Computational Intelligence for Analysis Concerning Financial Modelling and the Adaptive Market Hypothesis.

Matthew Butler

Ph.D. Thesis

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

University of York
Computer Science

December 2012

Abstract

This thesis concerns the field of computational intelligence (CI), an important area of computer science that predominantly endeavours to model complex systems with heuristic algorithms. Heuristic algorithms from CI are generally nature or biologically inspired programs that iteratively learn from experience. More specifically, the research focuses on the overlap between the fields of CI and financial analysis, where the financial markets (in the form of financial time series) provide the complex system of interest. Therefore the objective of the thesis can be summarized as a exercise in improving the abilities of CI algorithms for modelling financial time series.

CI has been applied to a whole spectrum of domains where techniques are developed at a more general level and then applied to a particular application area or complex system. The financial markets are somewhat unique. Unlike other complex systems in nature, the financial markets are of our own creation and their evolution is a by product of human nature, where the beliefs and bias of the participants (humans) in the complex system (financial markets) govern how the system behaves. This is in contrast to many complex systems where, for example, the opinions of experts have no effect on the outcome. Thus, the motivation of this work is to quantify meaningful characteristics (behaviour) of the financial markets as a means to improve and understand how heuristic algorithms respond to them. This process of applying more scrutiny to the analysis of the application area, yields an approach to algorithm development that takes into account the unique characteristics of the market.

To achieve this goal the thesis is structured into three sections that comprise four contribution chapters. The contribution chapters are labelled: validity, implications and innovations and each is motivated by a separate research question. The validity chapter is based on determining a reasonable characterization of the financial markets. This includes a detailed literature review of the popular competing market theories as well as some new innovative tests. There is not a general consensus as to which theory is correct but from a computational intelligence perspective the adaptive market hypothesis (AMH) is revealed

as a reasonable characterization of the financial markets and one that provides quantifiable characteristics to be utilized in following chapters.

The implications chapter concerns testing the effect of implications of the AMH, if any, on the robustness of models derived from CI. Specifically three implications are examined, i.e., variable stationarity, variable efficiency and the waxing and waning of investment strategies. The experiments concerned six algorithms from four of the major paradigms in supervised learning. The results from each of the studies demonstrated that the implications of the AMH affect CI derived models. This conclusion reveals that the unique properties of the financial markets should be taken into account when applying CI algorithms for modelling and forecasting.

The two chapters concerning innovations explore how CI techniques can be improved based on the results from the validity and implications chapters. The first chapter (chapter 6) concerns the development of a meta learner based on the implication of the waxing and waning of investment strategies, the meta learning algorithm called LATIS (Learning Adaptive Technical Indicator System) is a blend of micro and macro modelling perspectives and allows for online adaptive learning with an interpretable white box framework. The second innovations chapter (chapter 7) concerns the discretization of financial time series data into a finite alphabet. A discretization algorithm is developed, which extends an existing state-of-the-art algorithm to handle the characteristics of financial time series. The proposed algorithm, called alSAX (adaptive local Symbolic Aggregate approXimation), is demonstrated to be superior in terms of its symbolic mappings, in relation to a gold standard, and in the popular time series subsequence analysis task. Additionally, an invalid theoretical assumption of the existing algorithm is revealed. The flaw in the algorithm is discussed and its impact is determined based on the characteristics of the time series and the parameters of the algorithm. From the analysis, the thesis offers viable fixes to compensate for the flaw, where the suitability of the fixes are dependent on the problem domain and objective of the data mining task.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank the following people whom without their help the completion of this thesis would not have been possible. Thanks are due to my supervisor, Dr. Dimitar Kazakov, for his helpful encouragement and guidance throughout the PhD process.

I would also like to thank the Department of Computer Science and the University of York for granting my scholarships which made it possible to fund my PhD studies.

My sincere thanks go to my friends and family whom provided support and encouragement throughout my PhD, and especially my mother and father in-law (Wade and Karen), my sister (Gwen) and her family and my mother (Jane).

Finally, a very special and certainly well deserved thanks to my wife Vanessa, who's love and support gave me the confidence and the ambition to succeed.

Declaration

This thesis has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree other than Doctor of Philosophy of the University of York. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references.

List of Publications

[1] M. Butler and D. Kazakov. *A Learning Adaptive Bollinger Band System.* In proceedings of IEEE Computational Intelligence for Financial Engineering and Economics (CIFEr 2012), New York, NY, March 29-30, 2012.

[2] M. Butler and D. Kazakov. *Testing Implications of the Adaptive Market Hypothesis via Computational Intelligence*. In proceedings of IEEE Computational Intelligence for Financial Engineering and Economics (CIFEr 2012), New York, NY, March 29-30, 2012. **Won the Best Student Paper Award**.

[3] M. Butler and D. Kazakov. *The Effects of Variable Stationarity in a Financial Time-Series on Artificial Neural Networks*. In proceedings of IEEE Computational Intelligence for Financial Engineering and Economics (CIFEr 2011), Paris, France, April 11-15, 2011.

[4] M. Butler and D. Kazakov. *Particle Swarm Optimization of Bollinger Bands*. In ANTS Conference, Brussels, Belgium, September 8-10, 2010.

