest MSE over the previous 60-day period relative to today's (***t***) closing price of the S&P 500. This figure then provides the basis for my prediction of the S&P 500 tomorrow (***t+1***). For example, in Fig 5.2, over the 60 (trading) day cycle ending on the 24th November 2008, Euro 2 yr Swap rate (FMD input 7) was processed to be the most influential FMD input (lowest error over the 60-day period relative to the price of the S&P 500 today (***t***), contributing to a 22% weighting of the whole set, which formed the basis in the prediction of buy or sell position of the S&P 500's market closing price on 25th November 2008 (***t+1***). Euro/USD 4-month forward contracts (FMD input 37) was found to produce the least significant influence, only contributing 4% to the total weighting of the set of results used to predict tomorrow's (***t+1***) closing price of the S&P 500.

5.2 Significance of each set of chosen FMD inputs

The chosen 51 FMD inputs fall into 4 core categories: key selected indices; selected currency pairs; selected swap rates with varying maturity rates; and selected commodity indices. A possible explanation for the correlation is as follows.

Other indices

NASDAQ 100

The NASDAQ 100 is a modified capitalisation-weighted index based on the 100 largest non-financial companies. Many of the component companies within the S&P 500 are included in the NASDAQ100. The NASDAQ 100 is often viewed as a more polarised version of the S&P 500 as the majority of the NASDAQ100 component companies are included in the S&P 500. This close correlation is clearly shown in Fig 5.3 and Fig 5.4 with the S&P 500 which has less volatile movement due to a larger number of component companies creating more financial noise and so less reactive to changing market conditions.

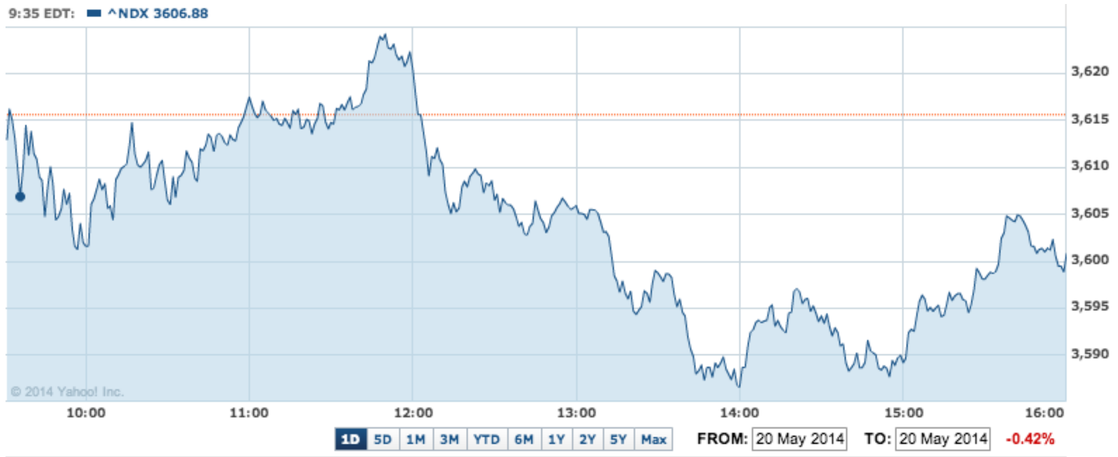


Figure 5.3: NASDAQ 100 (1 day) price chart 20.05.2014 (Yahoo Finance, 2014a)

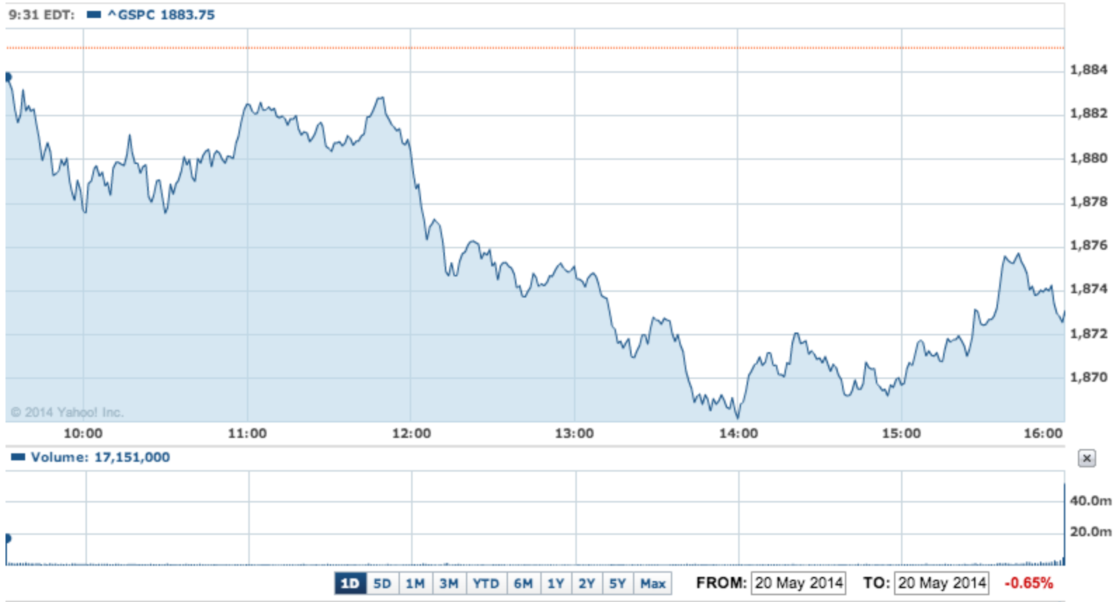


Figure 5.4: S&P 500 (1 day) price chart 20.05.2014 (Yahoo Finance, 2014b)

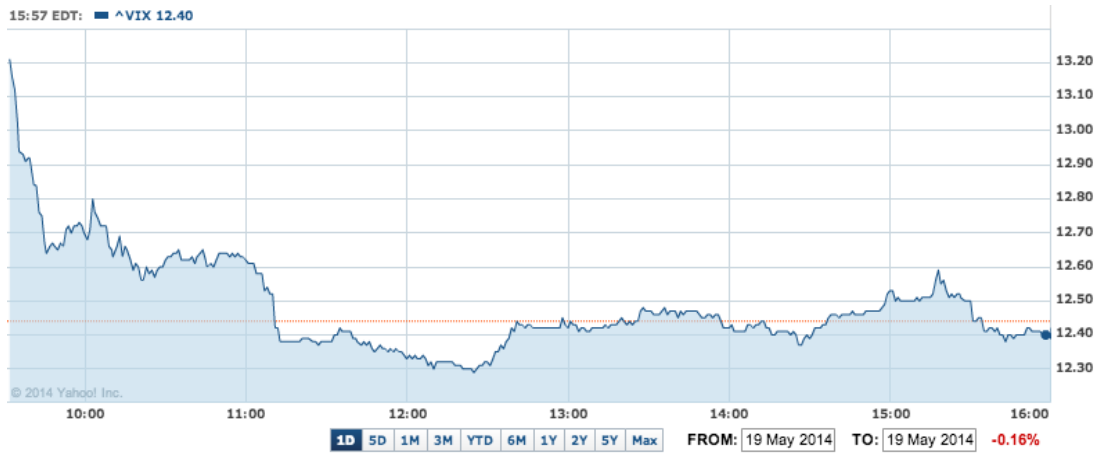


Figure 5.5: VIX (1 day) price chart 20.05.2014 (Yahoo Finance, 2014c)

European Indices

The indices selected (FTSE100, DAX, CAC40) are the key European indices. Based on my market experience, there is a positive correlation between the major global indices. That is, when one index is trading down, due to the market sentiment, others tend to follow.

VIX

In observing the VIX and the S&P 500, a clear correlation emerged. As a volatility index of S&P 500, the VIX has a negative correlation with the S&P 500 on a day-to-day basis. This correlation is clearly represented in Fig 5.5.

Currency Pairs

The fact the S&P 500 represents the 500 largest companies operating in the U.S. (based on market capitalisation), it is valid to assume that these component companies are multi-national in nature and hence effect, and are affected by events outside the U.S. often with significant parts of the revenue stream based in multi-currency. In the case of my algorithm and predicting the movement of the S&P 500, I felt it important to cater for global macro economic conditions. The architecture of my algorithm will also include the most relevant FMD inputs where necessary.

Swap Rates

Whilst the indices produce a virtual real-time view, the various swap rates are often an indicator of sentiment over a given time period. I have decided to include swap rates from 2 months to 30 years to cover the outlook of both short and long-term market conditions relative the key currencies. Through my experience in finance, swap rates represent a meaningful market indicator.

Commodities

The commodity markets represent a vital component to the prediction of the S&P 500. Today's global reliance on petrochemical derivatives has influenced the financial markets. The component companies of the S&P 500 are heavily influenced by the prices of petrochemical derivatives, which can account for a significant portion of their costs and in turn influence profits and growth potential.

As precious metals are often viewed as a safe haven commodity, they are relevant and tend to have an inverse correlation to positive market sentiment.

Chapter 6

Optimisation of the RPA Architecture and Future Work

In order to ensure the RPA can produce the highest possible directional trend accuracy, this chapter tests a number of trade offs that are possessed in the architecture of the RPA. These trade offs will be explored through extensive testing and a subsequent optimum for the RPA is then achieved.

6.1 Exploring factors that can increase the directional trend accuracy of the RPA

The nature of the RPA architecture ensures that the highest possible directional trend accuracy is achieved through the optimisation of key parameters. The optimum number of FMD inputs was determined in Chapter 5 and this chapter will explore two other key trade-offs identified, that I believe will directly influence the directional trend accuracy. These trade-offs must be tested to ensure that the RPA is computing the optimum parameters when processed through the GPU.

The results of the following two trade-offs will be tested and evaluated in this chapter:

1. Re-testing the relationship between the number of days historical data and directional trend accuracy in forecasting the S&P 500.

2. Testing to establish whether predicting tomorrow (***t+1***), and future dates (***t+2*** to ***t+4***) give a more accurate directional trend prediction result. This is established by maintaining the same RPA test architecture but 'time' being the variable, testing the RPA's performance in predicting the S&P 500 *beyond* tomorrow (***t+1***).

6.2 Re-testing the relationship between number of days historical data and directional trend accuracy

Section 6.2 will look to re-test the optimum number of days historical data to run each cycle of the RPA with the optimum number being the test result that produces the highest directional trend forecast of the S&P 500 at (***t+1***). Having already established the optimum number of FMD inputs to enter into the RPA in Chapter 5, I felt it important to build on these results and test for the optimum number of days historical data using the optimal 9 FMD inputs (see Fig 5.1) for each test cycle.

I chose to test 5 different sets of historical data (15, 30, 60, 75 and 90 days) in order to establish the optimum range. To test the trade-off between each set of historical data and the RPA's directional trend result, I applied the same methodologies used in Chapter 5, which focussed on attaining the optimum number of FMD inputs.

6.2.1 Results: testing a varying number of historical days data

Re-testing for the optimum number of historical days data produced a set of results in line with my initial tests carried out in Chapter 5. Table 6.1 indicates the '60-day' data as again being the most accurate range to use in the RPA. The directional trend results for '60-day' data set is at 55.1%, which is an increase of 2% on the same test carried out in Chapter 5 using 6 FMD inputs.

Table 6.1: Testing for optimum number of historical days data

Changing variable	Variable	Correct	Directional trend (%)
Historical Data	15 trading days	43/98	43.88
	30 trading days	51/98	52.04
	60 trading days	52/98	55.10
	90 trading days	34/68	50.00

Fig 6.1 produces a clear comparison of all four sets of data tested. The chart shows the 15-day range as producing an extremely low directional trend result of 44%, well below the >50% threshold that indicates a commercially viable result. A 30-day historical day data range displays a huge 8.16% increase in directional trend accuracy and moves to above the all important >50% threshold to 52.04% correct. Fig 6.1 clearly displays the 60-day data range as being the optimal data set to run the RPA. Using this data, the line chart peaks at 55.1% (52 correct out of the 98 trials) directional trend accuracy, a result that is significantly above the >50% threshold and would in turn give positive returns of 5.1% on an unleveraged position trading on the S&P 500 index. Having reached the optimum data range, the RPA's performance decreased to 50% when being computed using a 90 trading day historical data set.