[5] M. Butler and D. Kazakov. *Modeling the Behavior of the Stock Market with an Artificial Immune System*. In proceedings of IEEE Congress on Evolutionary Computation (CEC 2010), Barcelona, Spain, July 18-23, 2010.

Papers Under Review

[1] M. Butler and D. Kazakov. *Creating a level playing field for all symbols in a discretization: Updating SAX for lowly correlated time series*. Under review at the journal of Data Mining and Knowledge Discovery.

[2] M. Butler and D. Kazakov. *A SAX Discretization of Non-stationary Data with Periodic Departures from a Gaussian Distribution*. Under review at the IEEE Transactions on Knowledge and Data Engineering.

[3] M. Butler and D. Kazakov. *Forecasting Non-stationary Data with Artificial Neural Networks*. Under review at Neurocomputing.

[4] S. Mazzi and M. Butler. *Dynamic risk analysis with incomplete information in an adaptive market*. Under review at the Journal of Business and Economic Statistics.

For Vanessa.

Part I

Introduction and Background

Chapter 1

Introduction

This thesis concerns the field of computational intelligence (CI), an important area of computer science that predominantly endeavours to model complex systems with heuristic algorithms. Heuristic algorithms from CI are generally nature or biologically inspired programs that iteratively learn from experience. More specifically, the research focuses on the overlap between the fields of CI and financial analysis, where the financial markets (in the form of financial time series) provide the complex system of interest. Therefore the objective of the thesis can be summarized as a exercise in improving the abilities of CI algorithms for modelling financial time series.

1.1 Motivation

The financial markets are a critical component of all modern capitalist economies and forecasting of financial assets and indicators is an important task for economists, financial professionals and academics alike. Several studies have applied CI algorithms to modelling and forecasting the financial markets with varying degrees of success. By far the most represented learning paradigm in the literature is supervised learning (SL) where an algorithm is provided with labelled data that is of an input-output format, however, other areas such as unsupervised learning, population-based optimization and reinforcement learning are also present. The main distinction of this work from the related literature is that the complex system accounts for more than just an application area. Thus the motivation of this work is to quantify meaningful characteristics (behaviour) of the financial markets as a means to improve and understand how heuristic algorithms respond to them. Unlike other complex systems in nature, the financial markets are of our own creation and their evolution is a by product of human nature, where the beliefs and bias of the participants (humans) in the complex system (financial markets) govern how the system behaves. This is in contrast to many complex systems where, for example, the opinions of experts have no effect on the outcome, e.g. the ocean and marine biologists' attempts to forecast the size of the cod stock in the Atlantic. Regardless of how many experts believe that the current conditions will lead to an increase in the number of cod, those beliefs do not influence that outcome. The same cannot be said for financial systems where throughout the recorded history of modern markets we have seen speculative bubbles form and bust as a result of crowd like mentality. Ignoring the unique nature of this complex system yields a sub-optimal approach to understanding it when using computational intelligence.

1.2 Problem Statement

This stance that the complex system requires more scrutiny as a vehicle for algorithm development and improvement leads to a formal problem statement:

Improve computational intelligence techniques for financial modelling through a better understanding of the interactions between heuristic algorithms and the unique characteristics of financial time series.

This can be expressed by three research questions explored in the thesis:

1. What is a reasonable characterization of the complex system we are modelling?
2. What implications does this hold for computational intelligence?
3. What improvements can be made to compensate for these implications?

1.3 Machine Learning

The term machine learning (ML) defines an area of AI that develops algorithms that can learn from examples. Common tasks the algorithms would be learning are *classification*, *regression* and *anomaly detection* . The classification task is where a set of data belongs to two or more distinctive classes and the algorithm attempts to learn which class a given observation in a member of. Where membership in a class can be determined by a set of measurable characteristics, the measurable characteristics form the attributes of the data with the class membership being the class attribute. This representation defines an input-output relation between what is observed and what the desired outcome is. Another way of describing it, is that the inputs are the independent variables and the output is the dependent variable. One observation from a set of training data contains the input attributes and a class attribute and is commonly referred to, in the literature, as an instance, exemplar or tuple. The regression task is also referred to as level estimation and is the same as the statistical approach. In this case the dependent variable is real-valued and typically does not belong to one class or another. In this case the output value may be a scalar or if the output defines a mult-dimensional state it may be a vector of real-valued scalars. The final task in anomaly detection which is similiar to classification where the algorithm is learning to identify abnormal behaviour within a system. Generally speaking a system could be categorized by normal and abnormal behaviour, so it is similar to two class classification, but the added difficulty is that abnormal behaviour is generally underrepresented in the training data and therefore different learning techniques need to be utilized to accommodate this added complexity.

In the financial and econometrics domains, time series modelling is a popular task. Time series modelling could be facilitated by any of the common ML tasks, i.e., classification, level estimation and anomaly detection. The most widely used ML algorithm, for time series analysis, in the literature is the artificial neural network (ANN). An ANN is a supervised learning algorithm inspired by the brain and will be described in detail in chapter 2.1.4. However, despite its popularity, nearly every paradigm within AI is represented in the related work and the popularity of these techniques, in general, is also a motivating factor behind the thesis.