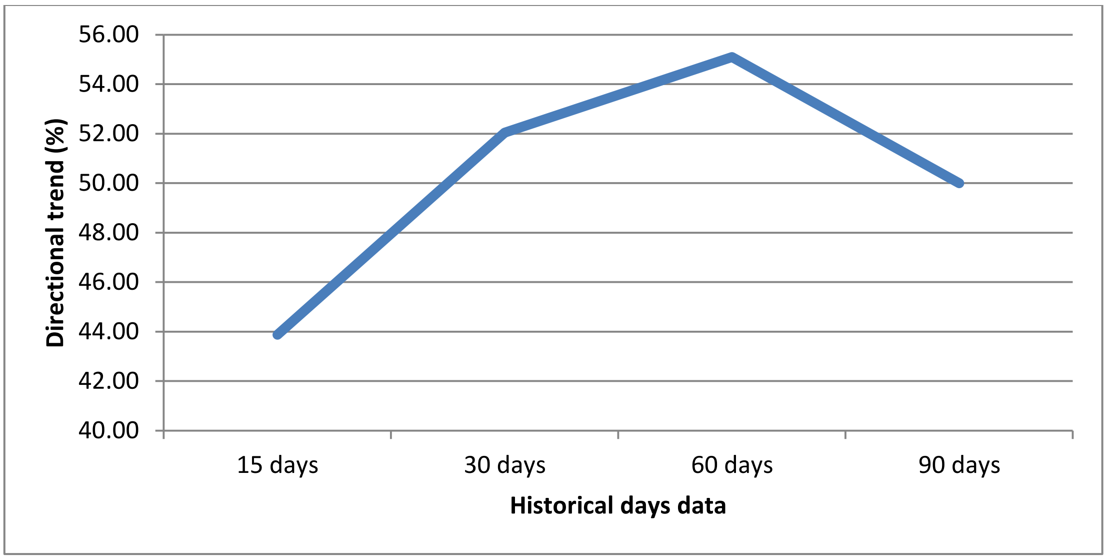


Figure 6.1: Historical days data vs. RPA directional trend accuracy

6.3 Establishing whether tomorrow (t+1) is the optimum time period in predicting the S&P 500

This section will seek to establish whether predicting tomorrow (***t****+****1***), and future dates (day after tomorrow (***t****+****2***), and up to four days in advance (***t****+****4***) give a more accurate directional trend prediction result. As the optimum number of FMD inputs and historical days data to use in computing the RPA, the final trade-off that can be explored through the architecture of my algorithm is understanding whether a lag exists between the movement in the data engine (combination of the 9 chosen FMD input prices over the prior 60 days) and the prediction time (***t****+****1*** to ***t****+****4***). My initial thought was that tomorrow (***t****+****1***) would be the optimum time frame due to the highly liquid and hence reactive nature of the S&P 500 index to micro and macro economic conditions, but I felt it was important to test all possible parameters. In Chapter 7 I will introduce the future possibility of future work applying this architecture to a timescale of less than tomorrow (***t****+****1***) that could allow the RPA to carry out multiple trades in a single one-day cycle.

6.3.1 Results: testing the responsiveness of the selected FMD data to the movement of the S&P 500

In the final series of RPA optimisation, I ran my algorithm (using the already established optimum parameters - FMD inputs and historical days data set) to test whether using this data engine to predict *tomorrow's* (***t+1***) price of the S&P 500 produced different results to predicting the price of the index, using the same data engine over a different time period. Table 6.2 displays the tests carried out with the moving variable being 'timescale of prediction'. As with previous testing, 98 trials were conducted.

Table 6.2: Testing responsiveness of the RPA

Changing variable	Variable	Correct	Directional trend (%)
Timescale of future prediction	***t+1***	54/98	55.10
	t+2	49/98	50.00
	t+3	47/98	47.96
	t+4	43/98	43.88

Fig 6.2 allows a clear comparison to be made between the four differing time scales. This chart shows a consistent negative correlation between the increasing time of predicting the S&P 500 and directional trend accuracy. The optimum time scale is tomorrow (***t+1***) which produces a directional trend accuracy of 55.1%. From day after tomorrow (***t+2)*** to four days in advance (***t+4)***, the directional trend accuracy decreases at an average of 3.74% as each day is added. These results re-affirm my initial hypothesis that a time lag between changing FMD and the movement of the S&P 500 does not exist.

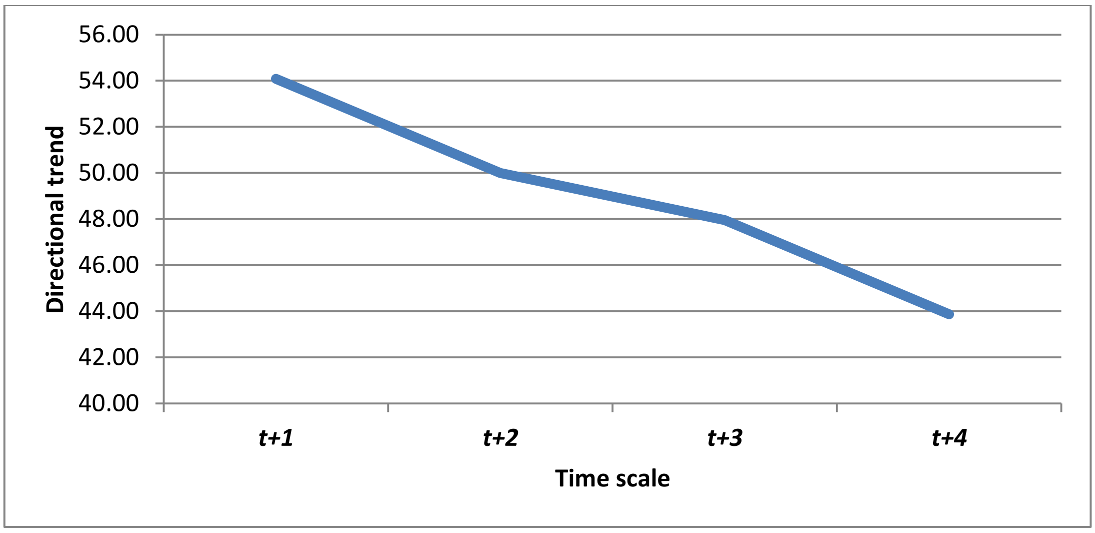


Figure 6.2: Time scale vs. RPA directional trend accuracy

6.4 Implications of test results in optimisation of RPA

The results in this series of testing have a similar pattern to testing for the optimum number of FMD inputs. This is due to the fact that increasing the number of historical days data significantly increases the amount of data to be computed through the RPA. As shown when testing the optimum number of FMD inputs in Chapter 5, a point was reached where any amount of additional data merely produced inaccurate 'noise'. As more data was input into the RPA, this led to a further reduction in directional trend accuracy. Both results, the Chapter 5 pre-testing, and the second phase of testing to establish the optimal number of historical days data, confirmed that 60 days is the optimum and should therefore be used as a constant in my RPA architecture. Through my experience, 60 days presents a range that is close enough to best capture the sentiment of the combination of the chosen 51 FMD inputs that I have demonstrated influence or are influenced by the S&P 500. I used 15 and 30 day increments when back-testing historical data; in Chapter 7, further work looks at whether there is a benefit in refining to smaller increments.

Furthermore, as I am using the difference between (***t***) and (***t-1***), logic dictates that the ideal period to forecast is (***t+1***) rather than larger data ranges. Further work in

Chapter 7 will look at the possibility of combining different forecasting time frames using different comparable time frames.

6.5 Future work

The architecture and methodologies used to build the RPA has provided a platform that successfully predicts the S&P 500 at (***t+1***). However, there are many aspects of the RPA that could be taken, adapted and used to develop it further for different applications. The mechanics of the RPA and its many different trade-offs can be altered to develop and optimise its capability in predicting the S&P 500. However, it is also important to understand and realise the scope in applying the structure of the RPA to predict other financial indices. The advances of the RPA provides a strong basis for the development of further work. This chapter is split into individual components of the RPA that I believe holds a significant scope to adapt and improve further.

6.5.1 Linking the RPA with a 'bio-inspired' system

On a functional level, in attaining a prediction result from the RPA, I am replicating the retina and applying it to financial prediction. In the RPA, I have successfully modelled the way in which the human retina works on a functional level, but not yet looked into the possibility of modelling the actual process of the eye. Currently a prediction is formed on an empirical basis, using MSE. I believe that a closer correlation with the human eye is possible, and with future research, I believe it would be possible to create an algorithm to replicate this process, which coupled with the architecture of the RPA, should produce an even more accurate prediction result.

So, my purpose would be to implement mathematical and computational modelling used to attain a better insight into the functionality of the retinal neuro-circuits to improve prediction of the S&P 500 index day-to-day movements.

I believe that the type of model to explore is a 'top down' approach based on the theory of nonlinear systems to look for correlations between the input and output of the system without going into physiological details about how the equivalent neuron or neural circuit achieves this.

In the simplest model of a retinal ganglion cell (Shwartz et al., 2006), the functional behaviour can be obtained by first identifying the receptive field vectors that the cell is sensitive to and then using these to characterise the functional form of the cells response. Fig 6.1 depicts the architecture of the so-called Linear-Nonlinear-Poissson (LNP) model of the human retina where $\boldsymbol{X}$ is high-dimensional visual input, $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_n}$ are the receptive field vectors (RFV) and $\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_n}$ are projections of the input vector $\boldsymbol{X}$ onto the RFV. These values are passed as an argument to nonlinear function $\boldsymbol{f}$, to create a generator signal $\boldsymbol{G}$. This signal is now the input for a spike generator $\boldsymbol{P}$, which outputs random spikes generated with Poisson statistics.

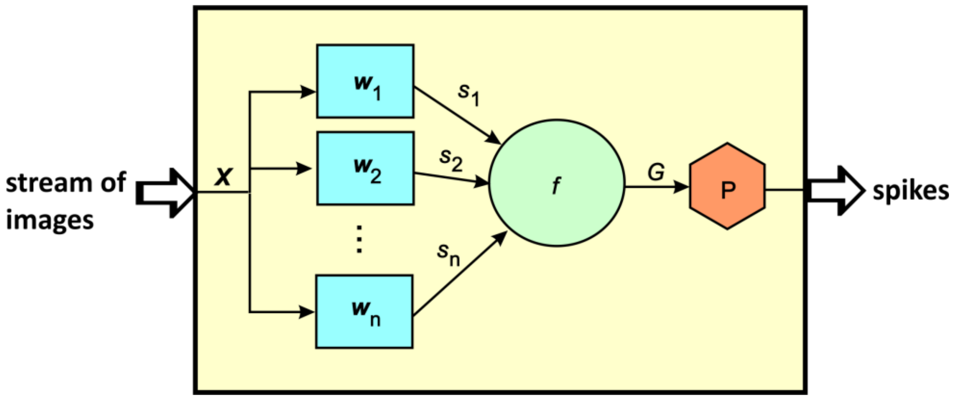


Figure 6.3: Block diagram of the LNP model (Shwartz et al., 2006)

Advancing this logic and applying it to financial prediction creates a very interesting opportunity that could be explored with a feature extraction of the inputs. Referring to Fig 6.3, the 'input data' $\boldsymbol{X}$ would be formed of many different closing prices of certain financial markets over a set time period. These financial data input vectors will convolve with vectors $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots,$ (receptive field vectors) and after passing through a non-linear function $\boldsymbol{f}$ an output result $\boldsymbol{G}$ is obtained. Understanding this process would require further research but the end output (***G*** in Fig 6.3) would be the rate of 'spikes' correlating to some form of reaction in the financial data (e.g. predicting the S&P 500/FTSE100/Currency pairs). The extent and pattern of these extracted 'spikes' could then provide the basis for a predictive decision for a chosen feature.

6.5.2 FMD Inputs

Minimising redundant FMD inputs

One possibility to minimise redundant FMD inputs within the RPA architecture and hence speed up the computational time, would be to take the bottom 10% of FMD inputs that contribute least to the movement of the S&P 500: this 10% will continue to adapt and change over time. It would then be interesting to adopt an 'improvement strategy' to the inputs that contribute least (highest MSE) to form an architecture that minimises error and maximises predictive efficiency.

Development of a 'brilliant index'

This section considers putting together a 'brilliant index' by grouping FMD inputs into relevant subsets (e.g. based on location - Japan) and establishing whether they become relevant at different times (E.g. if Japan had a crisis/credit default). If a specific subset became 'relevant', it would be programmed alongside the other chosen FMD inputs adding another element to aid the predictive strength and dynamic performance of the RPA.

Foveation

Another angle of the RPA that can be exploited is by looking at the FMD inputs alongside the notion of 'foveation of the eye' (hot spots) that are most relevant and most regularly included in the chosen FMD input set when computed by the RPA. The next step would be to establish whether it would be possible to direct computational power toward the highlighted 'hot focus areas' and less on the 'colder areas'. This could allow for a computational speed up as the GPU could direct and focus its computations on specific areas, only focusing on the ‘colder’ areas when they become more relevant. The benefits of this would be the elimination of redundant memory usage; the algorithm would only consider ‘relevant’ inputs at that particular point in time, providing both a faster and more significant calculation result.

6.5.3 GPU

I will continue to take advantage of the continually evolving GPU technologies. When forecasting the S&P 500 at (***t+1***), a set criterion is that I am able to forecast in the 24-hour period. Therefore I am limited, even with a fully optimised GPU, to a certain amount of data that can be used to help forecast the S&P 500 at (***t+1***) – this clearly limits the framework of my algorithm. However, in accordance with Moore’s Law, which states that the number of transistors in a dense integrated circuit doubles approximately every two years (Moore, 1965), this will give rise to further opportunity to test and utilise more data that should impact on forecast accuracy. A possible area of focus would be to increase the number of total FMD beyond 51 and to include specific economic data when focusing to forecast regional indices and possible use of subsets.

6.5.4 Figure of Merit (FoM)

It has been proved that the RPA can correctly forecast the directional trend of the S&P 500 55.1% of the time based on a once a day value of both the S&P 500 and 51 FMD. From a commercial point of view, it would be essential at this stage to risk the same bet of the same monetary value each day (because the accuracy is based on a ‘once a day’ value).

If, for example, a varied monetary amount is placed on each day, it is possible to maintain the directional trend accuracy of 55.1%, but still lose money. Therefore, ideally a strength index measurement would be perfect in allowing a varied bet size being feasible. I believe that adding a further process to the RPA which looks at the result for today (***t***) and then references that result to a previous result closely matching that of today's, can be used to form a strength index. Therefore the RPA could end up with a 55.1% directional trend but with an added 'strength index' based on how strong that movement for (***t+1***) might be. A strength index could be a multiplier index range, with a proposed 'strong' S&P 500 movement attributing a larger multiplier to the base level bet.

The addition of a further process to the RPA will create a significant amount of computational complexity, and hence increase in computational time, but would clearly refine the forecast and increase profit potential. Being able to vary bet sizes each day would lead to a greater profit potential – the architecture would be able to facilitate and exploit larger moves in a specific index.

6.5.5 Time frame of prediction result

Currently I am using the difference in movement between (***t***) and (***t-1***) when compared with a historical data set to form my prediction for the S&P 500 at (***t+1***). Results in Chapter 6 show that forecasting (***t+2/3/4***) is less accurate (See Fig 6.2). Further work could be carried out to establish whether the actual difference between (***t***) and (***t-2***) gives a more accurate forecast to predicting the S&P 500 at (***t+2***). Similarly it would be interesting to look at (***t+0.5***) and looking at whether there is any shorter term reaction of the S&P 500 to a change in the chosen FMD inputs when computing the RPA. Establishing a (***t+0.5***) prediction result would mean i) that the RPA would have to compute two prediction results each day, and ii) that two prices would be required for each of the 51 FMD inputs, perhaps a mid-session price and a close price.

6.5.6 Historical days data

In my research I only tested 15,30, 60 and 90 historical days data. In order to improve the efficiency and accuracy of the RPA the number of historical days data used to form the prediction result could be tested either side of the 60-day optimum.

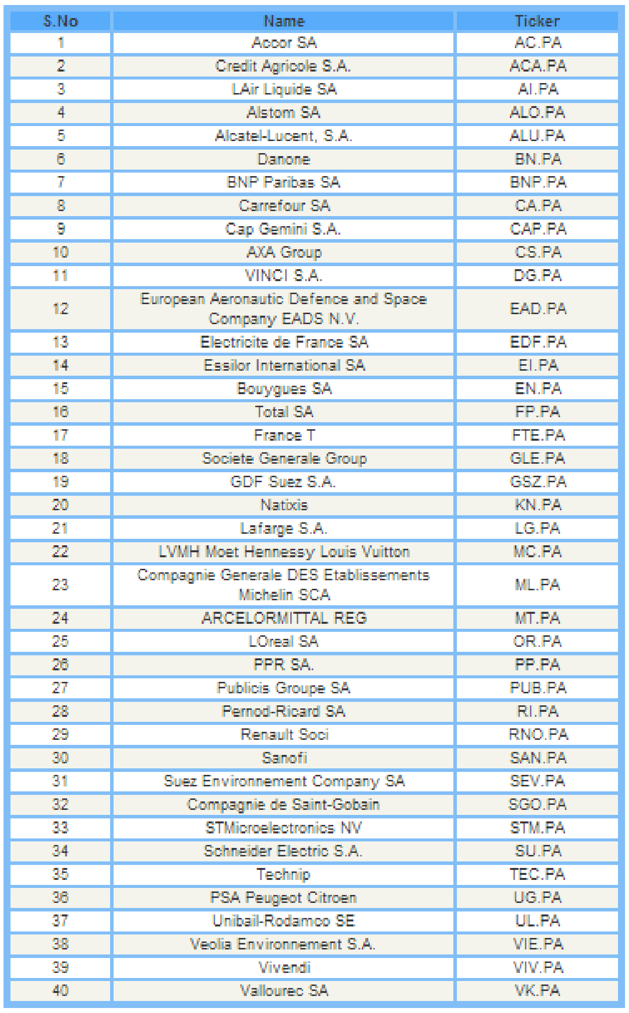
6.5.7 Establishing a reliable data source

The daily values of the 51 FMD inputs are available from a number of different sources, including Bloomberg, Reuters, Citi Bank, Goldman Sachs. One of the problems I have experienced is that the data can vary slightly from source to source, which could obviously lead to a difference in the prediction outcome when computed through the RPA. It is essential to check that the actual values used are accurate. My approach will be to continue to look at varied data sources using a mid-price approach. As the RPA is currently being used to predict the directional trend of the S&P 500 at (***t+1***), there are only 51 actual values to be used each day. These will be input manually into an electronic spreadsheet.

An area that I wish to explore as part of my future work is to link-in the input data relating to the FMD, directly into the GPU to remove human intervention. This will also be an important approach if looking to reduce the prediction timeframe to less than (***t+1***) and understand whether adopting multiple prediction per day strategy influences the directional trend forecast accuracy.

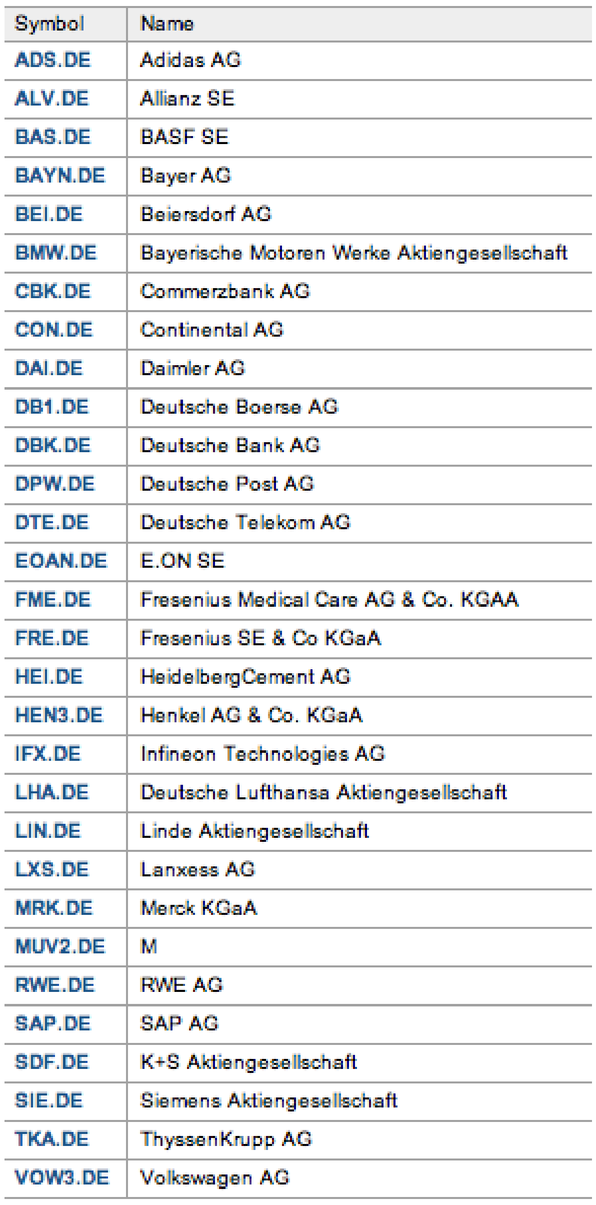
6.5.8 Predict other indices

As there is a variety of indices which combines groups of companies, currencies and commodities, an opportunity exists to see if using the RPA architecture enables a directional trend prediction to be made on these indices. Obviously the FMD inputs would be different for each individual index and further work would be required to understand which FMDs would be most relevant to predict the movement of these different indices. For example, work could be conducted to analyse indices such as the CAC40 (see Fig 6.3), over a benchmark French stock market index that represents a capitalisation-weighted measure of the 40 most significant values amongst the 100 highest market caps on the EuroNext Paris (see Fig 6.4), and the DAX30, a blue chip stock market index comprising the 30 major German companies trading on the Frankfurt Stock Exchange. Both these indices have a much smaller number of component companies compared with the larger S&P 500.



S.No	Name	Ticker
1	Accor SA	AC.PA
2	Credit Agricole S.A.	ACA.PA
3	LAir Liquide SA	AI.PA
4	Alstom SA	ALO.PA
5	Alcatel-Lucent, S.A.	ALU.PA
6	Danone	BN.PA
7	BNP Paribas SA	BNP.PA
8	Carrefour SA	CA.PA
9	Cap Gemini S.A.	CAP.PA
10	AXA Group	CS.PA
11	VINCI S.A.	DG.PA
12	European Aeronautic Defence and Space Company EADS N.V.	EAD.PA
13	Electricite de France SA	EDF.PA
14	Essilor International SA	EI.PA
15	Bouygues SA	EN.PA
16	Total SA	FP.PA
17	France T	FTE.PA
18	Societe Generale Group	GLE.PA
19	GDF Suez S.A.	GSZ.PA
20	Natixis	KN.PA
21	Lafarge S.A.	LG.PA
22	LVMH Moet Hennessy Louis Vuitton	MC.PA
23	Compagnie Generale DES Etablissements Michelin SCA	ML.PA
24	ARCELORMITTAL REG	MT.PA
25	LOreal SA	OR.PA
26	PPR SA.	PP.PA
27	Publicis Groupe SA	PUB.PA
28	Pernod-Ricard SA	RI.PA
29	Renault Soci	RNO.PA
30	Sanofi	SAN.PA
31	Suez Environnement Company SA	SEV.PA
32	Compagnie de Saint-Gobain	SGO.PA
33	STMicroelectronics NV	STM.PA
34	Schneider Electric S.A.	SU.PA
35	Technip	TEC.PA
36	PSA Peugeot Citroen	UG.PA
37	Unibail-Rodamco SE	UL.PA
38	Veolia Environnement S.A.	VIE.PA
39	Vivendi	VIV.PA
40	Vallourec SA	VK.PA

Figure 6.4: CAC40 component companies (Yahoo Finance, 2014d)



Symbol	Name
ADS.DE	Adidas AG
ALV.DE	Allianz SE
BAS.DE	BASF SE
BAYN.DE	Bayer AG
BEI.DE	Beiersdorf AG
BMW.DE	Bayerische Motoren Werke Aktiengesellschaft
CBK.DE	Commerzbank AG
CON.DE	Continental AG
DAI.DE	Daimler AG
DB1.DE	Deutsche Boerse AG
DBK.DE	Deutsche Bank AG
DPW.DE	Deutsche Post AG
DTE.DE	Deutsche Telekom AG
EOAN.DE	E.ON SE
FME.DE	Fresenius Medical Care AG & Co. KGAA
FRE.DE	Fresenius SE & Co KGaA
HEI.DE	HeidelbergCement AG
HEN3.DE	Henkel AG & Co. KGaA
IFX.DE	Infineon Technologies AG
LHA.DE	Deutsche Lufthansa Aktiengesellschaft
LIN.DE	Linde Aktiengesellschaft
LXS.DE	Lanxess AG
MRK.DE	Merck KGaA
MUV2.DE	M
RWE.DE	RWE AG
SAP.DE	SAP AG
SDF.DE	K+S Aktiengesellschaft
SIE.DE	Siemens Aktiengesellschaft
TKA.DE	ThyssenKrupp AG
VOW3.DE	Volkswagen AG

Figure 6.5: DAX30 component companies (Yahoo Finance, 2014e)

It would be interesting to see which inputs are affected, or influenced by these smaller indices, and whether by using these chosen FMD inputs, it is possible to predict their directional trend with the same success as achieved by the RPA when predicting the S&P 500. It would be important to understand how these indices are constructed and the way in which component companies are admitted or removed. The liquidity of the indices may be relevant when commercialising the RPA.