1.4 Research Strategy

The research concerns a three stage process based on the questions listed in section 1.2, namely: validity (question 1), implications (question 2) and innovations (question 3).

1.4.1 Validity

In order to gain a better understanding of the interactions between CI and financial time series we require a reasonable characterization of the financial markets. This will allow us to identify characteristics that present non-trivial implications to modelling with CI.

This section on validity explores and attempts to validate a relatively new but popular market theory called the adaptive market hypothesis (AMH). The validation is performed through a discussion (and reproduction) of previous work from the econometrics literature and with new evidence being introduced by two novel tests. The first test explores the idea of non-stationarity in the financial markets and whether or not this characteristic is in fact constant. This test utilizes a framework from previous studies where a unit-root test is iteratively applied to a time series using a sliding window. A unit-root test is used to determine if a time-series is stationary .

Secondly, the *risk-to-reward relationship* of a financial asset is examined and we test if this relationship is constant, as implied by the efficient market hypothesis (EMH), or is time varying as implied by the AMH. The risk-to-reward relationship is the amount of risk associated with a financial asset in relation to the amount of expected return of that asset. The test for this time varying relationship is based on a state space model formulation of the traditional Capital Asset Pricing Model (CAPM) that is fitted to real world data using the diffuse Kalman filter (DKF). The advantage to using a state space representation is that we can allow the parameters of the model to be stochastic and therefore potentially time-varying. However, if the preferred fit for the model is to have constant parameters then this is also admissible.

1.4.2 Implications

Once we are satisfied that we have a reasonable characterization of market behaviour we can explore what the implications of said characterization hold for computational intelligence. In this section three implications of the AMH are tested: variable stationarity, variable efficiency and the waxing and waning of investment strategies. Variable stationarity suggests that the optimal pre-processing step for preparing the data for computational intelligence algorithms is time varying. To test this implication an experiment is performed on simulated and real world data that gauges the performance of an artificial neural network (ANN) in a level estimation task when using different pre-processing steps.

Secondly we examine variable efficiency. Most attempts to validate the AMH from econometricians have focused on the idea of variable efficiency and whether or not "informationally efficient" markets exist. Several metrics have been proposed and are generally concerned with measuring dependence (linear or non-linear) in market index data. From a CI perspective, we are interested in whether or not variable efficiency has any effect on the forecasting accuracy of heuristic algorithms. In other words, can we expect an increase/decrease in the forecasting accuracy of CI algorithms when markets are less/more efficient? A less efficient market implies statistically significant dependence in the time series and therefore opportunities for excess returns. To test this effect a simulated time series is generated that mimics real world data and is then segmented into samples that can be considered either random walks or containing non-linear dependence (as identified by a given statistical test). Next we train and test six supervised learning algorithms on the different sets of data and compare their classification accuracies to determine if an increase is observed in forecasting accuracy.

Thirdly, the implication of the waxing and waning of investment strategies is tested. To facilitate this objective we formalize the implication into a measurable performance metric called *cyclical effectiveness*. Cyclical effectiveness is based on the belief that an active trading strategy should outperform a passive buy-and-hold approach in terms of realized investment returns. A trading strategy is considered to be effective during a given time-period when it is able to outperform the market benchmark (buy-and-hold approach), is active in the market (not just investing in a risk-free rate) and producing a positive return. The investment strategy utilized in the study is a novel CI trading algorithm based on a popular quantitative trading technique (mean reversion). We train the algorithm, called an adaptive Bollinger band (ABB) (described in chapter 5), on a training set using a sliding window, where a new ABB is created for each window. We then test the cyclical effectiveness of this population of ABBs on a test set and observe how often on average an ABB is effective after the time period it is created for.

1.4.3 Innovations

In the final section of the thesis we develop two new algorithms for modelling and forecasting financial time series taking into consideration a number of the implications of the AMH.

The first is largely based on the results from the test for cyclical effectiveness. Given that the validity of signals produced from CI trading models will be cyclical there is a need to indentify when a given trading model should be trusted, or more importantly, when it should not. This leads to the development of an online adaptive learning algorithm that creates and maintains a population of investment strategies that are continually updated as new information is received. A specific implementation of the proposed algorithm is described that is based on the ABBs developed in the chapter 5.

The second innovation concerns the pre-processing or discretization of continuous financial time series into a symbolic (discrete) representation. This pre-processing step has been applied in numerous studies but in this chapter a novel algorithm is developed based on a state-of-the-art technique and which takes into account the variable and non-stationary nature of market characteristics. This algorithm, which transforms a financial time series into a symbolic representation, allows the data to be modelled with the various algorithms from machine learning that require discrete or symbolic data and provides a means for dimensionality reduction. The symbolic algorithm is validated based on tests of the ex-post symbolic distribution it generates in relation to the desired gold standard distribution and the ex-post distributions of alternative approaches.

1.5 Thesis Contributions

As previously stated, the thesis concerns the overlap between two fields, namely, computational intelligence and financial analysis. As a result, the contributions can be categorized as either purely computational intelligence, purely financial analysis or in computational intelligence for financial analysis. This section will highlight the contributions made in the thesis and the category under which they fall. It should be noted that all contributions are quantitative in nature and are generally the result of fully implemented algorithms and experiments.

1.5.1 Computational Intelligence for Financial Analysis

The majority of the contributions fall into this category and are as follows:

1. Demonstrating the effects of variable stationarity on price level estimation of the Artificial Neural Network (ANN).
2. Demonstrating the effect of variable efficiency of supervised learning (SL) algorithms .
3. Demonstrating the existence of cyclical effectiveness in computational intelligence derived financial models.
4. Developing an online learning algorithm for optimizing financial technical analysis.
5. Demonstrating that artificial immune systems (AIS) are a viable modelling technique for financial time series. An AIS is a SL algorithm inspired by the natural immune system.

6. Development of a new technical indicator based on Bollinger Bands (BB) and particle swarm optimization (PSO).

7. Developing a symbolic discretization algorithm for financial time series.

1.5.2 Computational Intelligence

The three pieces of work that are general contributions to computational intelligence as a whole are:

1. Extending dynamic heterogeneous particle swarm optimization (dHPSO) to multi-objective optimization (MOO).

2. Refuting work concerning non-stationary data and the Artificial Neural Network (ANN).

3. Discovering an invalid assumption of the Symbolic Aggregate approXimation (SAX) algorithm and correcting it.

1.5.3 Financial Analysis

These contributions are more important from a financial perspective but some of them would not have been possible without computational intelligence techniques.

1. Revealing the time-varying nature of alpha and beta from the Capital Asset Pricing Model (CAPM) as implied by the adaptive market hypothesis.

2. Revealing the variable nature of stationarity in financial time series as opposed to the commonly held view that it is constant.

3. Reproducing results on variable efficiency from the financial literature.

4. Demonstrating that variable efficiency is a non-trivial consideration for trading models.

1.6 Thesis Organization

The structure and scope of the three parts and eight chapters of the thesis are as follows:

Part I: Introduction and Background

Chapter 1: Introduction

Chapter 1 (this chapter) provides a general introduction to the thesis. This includes the motivation and the research objectives, as well as, the thesis structure and research strategy.

Chapter 2: Methods and Theories

Chapter 2 provides the relevant background on the algorithms and financial theories utilized or discussed in the thesis. This is a literature review of sort but is intended to be much more general and a more specific discussion of related work that directly leads into the research in thesis is covered in chapter 3. The first part of this chapter details the algorithms and learning paradigms that are most represented in the thesis. It then goes on to describe the major financial theories of the financial markets, this section discusses the similarities and differences that define the different camps of thought on what drives the behaviour of markets.

Chapter 3: Literature Review

Chapter 3 is the final chapter of part I, in this chapter a more detailed account of the related work is provided. It provides a literature review of research that directly impacts the work presented in the thesis and generally concerns the overlap between computational intelligence and financial analysis. It is broken up in to sections where each section pertains to a specific contribution chapter in thesis.

Part II: Thesis Contributions

Chapter 4: Validity

The first contribution chapter (chapter 4) presents research that examines the adaptive market hypothesis. This includes the reproduction of work on variable information efficiency in the markets from the econometrics literature, as well as an examination of other characteristics of the markets held to be static under the efficient market hypothesis. It then goes on to describe the work on the time varying relationship of risk and reward. This includes the definition of a state-space approach to a linear financial model and the results from fitting such as model using the diffuse Kalman filter.

Chapter 5: Implications

The second chapter of Part II, chapter 5, examines the effects of the variable nature of certain characteristics identified in the previous chapter and the AMH implication of cyclical effectiveness. This begins with an experiment on Artificial Neural Networks and how variable stationarity impacts the forecasting of simulated and real world time series. It then goes on to describe the experiments on supervised machine learning and if time varying market efficiency has any effect on forecasting with these techniques in terms of classification accuracy. In the final section it discusses a formal definition of the waxing and waning of investment strategies, proposes a novel forecasting algorithm based on particle swarm optimization and technical analysis, and tests this algorithm for cyclical effectiveness.

Chapter 6: Innovations in Technical Anlaysis

Chapter 6 describes the proposed meta-learning algorithm that is based on the results from cyclical effectiveness in chapter 5. The general theory and framework is provided followed by a specific implementation using the novel algorithm proposed in the previous chapter. The specific implementation is then tested in a Monte Carlo simulation and then on real world data.

Chapter 7: Innovations in Discretization

The final contribution chapter, chapter 7, concerns the discretization of continuous time series into a symbolic representation. This begins with a discussion and then an experiment which reveals an invalid assumption of a state of the art algorithm from this field and a fix for this issue. Then it goes on to describe a proposed novel discretization algorithm for mapping a continuous financial time series to a symbolic representation. Due to the specific and time varying nature of financial time series characteristics the benefits of the proposed algorithm are demonstrated on simulated and real world data and compared to alternative techniques from the field.

Part III: Conclusions and Future Work

Chapter 8: Conclusions and Future Work

The final chapter summarizes the contributions of the thesis and examines to what extend the three research questions in the introduction have been satisfied. The key research contributions are highlighted in relation to the thesis objectives and finally the directions for future work are discussed.