6.5.9 Predicting currency pairs

It would be interesting to apply the RPA to predict currency pairs. The foreign exchange (forex) market is the largest and most liquid of the financial markets (Oanda, 2014). Using the U.S. Dollar as an example, daily market activity often exceeds $4 trillion a day, with over $1.5 trillion of that conducted in the form of spot trading. The volatility of the forex market enables traders to take advantage of exchange rate fluctuations for speculative purposes (*ibid.*). Fig 6.5 displays the sheer size of the forex market in comparison to both the NASDAQ and the NYSE. For this reason, there is a real opportunity to utilise the benefits of trading currencies, combined with the reactive nature of the RPA, to produce a predictive architecture that operates under more volatile and highly liquid market conditions.

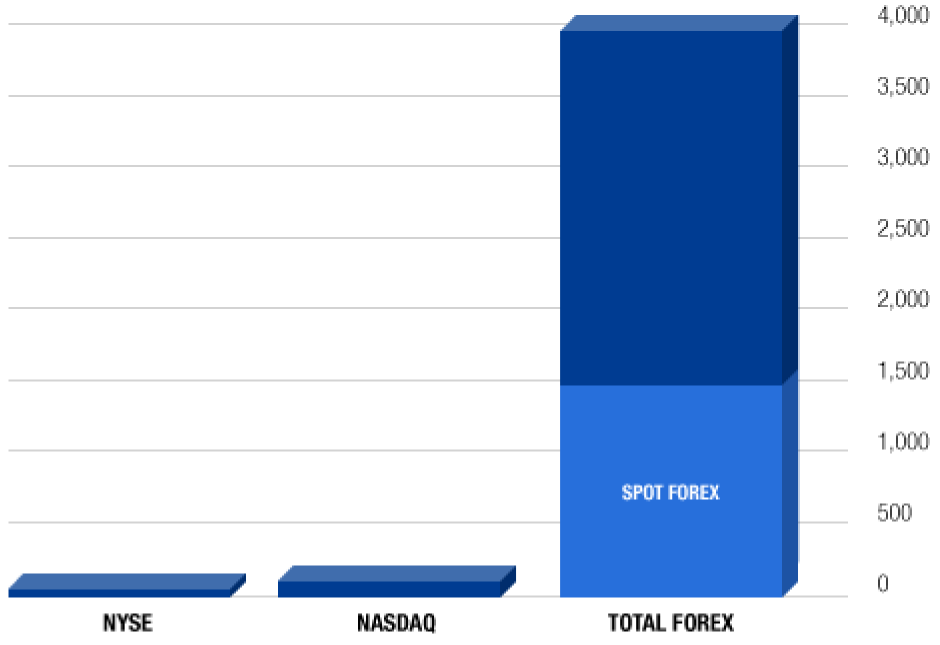


Figure 6.6: NASDAQ and NYSE dollar value in billions (USD)

Having outlined the key areas where I believe the architecture of the RPA can be exploited and developed for application in other areas in finance, the final chapter will draw my findings into concluding remarks.

Chapter 7

Conclusions

The primary aim of this thesis is the development of a new approach to forecasting financial markets using a novel computer algorithm, which I have referred to throughout as the RPA. The S&P 500 was used as the test index throughout the thesis. In this chapter I will reflect on the end points of the proceeding chapters, from which several indicators arise for consideration as future work.

The focus in Chapter 2 was to review previously published work on financial forecasting techniques. Current methods of forecasting the financial markets require highly intensive algorithms as the parameter inputs and computational complexity needed to create meaningful prediction are very large. Two of the most popular methods used are Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs), which are able to predict the stock market with some success. Different approaches have inherent advantages and disadvantages. An ANN involves a network of processing artificial neurons so that they exhibit complex global behaviors, determined by the connections between the processing elements and element parameters. The disadvantage of the ANN approach lies in scalability; whilst it is possible to implement in hardware, it is computationally inefficient and relatively computationally power hungry. Scalability lies in the ability of a system, network or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth (Bondi, 2000). Traditional ANNs using the back-propagation algorithm do not scale well as each neuron in one level is fully connected to each neuron in the previous level (Long & Gupta, 2008), hence slowing the process down considerably.

The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input, making the SVM a non-probabilistic linear classifier. SVMs are not readily scalable and are hence unsuitable as a solution to the task of predicting the S&P 500. The conclusions of my review of ANNs and SVMs are that these methods are not appropriate for further development to predict the directional trend of the S&P 500 and that a new approach is required.

In Chapter 3 my new bio-inspired algorithmic approach, the Regan Predictive Algorithm (RPA), is described, based on predicting tomorrow's value of the S&P 500 by looking closely at a wide range of financial factors, 51 in total. These factors were chosen as they influence the performance of the member companies within the index and link the predicted values back to the past values of the index. In the latter part of the chapter, the algorithm was successfully tested to prove the concept using MatLab, and up to 6 FMD inputs due to the severe computational limitations of its implementation. Having explored different hardware options, including FPGA, it became clear that a novel approach was required to facilitate the computational complexity of the RPA so that a result is obtained in less than the 24-hour time frame in order to predict (t+1) tomorrow's S&P 500 closing price.

Having assessed and considered various hardware options to meet the RPA's requirements, the latest graphics processing units (GPU) appeared to offer a practical solution and this development formed the basis of Chapter 4. The GPU was chosen because of its inherent capability of handling very large data sets and processing information in parallel extremely rapidly. This meant that the RPA was able to compute a far greater number of FMD inputs and hence significantly improve prediction performance and the scope for further development of the system. Initial results using a GPU computational engine confirmed that the GPU was ideally suited to facilitate the computationally intense RPA. The results from my initial proof of concept in MatLab were mirrored when applied to the GPU. Moreover, as anticipated, the GPU greatly reduced the computational time with a speed-up of 150x when running the RPA with 6 FMD inputs.

The focus of Chapter 5 was to establish at what point increasing one extra FMD input used to produce a buy/sell position for predicting the S&P 500 will lead to a detrimental effect in the directional trend forecasting. Also in this chapter, a further aim was to establish the optimum number of FMD inputs (out of the total 51) to be computed into the RPA, the optimum being the number of FMD inputs that produces the highest directional trend accuracy in predicting the S&P 500. I established that, even using the optimised GPU hardware, it would not be possible for the RPA to produce a prediction result using all 51 FMD inputs. This is because the number of combinations would extend the computational time to significantly over the 24-hour maximum. Interestingly, through extensive testing, this limitation did not prove to be important because I found that increasing the number of chosen FMD inputs above 9, would lead to severe financial 'noise' and actually start to reduce the directional trend accuracy of the prediction result. Financial noise refers to a theory of pricing developed by Fischer Black (1986). To Black, noise is the opposite of information in that sometimes it is hype and other times it is inaccurate ideas or data. Noise is everywhere in the economy and we can rarely tell the difference between it and information (*ibid.*)

Having identified 9 FMD inputs as the optimum, it is important to highlight the fact that these 9 chosen FMD inputs are dynamically selected by the RPA on a daily basis. This somewhat counterintuitive finding, I believe, is because the RPA automatically reacts to the changing market environment.

The final step in developing the RPA is reported in Chapter 6, which is focused on further improvements and optimisation with the objective of maximising the directional trend accuracy. In this chapter I tested the two remaining trade-offs which I believed would challenge the RPA's efficiency. These tests focused on establishing the optimum number of historical days data to be used when forming a prediction result and understanding whether predicting tomorrow (***t+1***) or future days (***t+2*** etc.) influences the RPA's directional trend accuracy. Having tested from 15 to 90 days historical data, I found that 60 days represented the optimum. One might have expected that using 90 days would give a better result than 60; this could well be due to the longer period extending across a greater variation in financial market stability, thus resulting in poorer directional trend accuracy than the 60-day time period.

In testing for the optimal prediction time frame, I found, as expected, that predicting for tomorrow (***t+1***) is the optimum. This confirms that it is necessary to run the RPA on a daily basis as the accuracy of predicting (***t+2, t+3*** etc.) is significantly worse, indicating that the movement of the chosen FMD inputs is correlated with a change in the price of the S&P 500. The fact that the algorithm relies on historical data does mean that it will not be able to predict any violent downward trends such as those encountered at the onset of a financial crash.

In August 2014 I had a paper published at the World Academy of Science Engineering and Technology (WASET) conference in Vancouver, Canada. The paper is titled "Novel GPU Approach In Predicting The Directional Trend Of The S&P 500" and focuses on the core findings of this research thesis [See Appendix 3]. Upon presenting my findings of this thesis in Vancouver, I was happy to be met with a series of interested and insightful questions on my work and subsequently, I have been invited to talk at a conference run by the same organisation in London later this year.

Going forward, the intention is to create an investment fund using the RPA system. I am passionate that when trading real time data to predict the S&P 500, there is to be virtually no human intervention and the fund will operate on purely a mathematical basis. Whilst I have proven to successfully predict the S&P 500 at more than 55%, it must be said that my RPA can in no way forecast sudden unexpected changes in market conditions, for example the 2007 economic crash. See Appendix 4 – a template for the RPA output page. This page includes the following information that is processed by the RPA market close: RPA chosen inputs, % of chosen inputs as proportion of total set, S&P close price value (t-1), S&P close price value (t), RPA S&P prediction value, Actual S&P % movement (t-1 to t).

Looking even further forward, I will continue to research and develop areas outlined in the section of this thesis on ‘future work’ and apply the RPA architecture to predict other areas within finance, namely the hugely liquid foreign exchange market. I will also look at analysing the actual process of the retina and applying these methodologies to financial prediction. I am extremely excited at the prospect of commercialising the RPA and further developing other areas that the RPA can be applied.

Finally, I have reached the end of the thesis and can confidently state that my original goal has been achieved. Returning to Lao Tzu’s quotation that was the opening gambit at the start of the thesis (Chapter 1 Introduction, p1) “Those who have knowledge don’t predict. Those who predict don’t have knowledge”. I can add a third sentence, “Those who have the right information and the resources to interpret it can predict with reasonable confidence.”

References

Bates, J, and Granger, C. (1969) “The combination of forecasts.” Operational Research Quarterly 20, pp 451-468.

Bergerson, K and Wunsch, D. (1991) “A commodity trading model based on a neural network-system hybrid” pp 289-293

Black, F. (1986). Noise. The Journal of Finance, 41(3), 529-543.

Bloomberg (2014a) Deutsche Boerse AG German Stock Index DAX. [online] Available at http://www.bloomberg.com/quote/DAX:IND [Accessed on 15.02.2014]

Bloomberg (2014b) Chicago Board Options Exchange SPX Volatility Index. [online] Available at http://www.bloomberg.com/quote/VIX:IND [Accessed on 17.02.15]

Blythe, D (2006). The Direct3D 10 system, ACM Trans. Graph., vol. 25, no. 3, pp. 724-734

Bondi, A (2000). Characteristics of scalability and their impact on performance, WOSP '00 Proceedings of the 2nd international workshop on Software and performance, 195-203

Breiman, L. Combining Predictors. In: Sharkey, A.J.C. (1990): Combining Artificial Neural Nets, Ensemble and Modular Multi-net Systems. Springer, Berlin, pp31-50

BullionVault (2014) Guide To Gold. [online] Available at https://www.bullionvault.com/guide/gold/Gold [Accessed on 15.02.14]

Campbell, J. Y., & Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average?. Review of Financial Studies, 21(4), 1509-1531.

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., & Skadron, K. (2008). A performance study of general-purpose applications on graphics processors using CUDA. Journal of parallel and distributed computing, 68(10), 1370-1380.

Chang, P., Wang, Y., and Yang, W. (2004) “An investigation of the hybrid forecasting models for stock price variation in Taiwan”. Vol. vol. 21, no. 4, pp. 358–368.

CME Group (2014a) Light Sweet Crude Oil (WTI) Futures and Options. [online] Available at http://www.cmegroup.com/trading/energy/light-sweet-crude-oil.html [Accessed on 15.02.14]

CME Group (2014b) Light Sweet Crude Oil (WTI) Futures and Options. [online] Available at http://www.cmegroup.com/trading/energy/natural-gas/natural-gas_contract_specifications.html [Accessed on 15.02.14]

Coates, A., Baumstarck, P., Le, Q., & Ng, A. Y. (2009). Scalable learning for object detection with GPU hardware. In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on (pp. 4287-4293). IEEE.

Cortes, C., & Vapnik, V. (1995) Support-vector networks. Machine learning, 20 (3), pp273-297.

Cowles, A. (1933). Can stock market forecasters forecast?. Econometrica: Journal of the Econometric Society, 309-324.

Crookes, D et al. (2009) "GPU implementation of the affine transform for 3D image registration." Machine Vision and Image Processing Conference, 2009. IMVIP'09. 13th International. IEEE.

Currencies Direct (2014) Forward contracts. [online] Available at http://www.currenciesdirect.com/business/forward-contracts/ [Accessed on 15.02.14]

EuroNext (2014) CAC 40. [online] Available at https://indices.euronext.com/products/indices/FR0003500008-XPAR?selectedMep=1 [Accessed on 15.02.14]

Evga.com. (2014) EVGA - Products - EVGA GeForce GTX TITAN - 06G-P4-2790-KR. [online] Available at: http://www.evga.com/Products/Product.aspx?pn=06G-P4-2790-KR [Accessed: 10 Feb 2014].

Fama, E. (1991) “Efficient capital markets: II” Journal of Finance 46, pp 1575-1617.

Fang, Z et al. (2013) "Fully CMOS-Compatible 1T1R Integration of Vertical Nanopillar GAA Transistor and Oxide-Based RRAM Cell for High-Density Nonvolatile Memory Application," Electron Devices, IEEE Transactions on , vol.60, no.3, pp.1108,1113

Fatahalian, K & Houston, M (2008). A closer look at GPUs. Communications of the ACM, 51(10).

Fernand-Rodriguez, F., Gonzdlez-Martel, C, Sosviall-Rivero, S. (2000) "On the Profitability of Technical Trading Rules based on Artificial Neural Networks: Evidence from the Madrid Stock Market". Economics Letter.

Fung, J., & Mann, S. (2004). Computer vision signal processing on graphics processing units. In Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on (Vol. 5, pp. V-93). IEEE.

Gershenson, C. (2003). Artificial neural networks for beginners.

Gilbert, E. & Karahalios, K. (2010). Widespread Worry and the Stock Market. In ICWSM pp. 59-65

Global Rates (2014) 6 month euro LIBOR interest rate. [online] Available at http://www.global-rates.com/interest-rates/libor/european-euro/eur-libor-interest-rate-6-months.aspx [Accessed on 15.02.14]

Goldberg, D., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. Urbana, 51, 61801-2996.

Goldman Sachs (2014) S&P GSCI COMMODITY INDEX. [online] Available at http://www.goldmansachs.com/what-we-do/securities/products-and-business-groups/products/gsci/ [Accessed on 17.02.14]

Goyal, A., and Welch, I (2003) "Predicting the equity premium with dividend ratios" Management Science 49, pp 639-654.

Goyal, A., and Welch, I (2008) "A comprehensive look at the empirical performance of equity premium performance" Review of Financial Studies 21) (2008) pp 1455-1508.

Granger C. (1990) "Modeling economic time series: Readings in econometric methodology", Oxford University Press: Oxford, UK

Gross, B. (2013) PIMCO; A unique fundamentals- based equity strategy. [online] Available at https://investments.pimco.com/MarketingPrograms/External%20Documents/PO4088_fundamental_advantage_absolute_return_strategy_fund_overview.pdf [Accessed on 12.02.14]

Gupta, S. and Wang, L. (2011) "Stock Forecasting with Feedforward Neural Networks and Gradual Data Sub-Sampling" School of Electrical and Electronic Engineering Nanyang Technological University

Hamid, S. (2004). Primer on using neural networks for forecasting market variables.

Hearst, M et al. (1998) Support vector machines. Intelligent Systems and their Applications, IEEE,13(4), pp.18-28.

Houck, C., Joines, J., & Kay, M. (1995). A genetic algorithm for function optimization: a Matlab implementation. NCSU-IE TR, 95(09).

Huang, Z., Chen, H., Hsu, C. J., Chen, W. H., & Wu, S (2004) "Credit rating analysis with support vector machines and neural networks: a market comparative study" Decision support systems, 37(4), 543-558.

Inoue, A., & Kilian, L. (2002). Bootstrapping autoregressive processes with possible unit roots. Econometrica, 70(1), 377-391.

Investopedia (2014) Long Bond. [online] Available at http://www.investopedia.com/terms/l/longbond.asp [Accessed on 17.02.15

Kaku, M. (1999). Visions: how science will revolutionize the 21st century. Oxford University Press.

Karolyi, G., & Stulz, R. (1996) "Why do markets move together? An investigation of US Japan stock return comovements" The Journal of Finance, 51(3), 951-986.

Keckler, S. W et al. (2011). GPUs and the future of parallel computing. IEEE Micro, 31(5), 7-17.

Khan, A., Bandopadhyaya, T. and Sharma, S. (2008) "Genetic algorithm based backpropagation neural network performs better than backpropagation neural network in stock rates prediction" IJCSNS International Journal of Computer Science and Network Security, Vol 8, No.7, July 2008, pp162-166

Kim, K-J and Han, I. (2000) "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index." Expert systems with applications 19.2, pp 125-132.

Klamt, S., Saez-Rodriguez, J., & Gilles, E. D. (2007) Structural and functional analysis of cellular networks with CellNetAnalyzer. BMC systems biology, 1(1), 2.

Lai, K-K et al. (2006) "A novel nonlinear neural network ensemble model for financial time series forecasting." Computational Science, ICCS 2006. Springer Berlin Heidelberg, pp790-793

Lapedes, A. and Farber, R. (1987) "Nonlinear Signal Processing Using Neural Networks," paper presented to the IEEE Conference on Neural Information Processing System-Natural and Synthetic

Lee, V., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A., ... & Dubey, P. (2010). Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. In ACM SIGARCH Computer Architecture News (Vol. 38, No. 3, pp. 451-460). ACM.

Liu, H, & Motoda, H. (1998) Feature selection for knowledge discover and data mining, Norwell, MA: Kluwer Academic

Liu, Y. (2014). GPU-accelerated Computation for Statistical Analysis of the Next-Generation Sequencing Data (Doctoral dissertation, Worcester Polytechnic Institute

London Stock Exchange (2014) FTSE 100. [online] Available at http://www.londonstockexchange.com/home/homepage.htm [Accessed on 15.02.14]

Long, L and Gupta, A (2008). Scalable Massively Parallel Artificial Neural Networks, JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION, Vol. 5, January 2008, 3-15

Malliaris, M. (1994) "Modelling The Behaviour of the S&P 500 index: a Neural Network Approach." Artificial Intelligence for Applications, Proceedings of the Tenth Conference on. IEEE, pp 86-90

MathWorks (2014) MATLAB: The Language of Technical Computing. [online] Available at http://www.mathworks.co.uk/products/matlab/ [Accessed on 06.03.14]

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.Meyer, D. (2012) Support Vector Machines. The Interface to libsvm in package e1071. e1071 Vignette

Moore, G. E. (1965). Cramming more components onto integrated circuits. Chica

NASA (2014) High End Computing Capability: Computing Power to Answer NASA Complex Science and Engineering Questions. [online] Available at http://www.nas.nasa.gov /hecc/support/kb/glossary/?let=G [Accessed on 16.04.14]

NASDAQ (2014) NASDAQ-100 Index and QQQ. [online] Available at http://www.nasdaq.com/about/financial_products.pdf [Accessed on 15.02.2014]

Neely, C et al. (2003). Forecasting the equity risk premium: the role of technical indicators. Federal Reserve Bank of St. Louis Working Paper

Nickolls, J. (2007) Hot Chips 2007: GPU Parallel Computing Architecture and CUDA Programming Model, NVIDA Corporation

Nikolic, K., Grossman, N., Yan, H., Drakakis, E., Toumazou, C., & Degenaar, P. (2007, August). A non-invasive retinal prosthesis-testing the concept. In Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pp. 6364-6367. IEEE.

NVIDA (2014) CUDA Parallel Computing: GPU Computing on CUDA Architecture. [online] Available at http://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html [Accessed on 10.04.14]

Oanda (2014) Intro to currency trading [online] Available at http://fxtrade.oanda.co.uk/learn/intro-to-currency-trading/benefits/trading [Accessed 23.06.14]

Owens, J., Houston, M., Luebke, D., Green, S., Stone, J., & Phillips, J. (2008). GPU computing. Proceedings of the IEEE, 96(5), 879-899. Chicago

Owens, J. et al. (2008) GPU computing. Proceedings of the IEEE 96.5: 879-899

PIMCO (2014) What Are Interest Rate Swaps and How Do They Work?. [online] Available at http://www.pimco.co.uk/EN/Education/Pages/InterestRateSwapsBasics1-08.aspx [Accessed on 15.02.14]

QNX (2014) Writing a graphics driver. [online] Available at http://www.qnx.com/developers/docs/6.4.1/ddk_en/graphics/writing.html [Accessed on 23.03.14]

Rumelhart, D et al. (1986) Learning Internal Represations by Error Promogation, in Parallel Distributed Processing, Vol 1, Chapter 8, MIT Press, Cambridge

S&P Dow Jones Indices (2014) Index Mathematics Methodology

S&P Dow Jones Indices (2014) S&P 500

S&P Indices (2012) S&P US. Indices Methodology, pp3-23

Saxenian, A. (1991). The origins and dynamics of production networks in Silicon Valley. Research policy, 20(5), 423-437. Chicago

Scholkopf, B., & Smola, A. (2002) Learning with kernels. MIT Press, pp 366-369

Sexton, R et al. (1998) Toward global optimization of neural networks: a comparison of the genetic algorithm and backpropagation. Decision Support Systems, 22 (2) pp 171-185.

Shockley, W. (1949) The Theory of Junctions in Semiconductors and Junction Transistors. Bell System Technical Journal, 28(3), 435-489.

Schwartz, O et al. (2006) Spike-triggered neural characterization Journal of Vision, 6, pp484–507

Simon, H. (1982) Models of Bounded Rationality (2 vols). Cambridge: MIT Press

STOXX (2014) EuroStoxx 50. [online] Available at http://www.stoxx.com/indices/index_information.html?symbol=sx5e [Accessed on 15.02.14]

Swap-Rates (2014) US swap (fixed) rates. [online] Available at http://www.swap-rates.com/USSwap.html [Accessed on 15.02.14]

Tay, F, and Cao, L. (2001) "Application of support vector machines in financial time series forecasting." Omega 29.4, pp 309-317.

Thaler, R. (1985). Does the stock market overreact?. The Journal of finance, 40(3), 793-805.

Thiesen, M. (2010) Video Graphics and Genomics: A Real Game Changer?. [online] Available at http://blog.goldenhelix.com/?p=374 [Accessed on 23.04.14]

Timmermann, A. (2006) Forecast combinations. In Elliot, G, et al, Handbook of Economic Forecasting, vol. 1. Amsterdam: Elsevier

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. J. of Math, 58, 345-363.

White, H. (1988) "A Performance Comparison for some On-Line and Off-Line Learning Methods for Single Hidden Layer Feed forward Kets" UCSD Department of Economics Discussion Paper

White, H. (1988) "Economic prediction using neural networks: The case of IBM daily stock returns." Neural Networks, IEEE International Conference on. IEEE

Yahoo Finance (2014a) "NASDAQ100 (1 day) price chart 20.05.2014" [online] Available at https://uk.finance.yahoo.com/q?s=%5ENDX [Accessed on 20.05.14]

Yahoo Finance (2014b) "S&P 500 (1 day) price chart 20.05.2014" [online] Available at https://uk.finance.yahoo.com/q?s=%5EGSPC [Accessed on 20.05.14]

Yahoo Finance (2014c) "VIX (1 day) price chart 20.05.2014" [online] Available at https://uk.finance.yahoo.com/q?s=%5EVIX [Accessed on 20.05.14]