Chapter 2

Methods and Market Hypotheses

This chapter will provide an overview of the algorithms and financial theories that are utilized and discussed throughout the thesis. This chapter is a literature review but serves more to provide relevant background of a more a general nature and specific work related to the overlap of computational intelligence and financial analysis will be provided in chapter 3.

2.1 Methods

2.1.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is an algorithm inspired by swarm intelligence developed in 1995 by Kennedy et al. [70]. Swarm intelligence is the biological phenomenon of decentralised collective behaviour in the natural world, such as a flock of birds or a school of fish. It was recognized that collectively such animals displayed intelligence far beyond that of any one individual. A particular individual in the swarm governs its behaviour based on personal experience (perceptions) as well as benefits from the collective knowledge of the swarm as a whole. Mimicking the behaviour of an individual's neighbours turns out to be an efficient and effective way of communicating information throughout a decentralized group where the capacity for communication is unsophisticated by human standards. For example, a school of fish will quickly change direction once a predator is detected, thus each fish is more likely to evade threats even if they do not observe it themselves. The PSO algorithm solves an optimization problem by maintaining a population of particles that "fly" through an n-dimensional solution space and are represented by their position and velocity. The movement of the swarm is governed by a fitness function (as with many biologically inspired algorithms) where a particle's position and velocity are updated by equations 2.1 and 2.2. The *velocity update* is as follows:

$$v_{i,j} \leftarrow \omega * v_{i,j} + c_1 * r_1 * (local_{best\ i,j} - x_{i,j}) + c_2 * r_2 * (global_{best\ i,j} - x_{i,j}) \tag{2.1}$$

where $v_{i,j}$ is the velocity of j^{th} dimension of the i^{th} particle, c_1 and c_2 determine the influence on a particular particle by its optimal position previously visited and the optimal position obtained by the swarm as a whole, r_1 and r_2 are uniform random numbers between 0 and 1. The inertia weight term ω (introduced in [123]) can be a constant, usually between 0 and 1, or a function of time. The position update is as follows:

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j} \tag{2.2}$$

where $x_{i,j}$ is the position of the j^{th} dimension of the i^{th} particle in the swarm. As a means to avoid particles developing very high velocities and "flying" out of the solution space a constraint called maximum velocity (V_{max}) is introduced that essentially sets an upper threshold on jumps between iterations. The pseudo code for maintaining a swarm of particles to navigate a solution space is as follows:

Algorithm 1 Particle Swarm Optimization

```
maxIterations
number_of_particles
number_of_dimensions
initialize particle positions(number_of_particles, number_of_dimensions)
initialize particle velocities equal to 0
initialize global and local best fitness to worst possible(for example 0)
for $i \leqq$ maxIterations do
    evaluate fitness for each particle
    update local and global best
    for each particle do
        update velocity for each dimension (equation 2.1)
        update position (equation 2.2)
    end for
end for
```

2.1.1.1 Heterogeneous Particle Swarm Optimization

A more sophisticated version of PSO called dynamic heterogeneous particle swarm optimization (dHPSO) [34] is used in the thesis. The dHPSO was chosen as it was demonstrated in [34] to outperform the canonical form of PSO on a variety of optimization problems. With dHPSO the position update remains the same (in all but one case, described below) but the calculation of the velocity update is expanded to allow for alternatives. The swarm becomes heterogeneous as each particle in the swarm will have one of five possible velocity update profiles and the swarm is dynamic as the velocity update profile will change if a particle becomes stagnant. The additional velocity updates are as follows:

$$v_{i,j} \leftarrow \omega * v_{i,j} + c_1 * r_1 * (local_{best\,i,j} - x_{i,j}) \tag{2.3}$$

$$v_{i,j} \leftarrow \omega * v_{i,j} + c_2 * r_2 * (global_{best\,j} - x_{i,j}) \tag{2.4}$$

$$v_{i,j} \sim N\left(\frac{local_{best\,i,j} + global_{best\,j}}{2}, \sigma\right) \tag{2.5}$$

$$v_{i,j} \leftarrow \begin{cases} local_{best\,i,j} & \text{if U(0,1) < 0.5} \\ N\left(\frac{local_{best\,i,j}+global_{best\,j}}{2}, \sigma\right) & \text{otherwise} \end{cases} \tag{2.6}$$

where, N and U are normal and uniform distributions respectively. Equation 2.3 is the cognitive only profile where the social component has been removed. This promotes exploration as each particle becomes a hill-climber. Equation 2.4 is the social only profile where the cognitive component has been removed. In effect the entire swarm becomes one large hill-climber. Equation 2.5 is the Barebones PSO where the position update is the velocity update, so:

$$x_{i,j} = v_{i,j}, \quad and \tag{2.7}$$

$$\sigma = |local_{best\,i,j} - global_{best\,j}|. \tag{2.8}$$

Finally, equation 2.6 is the modified Barebones profile. One additional improvement has been made to the dHPSO algorithm where particles that continue to be stagnant after velocity profile changes will be re-initialized randomly in the solution space.

2.1.2 Genetic Algorithms

Genetic Algorithms (GA), developed by John Holland [51], are a range of learning techniques inspired from evolution, known as evolutionary computation (EC). A GA will navigate the solution space by evolving a population of candidate solutions which attempt to improve (as a whole) with each generation. Candidate solutions are evolved, under the *building block hypothesis* [51], by employing processes from evolution: breeding (crossover), mutation and reproduction (reproduction would be more commonly associated at a genetic level, such as mitosis). The building block hypothesis is one of the foundations of GA theory where schema of low order (short length) contain partial optimal solutions and through the evolutionary process this subset of optimal schema are eventually combined to form a complete optimal solution. GAs have been very successful in searching through complex solution spaces and are more adept at escaping local minima than other approaches such as greedy algorithms. The pseudo code for evolving a population of candidate solutions is as follows:

Algorithm 2 Genetic Algorithm

```
maxGenerations
number_of_individuals
length_ind
initialize_population(number_of_individuals, lenght_ind)
evaluate fitness of initial population
for $i \leqq$ maxGenerations do
    select k fittest individuals for reproduction
    create new individuals via crossover and mutation
    evaluate fitnees of new individuals
    replace least fit individuals from population with the newly evolved
end for
```

2.1.3 Genetic Programming

Another evolutionary based algorithm is genetic programming (GP) first introduced by John Koza [73] in 1990. Also population based, GP iteratively learns from previous experience by continually updating its population of potential solutions to a given problem of interest. As the name suggests, the individuals in the population define a working computer program that is essentially a set of functions which describe the mapping from an input domain to a target domain or output attribute. A common form of the program is a decision tree, similar in structure to those described in section 2.1.7, but the methodology for determining the structure of the tree is based on evolution, Darwinian natural selection and a fitness function. The fitness function is designed to effectively evaluate the quality of the solutions in the population.

2.1.4 Artificial Neural Networks

One of the most important and popular paradigms of learning algorithms from artificial intelligence are those based on the biological processes in the brain. Often referred to as Artificial Neural Networks (ANN), these models have all been inspired by how information is believed to be processed by humans. Essentially an ANN is a directed graph, where the nodes are based on the neurons in the brain and the arcs are synaptic weights between them. Information is processed by being fed to an input layer and then transmitted through the graph by the weighted arcs between the nodes. The nodes contain activation functions which allow them to “fire” when adequately stimulated, thus allowing further propagation of the signal. The ANN which is considered to have the greatest practical value [5] is the multi-layer perceptron (MLP) which is an extension of the original ANN the *perceptron* from the 1960s. An MLP can be considered a non-linear function approximator and, provided enough neurons are supplied, then a 2-hidden layer MLP should be able to approximate any signal [43]. The type of ANN utilized in the thesis is a feed-forward MLP and an example of a topology used in the research is displayed in figure 2.1. In this example there are two hidden layers, an input layer and an output layer. The neurons in the hidden layers have sigmoid activation functions (equation 2.10) and the output neuron has a linear activation function (equation 2.11). MLPs utilized in the study are trained for two distinct but related task, the first is classification where the training data is associated with a class attribute (C) that is numeric but discrete, for example C $\in$ {0,1}. The second task is level estimation , where the training data is associated with a target value that is also numeric but is continuous, such that C $\in \mathbb{R}$. This task is analogous to a linear regression and based on the output neuron in figure 2.1 this MLP would be used for a level estimation task.

The process of propagating a signal through an MLP with one hidden layer can be formally described by the following equation:

$$\hat{Y}(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{2.9}$$

where the (1) and (2) represent the input and hidden layers respectively. $\hat{Y}$ is the output of the MLP, $\mathbf{x}$ is an input vector of D real-valued variables, $\mathbf{w}$ is a matrix of weights, M is the number of neurons in the hidden layer, h denotes a hidden layer and σ is the sigmoid activation function given by:

$$\sigma(a) = \frac{1}{1 + \exp^{-a}} \tag{2.10}$$

where a is the sum of each input into the node multiplied by its respective weight. The linear activation function from figure 2.1 which is used in level estimation is calculated as follows:

$$\hat{Y} = \sum_{m=0}^{L} X_m W_m \tag{2.11}$$

where X_m is the output of neuron m the final hidden layer, W_m is the connection weight and L is the number of neurons in the final hidden layer.

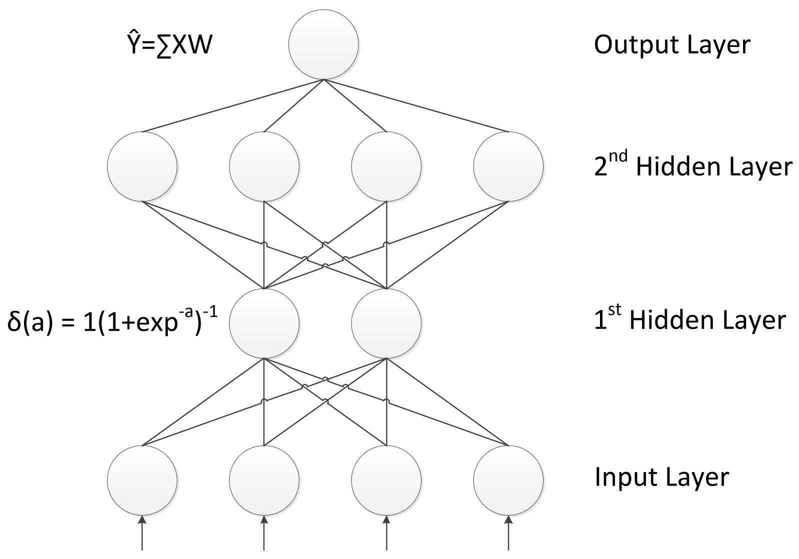

Figure 2.1: A typical feed-forward MLP topology.