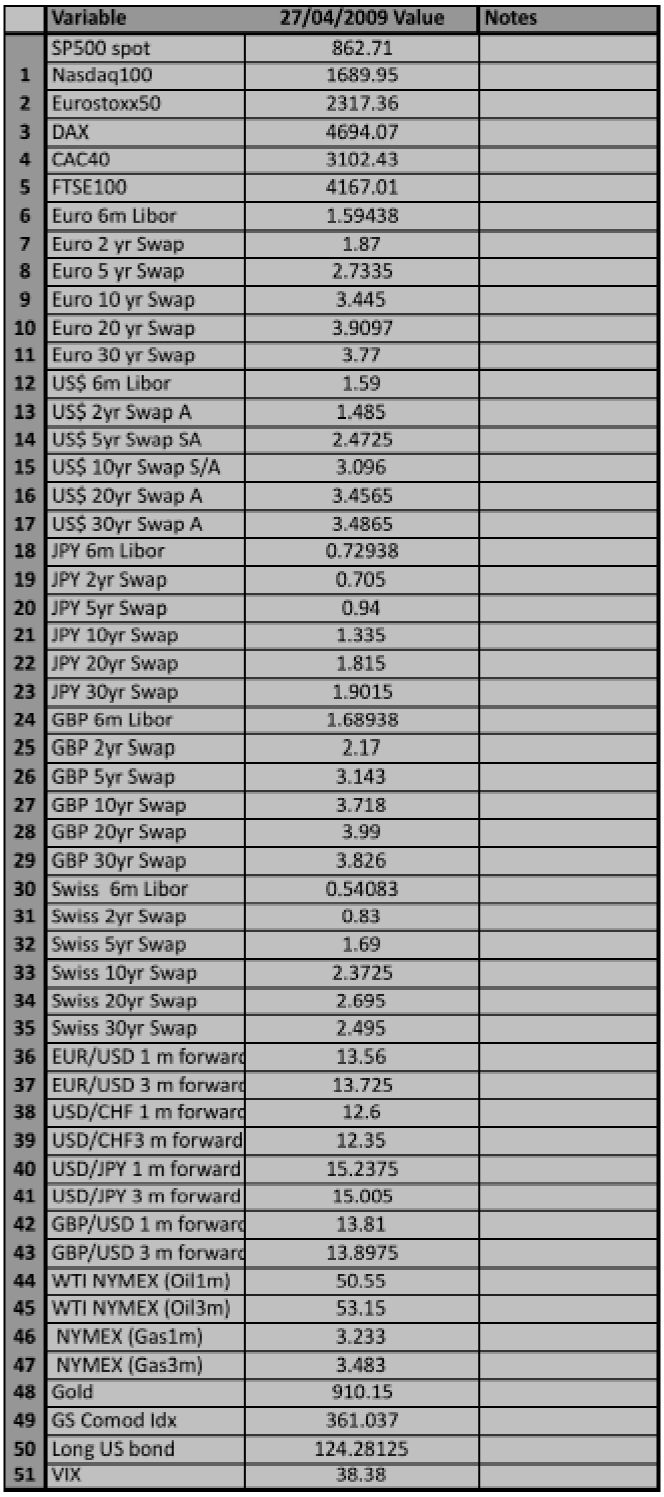
Yahoo Finance (2014d) "CAC 40 component companies" [online] Available at https://finance.yahoo.com/q/cp?s=%5EFCHI+Components [Accessed on 22.05.14]

Yahoo Finance (2014e) "DAX 30 component companies" [online] Available at https://finance.yahoo.com/q/cp?s=%5EGDAXI+Components [Accessed on 22.05.14]

Yuan, C. (2011) Predicting S&P 500 Returns Using Support Vector Machines: Theory and Empirics. Centre for Research in Economics and Strategies Fellowship

Appendices

Appendix 1: 51 chosen FMD inputs with their individual daily values



	Variable	27/04/2009 Value	Notes
	SP500 spot	862.71	
1	Nasdaq100	1689.95	
2	Eurostoxx50	2317.36	
3	DAX	4694.07	
4	CAC40	3102.43	
5	FTSE100	4167.01	
6	Euro 6m Libor	1.59438	
7	Euro 2 yr Swap	1.87	
8	Euro 5 yr Swap	2.7335	
9	Euro 10 yr Swap	3.445	
10	Euro 20 yr Swap	3.9097	
11	Euro 30 yr Swap	3.77	
12	US$ 6m Libor	1.59	
13	US$ 2yr Swap A	1.485	
14	US$ 5yr Swap SA	2.4725	
15	US$ 10yr Swap S/A	3.096	
16	US$ 20yr Swap A	3.4565	
17	US$ 30yr Swap A	3.4865	
18	JPY 6m Libor	0.72938	
19	JPY 2yr Swap	0.705	
20	JPY 5yr Swap	0.94	
21	JPY 10yr Swap	1.335	
22	JPY 20yr Swap	1.815	
23	JPY 30yr Swap	1.9015	
24	GBP 6m Libor	1.68938	
25	GBP 2yr Swap	2.17	
26	GBP 5yr Swap	3.143	
27	GBP 10yr Swap	3.718	
28	GBP 20yr Swap	3.99	
29	GBP 30yr Swap	3.826	
30	Swiss 6m Libor	0.54083	
31	Swiss 2yr Swap	0.83	
32	Swiss 5yr Swap	1.69	
33	Swiss 10yr Swap	2.3725	
34	Swiss 20yr Swap	2.695	
35	Swiss 30yr Swap	2.495	
36	EUR/USD 1 m forwar	13.56	
37	EUR/USD 3 m forwar	13.725	
38	USD/CHF 1 m forwar	12.6	
39	USD/CHF3 m forward	12.35	
40	USD/JPY 1 m forward	15.2375	
41	USD/JPY 3 m forward	15.005	
42	GBP/USD 1 m forwar	13.81	
43	GBP/USD 3 m forwar	13.8975	
44	WTI NYMEX (Oil1m)	50.55	
45	WTI NYMEX (Oil3m)	53.15	
46	NYMEX (Gas1m)	3.233	
47	NYMEX (Gas3m)	3.483	
48	Gold	910.15	
49	GS Comod Idx	361.037	
50	Long US bond	124.28125	
51	VIX	38.38	

Appendix 2: GPU testing results (6 FMD inputs/30 days historical data/predicting tomorrows directional trend of the S&P 500)

Chosen 1	Chosen 2	Chosen 3	Chosen 4	Chosen 5	Chosen 6	Coeff 0	Coeff 1	Coeff 2	Coeff 3	Coeff 4	Coeff 5	Coeff 6	Yesterday	Actual	Predicted	Correct	Error
0	7	10	24	33	37	174.9566	0.405799	-227.078	193.3567	149.7693	-181.952	10.05389	857.39	887.68	858.29	true	55344.76
0	8	10	24	29	37	375.0888	0.403997	-376.104	241.7645	114.3335	-84.9509	14.57439	887.68	896.24	893.22	true	54546.74
0	8	10	24	29	37	376.1832	0.404277	-375.511	240.9287	114.4685	-85.4788	14.59072	896.24	816.21	883.6	true	54552.92
0	7	10	24	32	37	116.3449	0.395306	-251.978	196.3215	155.0521	-160.602	12.12445	816.21	848.81	804.47	false	56526.06
0	8	10	24	29	37	325.3526	0.377678	-386.674	265.2222	118.4881	-83.3611	15.57265	848.81	870.74	845.75	false	55114.66
0	8	10	24	29	37	371.8874	0.38944	-407.758	268.0019	120.4865	-82.7219	15.82509	870.74	845.22	852.17	true	54958.42
0	8	10	24	29	37	349.0467	0.386922	-397.522	265.6543	118.8716	-83.3932	15.86488	845.22	876.07	844.43	false	54842.9
0	8	10	24	29	37	357.1331	0.378868	-388.007	257.5784	120.8512	-86.1574	15.70117	876.07	909.7	818.73	false	54399.01
0	5	24	27	28	37	709.3607	0.316569	-186.232	189.6986	-396.922	323.4165	12.71211	909.7	888.67	903.48	true	57246.63
0	5	24	27	28	37	711.0541	0.309506	-187.418	192.8305	-406.999	333.2163	13.09832	888.67	899.24	895.45	true	57227.57
0	5	24	27	28	37	742.4427	0.288245	-197.859	203.9208	-405.192	331.0682	13.53389	899.24	873.59	885.59	true	54198.93
0	5	24	27	28	37	732.5406	0.291251	-195.454	202.1301	-409.429	335.3894	13.64367	873.59	879.73	887.31	true	54273.25
0	5	24	27	28	37	707.4339	0.295948	-188.534	194.354	-396.819	325.2039	13.76692	879.73	868.57	921.55	false	51887.93
0	5	15	24	25	37	991.3112	0.219818	-254.352	87.46437	410.1666	-332.146	10.82977	868.57	913.18	923.49	true	51374.62
0	5	15	24	25	37	1017.333	0.209618	-254.451	90.41646	419.7301	-345.116	10.65938	913.18	904.42	961.84	false	51180.88
0	5	15	24	25	37	1042.098	0.196692	-246.106	95.37113	424.15	-357.8	9.256463	904.42	885.28	947.44	false	53649.72
0	5	16	24	33	37	679.8039	0.216945	-176.496	81.04516	196.0208	-164.012	7.963385	885.28	887.88	928.34	true	54900.69
0	20	24	29	35	50	-448.732	0.513371	128.8365	78.36035	-103.056	5.016551	2.806785	887.88	871.63	902.21	false	54745.87
0	20	24	30	35	50	-584.465	0.624357	145.608	70.21762	-147.307	4.99906	3.288126	871.63	863.16	857.17	true	53155.05
0	20	24	30	35	50	-701.024	0.691514	157.606	67.03557	-156.105	4.777779	3.724107	863.16	869.42	833.11	false	48023.65
0	20	24	30	31	40	-468.869	0.394086	248.8092	112.8649	-263.524	116.282	9.935666	869.42	890.64	843.6	false	44665.82
0	20	24	30	31	40	-398.273	0.373018	218.3701	107.3114	-271.815	141.8118	9.04584	890.64	931.8	862.38	false	45714.05
0	3	4	19	37	50	-771.699	0.546129	-0.40944	0.376708	223.3187	6.253488	3.207764	931.8	927.45	952.57	false	47408.48
0	11	21	30	31	35	-186.796	0.300839	77.58392	126.1518	-224.047	177.0818	7.169314	927.45	934.7	939.77	true	44833.44
0	5	30	31	40	48	-410.98	0.361884	185.3347	-513.143	433.4571	4.92163	-0.52212	934.7	906.65	917.25	true	41522.79
0	5	30	31	40	48	-409.695	0.362255	188.3902	-513.945	433.3563	4.840546	-0.5507	906.65	909.73	925.11	true	41595.38
0	5	30	31	40	48	-377.032	0.363898	186.2498	-498.111	424.5762	4.824035	-0.63871	909.73	890.35	918.92	false	40137.92
0	5	30	31	40	48	-440.622	0.377769	194.1801	-509.495	417.6544	5.23047	-0.54243	890.35	870.26	864.63	true	38817.05
0	5	30	31	40	48	-450.418	0.365052	194.0335	-518.53	423.1157	5.237797	-0.45601	870.26	871.79	855.63	false	38392.61
0	13	23	30	31	48	200.8965	0.309389	84.01376	53.39249	-301.327	218.1099	-0.63361	871.79	842.62	871.31	true	37016.69
0	13	23	30	31	48	226.5789	0.301479	84.03696	54.26023	-298.733	220.6403	-0.70596	842.62	843.74	842.6	false	37308.56
0	13	17	30	31	35	-239.813	0.258553	90.89164	338.8694	-248.747	192.4151	4.700333	843.74	850.12	833.79	false	33759.02
0	13	23	30	31	48	179.7917	0.334196	85.67708	53.75867	-291.596	201.3394	-0.64105	850.12	805.22	860.36	false	33842.24
0	13	17	30	31	35	-298.224	0.25144	99.34077	378.0062	-273.224	213.9411	4.645552	805.22	840.24	819.65	true	34033.13
0	3	4	26	27	35	236.5058	0.247463	-0.25897	0.224297	258.0522	-251.733	5.694381	840.24	827.5	826.53	true	33229.74
0	25	27	29	35	38	450.2321	0.2954	119.204	-125.658	-52.2105	16.40434	-13.3563	827.5	831.95	793.51	false	31333.86
2	7	25	35	44	48	704.2542	0.091119	-94.2095	109.3889	8.248094	17.2785	-3.83862	831.95	836.57	837.44	true	31075.35
2	27	28	35	44	48	372.1387	0.100068	255.6544	-182.154	9.369928	15.80966	-3.66316	836.57	845.71	852.21	true	30557.16
2	15	23	29	40	48	-50.9379	0.145559	65.59716	65.89044	-118.632	11.54269	-0.7028	845.71	874.09	867.18	true	30387.11
0	4	25	35	44	48	555.1336	0.159127	0.041981	60.24784	8.373215	16.34528	-3.70579	874.09	845.14	881.44	false	26869.21
0	25	28	35	44	48	651.3258	0.226965	57.22445	-32.7413	7.378963	13.69561	-2.96679	845.14	825.88	862.82	false	27664.58
0	25	28	35	44	48	612.3293	0.238166	63.0995	-36.2035	7.497245	12.20854	-2.72383	825.88	825.44	846.05	false	28149.25
0	25	27	35	44	48	638.5672	0.239739	69.61196	-44.5449	7.277479	12.17877	-2.72975	825.44	838.51	844.91	true	28220.77
0	17	35	36	44	48	478.1192	0.247583	281.9648	21.50247	-17.68	13.46643	-2.92671	838.51	832.23	827.66	true	27985.62
0	17	35	36	44	48	486.1652	0.242772	284.5168	22.21756	-17.9215	14.66957	-3.13579	832.23	845.85	830.54	false	27581.3
2	3	13	14	35	38	422.8235	0.549903	-0.65862	156.3215	-157.505	21.00227	-21.0297	845.85	868.6	852.39	true	25154.42
2	3	13	14	35	38	431.4258	0.55091	-0.66276	153.3187	-153.278	20.25167	-20.3992	868.6	869.89	889.17	true	25323.4
2	3	13	14	35	38	406.1748	0.555714	-0.65957	157.2033	-161.255	22.59023	-22.8179	869.89	827.16	888.2	false	24514.35