2.1.4.1 Training with Back-Propagation

One of the most popular methods for training feed-forward MLPs is the back-propagation (BP) algorithm. The BP algorithm works by propagating the error realized in the output layer back through the network of nodes to adjust the weights using the error gradient. A step-by-step process for implementing BP for a single hidden layer is displayed in algorithm 3, where we define Y_j to be the output of j^{th} neuron in the hidden layer, Y_t is the desired target, α is the learning rate and the i subscript indicates the input layer.

2.1.5 Support Vector Machine

A Support Vector Machine (SVM) is a machine learning algorithm for solving classification, regression and anomaly detection problems. The SVM is also referred to as a maximum margin classifier because the learning procedure attempts to find the largest

Algorithm 3 Back Propagation

```
for each training sample do
    propogate input signal X through network using equation 2.9 yeilding $Y_k$
    calculate error in output neuron: $e_k = Y_k$ - $Y_T$
    calculate the gradient for output neuron: $\delta_k = \hat{Y}[1\text{-}Y_T]e_k$
    calculate weight corrections between output neuron and hidden layer:
    $\triangle w_{jk} = \alpha Y_j \delta_k$
    update corresponding weights: $w_{jk} \leftarrow w_{jk} + \triangle w_{jk}$
    calculate error gradient for neurons is hidden layer:
    $\delta_j = Y_j[1\text{-}Y_j]\sum\delta_j w_{jk}$
    calculate the weight corrections: $\triangle w_{ij} = \alpha X_i \delta_j$
    update corresponding weights: $w_{ij} \leftarrow w_{ij} + \triangle w_{ij}$
end for
```

plane or hyper plane (in hyper-space) that maximizes the distance between different classes of data. In its canonical form the SVM would be used in a two-class classification problem where the plane of maximum margin is constructed in feature space, where the data is assumed to linearly separable. Formally we can demonstrate the intuition behind SVMs by considering the following linear model:

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b \tag{2.12}$$

where $y(\mathbf{x})$ would be a vector of target variables, $\mathbf{w}$ is a weight vector, b is a bias parameter and $\phi(\mathbf{x})$ is feature-space transformation. So an SVM will project a set of training data into a transformed feature-space where the data will not only be linearly separable but also the transformed space will be the unique solution that maximizes the margin. Without the extra criteria of the maximum margin several solutions would satisfy equation 2.12. Achieving the maximum margin is desirable as it guarantees the optimal generalization of the model [131] given the training set with target variables of the form $t_n \in \{\text{-}1,1\}$. In feature space the target values would form the decision boundaries such that the distance between

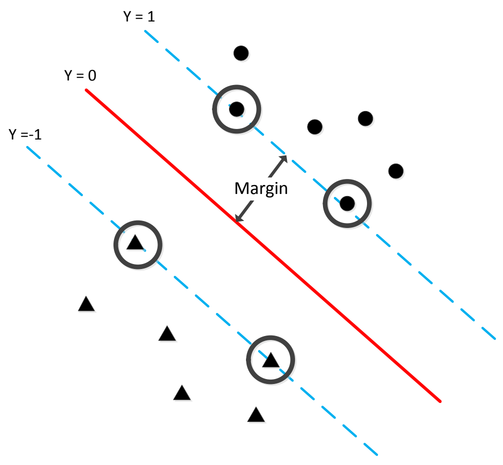

Figure 2.2: The data points falling on the decision boundary form the set of support vectors.

them is maximized, as depicted in figure 2.2. The solution to the maximum margin problem can be found by solving:

$$\underset{\mathbf{w},b}{\arg\max}\left\{\frac{1}{\mathbf{w}}\min_{n}[t_n(\mathbf{w}^t\phi(x_n)+b)]\right\} \tag{2.13}$$

which is in feature space and a direct solution would be difficult to calculate. As such the SVM is generally described in its dual form which uses a kernel trick and lagrange multipliers to solve an equivalent problem that is much simpler. The dual form is:

$$\tilde{L}(a)=\sum_{n=1}^{N}a_n-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}a_na_mt_nt_mk(x_n,x_m) \tag{2.14}$$

where a_n is a Lagrange multiplier and $k(x_n, x_m)$ is the kernel function that yeilds the feature space transformation. Equation 2.14 is maximized with respect to a training set of size N and the contraints:

$$a_n \geq 0, \quad n = 1, ..., N \tag{2.15}$$

$$\sum_{n=1}^{N}a_nt_n = 0. \tag{2.16}$$

Several algorithms have been proposed for solving equation 2.14 and for a more detailed account please refer to [5].