2	3	5	29	35	38	362.9554	0.442316	-0.53821	65.61565	-93.0051	20.24575	-20.5083	827.16	833.74	868.98	true	24473.57
1	2	5	18	23	35	175.3529	-0.77899	0.454995	111.7616	442.3306	-137.707	8.069269	833.74	835.19	863.95	true	23141.93
14	18	23	25	27	35	655.6693	55.73843	416.2215	-123.602	183.9008	-195.671	11.65491	835.19	826.84	813.55	true	21616.73
14	18	23	25	27	35	665.917	58.04478	402.2381	-122.61	183.8146	-197.132	11.54807	826.84	789.17	807.03	true	21525.18
11	23	25	28	41	45	689.9644	64.4122	-191.254	274.3183	-209.993	12.92866	36.14612	789.17	788.42	829.37	false	17811.69
18	23	25	28	41	45	568.9402	252.5385	-170.83	249.7933	-185.114	11.04513	39.25282	788.42	778.94	823.4	false	18037.37
6	11	23	30	42	46	127.1642	229.4491	78.86184	-189.252	-175.698	15.20765	74.23225	778.94	770.05	796.76	false	19261.42
18	22	27	30	32	45	644.0446	796.5523	-411.846	-153.947	-220.007	363.5674	40.78563	770.05	743.33	758.05	true	17346.38
18	22	27	30	32	45	637.0228	850.6473	-427.106	-156.408	-250.366	366.2227	45.59926	743.33	773.14	744.01	true	16683.85
18	22	27	29	32	45	668.166	545.567	-352.26	-115.341	-85.9981	269.928	40.48884	773.14	764.9	770.49	true	15339.75
16	19	21	30	31	45	505.5631	-46.281	656.7473	-529.304	-318.645	430.5927	53.8563	764.9	752.83	742.3	true	14293.08
16	19	21	30	31	45	494.8094	-45.1657	642.0548	-512.312	-313.96	428.7106	52.33351	752.83	735.09	763.41	false	14365.36
6	19	22	23	35	45	493.7854	160.9913	273.6419	-198.126	-144.642	4.495789	65.07069	735.09	700.82	747.99	false	14205.52
16	19	21	30	31	45	570.3142	-55.6956	723.1203	-597.242	-348.337	438.1816	59.27158	700.82	696.33	736.53	false	15272.73
2	3	15	39	44	48	695.6092	0.33505	-0.32695	-66.0341	6.046424	12.34743	-2.53255	696.33	712.87	732.22	true	15149.24
4	8	22	28	45	46	594.6733	0.067368	149.4556	-151.104	-102.636	233.1923	-219.896	712.87	682.55	700.66	true	13350.15
4	8	22	28	45	46	562.2465	0.064271	150.3691	-156.936	-93.5193	252.3122	-234.979	682.55	683.38	682.33	false	13251.09
4	8	22	28	45	46	556.2634	0.061781	148.7836	-159.574	-89.4449	255.1503	-235.399	683.38	676.53	679.71	true	13184.01
4	8	22	28	45	46	556.7219	0.061796	142.8604	-162.083	-85.8488	257.9181	-235.667	676.53	719.6	684.62	true	13122.87
8	20	21	27	45	46	836.8608	153.7425	389.4429	-466.492	-109.132	253.1492	-219.56	719.6	721.36	709.42	false	13421.57
4	8	22	27	45	46	510.5361	0.070103	160.0771	-138.61	-95.0507	218.7615	-203.894	721.36	750.74	717.79	false	13049.34
2	10	11	17	28	42	321.2962	0.099113	153.2712	73.03608	-301.026	-193.409	17.31438	750.74	756.55	747.54	false	13465.95
2	10	11	17	28	42	348.2354	0.097148	158.2758	69.45471	-306.785	-198.638	17.01722	756.55	753.89	765.65	false	13249.15
2	10	11	17	28	42	317.5011	0.099568	152.6063	69.03323	-291.078	-190.072	16.8423	753.89	778.12	764.93	true	13274.98
2	10	11	17	28	42	318.7996	0.099201	156.7454	71.09487	-296.726	-192.222	16.57713	778.12	794.35	768.62	false	13427.22
4	9	18	21	27	45	255.6244	0.093293	160.4121	388.7048	-216.3	-128.874	25.59092	794.35	784.04	719.46	true	13370.96
4	18	21	29	45	49	-525.18	0.135892	690.4014	-153.577	-323.723	40.31845	4.299092	784.04	768.54	788.35	false	13613.02
4	8	27	29	42	49	-235.512	0.108418	173.3864	-120.965	-364.778	11.27044	3.688933	768.54	822.92	793.28	true	12876.89
4	8	28	29	42	49	-164.882	0.098659	176.1748	-153.98	-514.317	12.80452	4.640113	822.92	806.12	807.39	true	12630.57
4	8	28	29	42	49	-160.244	0.099003	175.8306	-154.184	-515.136	12.81056	4.611725	806.12	813.88	808.48	true	12621.66
4	8	28	29	42	49	-235.903	0.095444	183.8283	-147.109	-506.625	12.31649	4.919128	813.88	832.86	787.05	false	12417.23
2	8	27	29	42	49	-396.786	0.088962	172.2257	-96.097	-449.034	12.95114	4.852757	832.86	815.94	810.19	true	13383.7
1	5	8	27	42	49	-660.896	0.214736	-131.167	192.024	-128.232	13.01583	6.554827	815.94	787.53	808.08	true	12977.95
1	5	8	27	42	49	-638.409	0.209975	-127.244	191.4211	-129.433	13.24199	6.41217	787.53	797.87	766.51	false	13343.78
0	10	17	26	27	42	463.2113	0.317629	129.7006	-419.767	321.9762	-456.75	18.62761	797.87	811.08	777.65	false	12578.28
0	10	17	26	27	41	514.3178	0.337413	108.9251	-375.659	281.2651	-399.862	12.24703	811.08	834.38	805.68	false	13097.42
4	8	15	28	29	42	629.5564	0.11714	185.4499	-76.9514	-130.225	-685.659	11.09869	834.38	842.5	843.5	true	12553.26
4	10	17	18	28	36	322.5479	0.111618	150.7461	-662.632	447.1471	-156.03	15.00047	842.5	835.48	831.61	true	12461.49
4	10	17	18	28	36	317.7705	0.112182	151.7993	-661.211	443.8282	-154.458	14.68004	835.48	815.55	830.94	true	12445.66
4	8	23	28	42	49	-398.692	0.109048	207.4708	-208.818	-123.114	12.14508	5.556453	815.55	825.16	814.45	false	12302
4	8	23	28	42	49	-430.374	0.107231	211.0351	-209.635	-118.043	11.91623	5.661752	825.16	856.56	817.82	false	12338.86
3	9	17	18	41	49	-1206.68	0.180526	115.3602	-707.9	527.7604	7.922055	8.248262	856.56	841.5	833.44	true	13354.76
1	10	23	28	42	48	709.1501	0.254677	140.8699	-178.231	-156.028	11.60655	-0.62731	841.5	852.06	857.2	true	13168.18
1	10	23	28	42	48	713.5934	0.253725	141.2197	-176.021	-159.327	11.59133	-0.61482	852.06	865.3	836.64	false	13161.6
1	10	17	18	41	49	-579.798	0.229863	59.9233	-744.991	521.9866	8.289558	5.623355	865.3	869.6	840.03	false	13401.41
1	10	17	18	21	49	-712.176	0.238142	76.09072	-423.287	522.0082	-104.531	6.686992	869.6	832.39	856.84	true	12606.65
1	10	17	18	21	49	-691.704	0.229501	76.36001	-460.828	593.4312	-124.495	6.742032	832.39	850.08	822.13	false	13047.94
1	10	28	29	48	49	-155.888	0.258647	141.473	-103.67	-269.876	-0.72236	5.438645	850.08	843.55	841.69	true	13058.77
1	10	28	29	48	49	-152.605	0.25867	141.927	-103.505	-271.879	-0.72873	5.421622	843.55	851.92	844.55	true	13061.69
1	10	28	29	48	49	-34.1257	0.266196	135.8959	-109.764	-307.161	-0.76958	5.003122	851.92	866.23	834.1	false	12890.57

Appendix 3: ICFPF Conference Paper (Regan et al., 2014)

Novel GPU Approach In Predicting The Directional Trend Of The S&P 500

A.J. Regan, F.J. Lidgey, M. Betteridge, P. Georgiou, C. Toumazou, K. Hayatleh, J.R. Dibble

***Abstract*— Our goal is development of an algorithm capable of predicting the directional trend of the Standard and Poor's 500 index (S&P 500). Extensive research has been published attempting to predict different financial markets using historical data testing on an in-sample and trend basis, with many authors employing excessively complex mathematical techniques. In reviewing and evaluating these in-sample methodologies, it became evident that this approach was unable to achieve sufficiently reliable prediction performance for commercial exploitation. For these reasons, we moved to an out-of-sample strategy based on linear regression analysis of an extensive set of financial data correlated with historical closing prices of the S&P 500. We are pleased to report a directional trend accuracy of greater than 55% for tomorrow (t+1) in predicting the S&P 500.

***Keywords*—** financial algorithm, GPU, S&P 500, stock market prediction

Introduction

Current methods of forecasting financial markets require computationally intense algorithms because the parameter inputs needed to create a meaningful prediction are extremely large. Two of the most popular methods used are Artificial Neural Networks (ANNs) [1] and Support Vector Machines (SVMs) [2], which are able to predict financial markets with some success. Both approaches have inherent advantages and disadvantages. An ANN involves a network of processing artificial neurons that can exhibit complex global behavior, determined by the connections between the processing elements and element parameters [3]. The disadvantage of the ANN approach lies in scalability; whilst an ANN is possible to implement in hardware, it is computationally inefficient and power hungry.

The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input, making the SVM a non-probabilistic binary linear classifier [4]. SVMs are not readily scalable and cannot be easily implemented onto a dedicated hardware [5], and hence we believe are unsuitable for predicting the directional trend of the S&P 500.

The aim of this work is to develop a forecasting algorithm for financial markets, which overcomes the scalability limitations inherent in both ANNs and SVMs. The expectation is that this should be able to process extremely efficiently in parallel, and at great speed, to obtain meaningful out-of-sample results. The S&P 500, which comprises the 500 largest US companies based on market capitalisation, was chosen as the test index. The scale and complexity of the hugely liquid S&P 500 creates an incredibly difficult entity to predict.

The implementation is capable of significant gains when choosing optimal parameters for forecasting future financial data. We have adopted a radically different approach in our work, namely to predict tomorrow's value of the index by looking closely at a wide range of financial factors that influence the financial performance of the member companies within the index, and link the predicted values back to the past values of the index. The values of these financial market data (FMD inputs) are readily available and include items such as currency pairs, key commodity indices and other financial indices. These FMD inputs are then used in a linear regression algorithm to compute tomorrow's (t+1) value of the S&P 500 index. Changing the number of FMD inputs affects the directional accuracy of the prediction result; increasing the number of FMD inputs significantly increases the computational complexity and hence the time taken to compute a daily result. Another key aspect of our algorithm is that its architecture is intended for commercial application. Therefore, it is imperative that each prediction result must be computed in less than 24 hours. This is so a result can be obtained today (t) for tomorrows (t+1) US market open.

Architecture of the Algorithm

Development of financial forecasting algorithms that depend entirely on historical data are referred to as in-sample modeling [6]. Algorithms that rely entirely on in-sample modeling are effectively trend based. However, as Granger [7] stated, “one of the main worries about the present methods of model formulation is that the specification search procedure produces models that fit the data spuriously well, and also makes standard techniques of inference unreliable”. Out of sample testing is essential to guard against curve fitting [8]. Many authors have attempted to use an in-sample approach with moderate success. However, the fundamental assumption that the future is entirely determined by the past misses the facts that an index such as the S&P 500 is an aggregation of the performance of 500 individual companies. In recognizing this limitation, we have decided to develop an algorithm based on an alternative methodology.

The key elements that underpin our system are:

1. Identification of 51 financial market data (FMD) inputs, including other indexes, currency pairs, swap rates, etc., that we have proved influence the movement of the S&P 500.

2. The use of an extensive historical data set (actual daily closing prices of the chosen 51 FMD inputs and S&P 500).

3. The ability to compute this large data set (comprising more than 12.7 billion combinations) in a time frame of less than 24 hours.

The data set is fed into a linear regression algorithm to determine the predicted value of tomorrow’s (t+1) S&P 500 closing price.

Distributed Algorithm

What we require is to rapidly compute the best results of a given subset of parameters. In order to reduce the complexity of the algorithms architecture, the available data was inspected by A.J. Regan, an experienced market trader, and reduced down to 51 key FMD inputs. Having identified these 51 significant market parameters for forecasting the directional trend of the S&P 500, the result is a mathematical function that is computationally intense. Fig. 1 displays the number of possible combinations increasing as each FMD input is added. Increasing the number of FMD inputs above 10 produces an unmanageably large data set for even the fastest computational technology to process within our stringent 24-hour time limit.

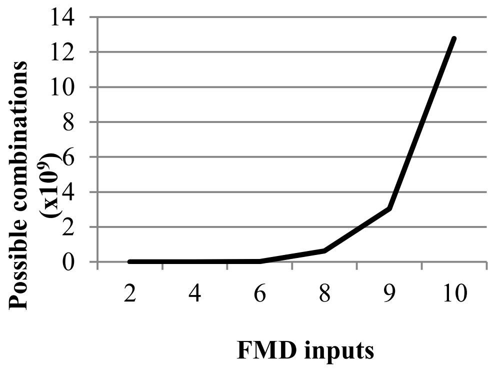


Fig. 1 Possible number of RPA combinations vs. FMD inputs

The requirements for a distributed implementation of the S&P 500 consist of two key components, a regression core, which forecasts using a given set of parameters, and a data bus, which is used to shift information in a daisy chain fashion to compare all the results. This architecture can be seen in Fig. 2.

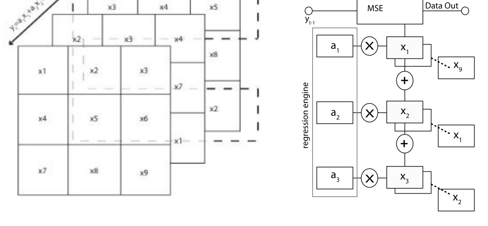


Fig. 2 Distributed architecture for forecasting the S&P 500 for 3 FMD inputs. (left) Regression core, (right) data bus for comparison

The regression core computes a simple linear regression based on the parameters described by (1):

$$y_t = a_1 x_1 + a_2 x_2 + a_3 x_3 \tag{1}$$

where y_t is our forecast, x_n is the input data and an are the weights which are optimized by computing the minimum mean squared error (MSE) of y_t with y_{t-1}.

The aim is then to select the 'best' FMD input parameters x_1, x_2, x_3 (in this case) from a given subset of FMD input parameters, 51, which affect S&P 500. The algorithm works on the basis that it can compute the best MSE for that given set. To select the best parameters, a data bus is used which shifts the input parameters such that a regression is computed for all possible combinations.

As is shown in Fig. 1, the initial architecture will compute in parallel 10 possible combinations at a time, and then sequentially cycle through the FMD input parameters until all possible combinations are evaluated. Throughout this process the optimum set of parameters yielding the minimum MSE is constantly saved. These are then used to conduct the forecast.

Proof of Concept

In order to test the feasibility of the algorithm, initially it was programmed in MATLAB and demonstrated on a 12-core Central Processing Unit (CPU). Results shown in Table 1 confirm the functionality of the algorithm achieving a directional efficiency of greater than 50% for a minimum of six inputs. However, the time to find the best six inputs was extremely long given the limitations of processing in MATLAB. In order to fully exploit the potential of the algorithm and to conduct a more exhaustive study of the optimum number of input parameters in less than our 24-hour time constraint, a more computationally powerful hardware platform was needed.

TABLE I

DIRECTIONAL RESULTS, ACCURACY AND SPEED OF ALGORITHM ON A PC BASED SYSTEM

51 FMD Inputs data	1	2	4	6
Directional Result	11/30	14/30	15/30	17/30
RMSE	28.57	33.02	24.784	19.446
Ave. Time Elapsed	0.02760	.66196	16.3922	36096.65

GPU Based Computational Engine

In order to successfully meet the requirement of handling such a large data set to complete the prediction target on time, we decided to adopt a novel Graphics Processing Unit (GPU) based computational engine which we anticipated would overcome the limitations of using MatLab on a conventional CPU system.

To support this, we decided to build a bespoke hardware unit, which consisted of a 6-core CPU (i73930K) and 3 GPU cards (EVGA GeForce GTX TITAN [9]), supported with 32GB of RAM. Each of the 3 GPU cards houses 2688 cores. The software is designed so that the CPU first initialises and organises the data, before feeding to the 8064 GPU cores, which undertake the extremely intensive processing in parallel. The algorithm is particularly suited to a GPU approach as we were able to re-engineer the algorithm from a linear execution path (as it was in MatLab), to a parallel execution path. This enabled us to utilise the full 8064 cores, all of which calculate in parallel, before passing the results back to the CPU for analysis.

In order to fully utilise the performance of the 8064 cores, a number of items were critical and needed to be thoroughly explored: for instance it is extremely important to organise the pipeline and the memory for optimum performance because severe cache misses will stall the pipeline and be extremely detrimental to performance. The modern GPU has been designed around handling very specific data, namely vertices, textures and shading models, and is unlike a purely CPU architecture, which has been designed to efficiently handle a huge variety of memory accesses. A deep understanding of this process allowed us to optimise and reconfigure the data set to be 'GPU friendly' in a format that perfectly suits its architecture for use in financial prediction. To highlight the importance of organising the data, it should be noted that in early tests almost 90% of theoretical performance was lost due to the GPU pipeline stalling caused by poor memory configuration.

Through this extensive optimisation of our GPU architecture, we were able to achieve a sufficient speed up to meet key element (3), reducing the initial MatLab processing time which was considerably over 24 hours, by approximately 95% with a non-optimised GPU and further reducing the procession time by a further 87% on the fully optimised GPU, as shown in Fig. 3.

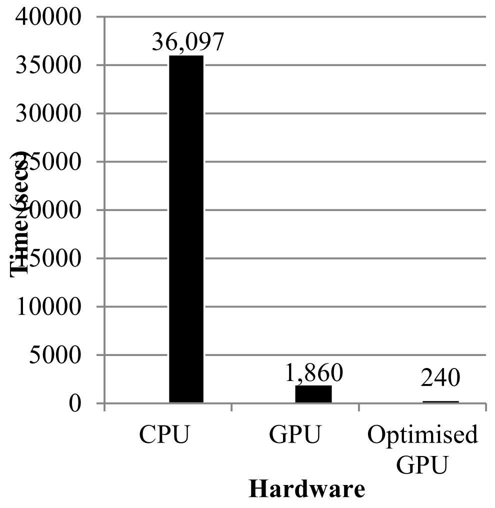


Fig. 3 Processing time of algorithm – 6 FMD inputs

Results and Evaluation

The In achieving our final and most accurate directional trend result in predicting the S&P 500, it was necessary to conduct an extensive range of tests. We explored a number of trade-offs which were tested using an 8-year data set of both the S&P 500 closing price and the chosen 51 FMD inputs. We focused on three significant trade-offs that we believed would influence the result of our algorithm to establish an optimum result in predicting the S&P 500:

1) Testing to determine the optimum number of historical days data that is used to create each prediction cycle. The optimum was established at 60 days.

2) Testing to determine the optimum number of FMD inputs to be selected when computing the algorithm. This varied from 1 to 10 and we established 9 to be optimal.

3) Testing whether a lag exists between the movement in value of each FMD input and the price of the S&P 500. We therefore tested establishing a prediction for tomorrow (t+1) to four trading days later (t+4). We concluded that no lag exists, hence (t+1) remains the optimum.

Having established the optimum parameters for our algorithm and in doing so generating a successful directional trend of greater than 55%, it is important to explain the architecture of each individual test cycle. Depending on market conditions at the time, the FMD inputs to form the basis of the prediction change. Each daily prediction cycle produces a ‘Figure of Merit’ (FoM), a weighted total of the chosen FMD inputs for that individual (daily) cycle. Fig. 4 displays the FoM in the form of a pie chart.

Fig. 4 represents one 60-day cycle computed by our algorithm. This cycle used the prior 60 days ending at time = t = 24th November 2008 to predict (t+1) = 25th November 2008. This particular test was carried out selecting 6 FMD points (of the total set of 51), these are the 6 inputs with the lowest MSE against the daily movement of the S&P 500 closing price.

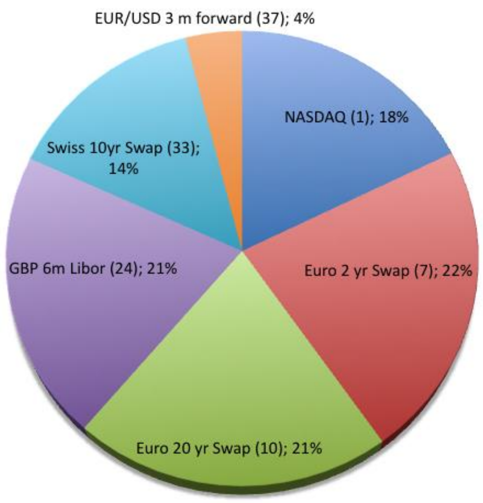


Fig. 4 Figure of Merit weighted total – 6 FMD inputs

Fig. 4 represents one 60-day cycle computed by our algorithm. This cycle used the prior 60 days ending at time = t = 24th November 2008 to predict (t+1) = 25th November 2008. This particular test was carried out selecting 6 FMD points (of the total set of 51), these are the 6 inputs with the lowest MSE against the daily movement of the S&P 500 closing price. It is important to note that depending on daily changing economic conditions, this weighted total will change; different inputs representing different weightings will emerge as the economic climate alters. This dynamic and automatic feature of our algorithm is imperative and ensures that the algorithm is constantly adapting to the ever-changing economic environment by producing an entirely different weighting of FMD inputs each day (cycle). This approach is particularly important when predicting the complex aggregated S&P 500 index.

As a result of establishing the optimum parameters for the algorithm, we can confirm that we have obtained a directional trend of greater than 55% in predicting the S&P 500 at (t+1).

Conclusion and Future Work

A novel architecture algorithm for predicting S&P 500 has been designed and implemented. The data that needs to be processed within a 24-hour period demands a very high speed and highly parallel computation engine which was realized using an advanced GPU design, optimised to meet our particular requirements. The work we have presented here is on going. The authors plan to refine the design further to improve accuracy and also and adapt it for other financial market prediction tasks, such as movement of currencies and other key stock indices.

Acknowledgments

The authors gratefully acknowledge the support received from Corvus Capital and Preciousbluedot Ltd for their contribution to this work.

References

[1] Hamid, S. “Primer on Using Neural Networks for Forecasting Market Variables” Working Paper No. 2004-03

[2] Sapankevych N. and Sankar R., “Time Series Prediction Using Support Vector Machines: A Survey,” Computational Intelligence Magazine, IEEE, vol. 4, no. 2, pp. 24-38, May 2009

[3] Veerabhadram, P., Lombard, A. and Conradie, P. “Artificial Neural Network” International Journal of Scientific & Engineering Research, vol. 3, no. 2, February 2012

[4] Yin, Y., Han, D. and Cai, Z. "Explore data classification algorithm based on SVM and PSO for education decision." Journal of Convergence Information Technology 6.10, 122-128, 2010

[5] Prosser, B et al. "Person Re-Identification by Support Vector Ranking." BMVC. Vol. 1. No. 3. 2010.

[6] Khan, A., Bandopadhyaya, T. K., & Sharma, S. “Genetic algorithm based backpropagation neural network performs better than backpropagation neural network in stock rates prediction” Journal of Computer Science and Network Security, 8(7), 162-166. 2008

[7] Granger C. “Modeling economic time series: Readings in econometric methodology”, Oxford University Press: Oxford, UK, 1990

[8] Inoue, A. & Kilian, L. “In-sample or out-of-sample tests of predictability: Which one should we use?” Econometric Reviews, 23(4), 371-402, 2005

[9] EVGA, 2014. EVGA GeForce GTX TITAN BLACK Superclocked http://www.evga.com/Products/ProductList.aspx?type=0&family=GeForce+TITAN+Series+Family&chipset=GTX+TITAN+BLACK [Accessed 26.06.14]

Appendix 4: Table – daily output results of RPA

Below is a template of the RPAs daily output results of the RPA

RPA S&P 500 daily outputs						
[Insert Date]	Chosen input	% of chosen inputs	S&P close price value (t-1)	S&P close price value (t)	RPA S&P prediction value	Actual S&P % movement (t-1 to t)
Input 1						#DIV/0!
Input 2						#DIV/0!
Input 3						#DIV/0!
Input 4						#DIV/0!
Input 5						#DIV/0!
Input 6						#DIV/0!
Input 7						#DIV/0!
Input 8						#DIV/0!
Input 9						#DIV/0!