2.1.6 k-Nearest Neighbours

The k-Nearest Neighbours (kNN) algorithm is a simple instance based learner also referred to as a lazy learner generally used for classification and anomaly detection. An advantage of kNN is that no training is required and new unseen instances from an out-of-sample data set are classified based on distances between them and the training data. The distance is defined in feature space and is generally measured using Euclidean distance but other distance measure such as Manhattan and Mahalanobis are also popular. Once the distances between the new data point and the training data have been determined a local neighbourhood of size k is constructed. The local neighbourhood then determines the class to be assigned to the new data point which is generally based on a majority vote. Another desirable property of kNN is that it only has one parameter k which is the size of the local neighbourhood, when k=1 the algorithm is referred to as nearest neighbour (NN).

2.1.7 Entropy-Based Decision Tree

A decision tree is commonly used for classification and is a type of graph that maps input data to specified outcomes by a making a series of decisions based on the instance attributes as the tree is traversed. An entropy based decision tree is one that makes decisions based on entropy or information theory. A popular entropy-based decision tree that is widely available is ID3 [115] or its numeric attribute extension C4.5 [116]. Each node in the tree represents an attribute that splits the training instances into 2 or more segments depending on the instances attribute values.

2.1.8 Naïve Bayes

The Naïve Bayes classification algorithm is a simplified approach to probabilistic modelling of data using Bayes rule. The attributes contained in the training instances are considered to be conditionally independent based on the class attribute. The decision as to what class a given unseen instance is assigned to is based on the prior probability of the class and the conditional probabilities of an attribute on a class. The equation for calculating the probability of assigning a class to a new instance, or the posterior probability, is:

$$P(C_i|A_1, ..., A_n) = P(C_i)P(A_1, ..A_n|C_i) \tag{2.17}$$

where the probability of class i (C_i) given the observed attribute ($A_1, ..., A_n$) is just the product of the probability of the class ($P(C_i)$) and the conditional probabilities of the attributes given the class ($P(A_1, ..., A_n|C_i)$. For each class this quantity (equation 2.17) is calculated and the class yielding the highest probability is assigned to the new data point.

2.1.9 Artificial Immune Systems

The Artificial Immune System (AIS) is a biologically inspired algorithm which draws its inspiration from the vertebrate natural immune system (NIS). The NIS can be generalized as a system which continually protects the body from harmful invaders, called antigens, and keeps the body in an equilibrium state. The AIS is not an exact model of how the NIS interacts with a living entity; rather, it draws on principles from the immune systems which are a natural fit for machine learning. The notion of “self” and “not self” is one of the more popular principles employed in AIS algorithms; it means that the immune system is able to recognize objects which are not harmful, such as red bloods cells, categorized as "self", commonly referred to as antibodies and antigens which are a threat to the wellbeing of the system, such as viruses. This principle is the main underlying theme to AIS for supervised learning , although the algorithms go into much more detail and employ other principles from the NIS to accomplish this goal; in all cases, the notion that the immune system can effectively distinguish between two distinct objects is the very basis of using it for an analogy in machine learning. In the NIS, if an antibody is activated by an antigen, then the two bind and the antigen is destroyed, eliminating the threat. How the immune system actually accomplishes this task forms the principles which influence the inner workings of AIS algorithms.

Negative selection (NS) was first used in AIS for recognizing "self" and "not self", with this type of learning the algorithm only uses examples of one-type of object such as in positive-only learning. NS comes from the thymus which is an organ responsible for generating the T-Cells which circulate the body looking for invading antigens. The thymus continually creates T-cells which are first held in the thymus and tested to see if they are activated by any of the "self", which would mean that they recognize and react to "self", if they do than they are destroyed, otherwise they are released into the body as they will only be activated by "non-self" pathogens. A T-cell is activated if its degree of similarity is sufficiently close to an antigen and this degree of similarity is determined by an affinity measure. This type of learning can be very ineffective for supervised learning as the "non-self" space can be quite large and require a massive number of T-cells to accurately map it. Also by ignoring counter examples and only training on one type of data a large amount of useful information is ignored. To improve upon the principle of NS, the algorithm implemented in the thesis, which will be introduced next, is referred to as clonal selection-based AIS where training is conducted with both negative and positive exemplars. The algorithm learns and builds a memory of negative and positive exemplars and later uses this experience to classify new antigens which enter the system or, in other words, new unlabelled examples to which the algorithm is exposed.

One of the most well known clonal selection algorithms is the Artificial Immune Recognition System (AIRS) developed in [133]. The metaphors from the NIS employed in AIRS are antibody-antigen binding, affinity maturation, clonal selection process, resource competition and memory acquisition. The algorithm has four main stages, initialization, memory cell recognition and *ARB* generation, resource competition and revision of resulting memory cells. Artificial Recognition Balls (ARB) are a combination of a feature vector with a class attribute used to recognize possible antigens (training data). Although the implementation in the thesis is based on AIRS it might be different from existing AIRS implementations in certain details. A high-level algorithm that is used to evolve a set of memory cells is as follows:

Algorithm 4 Artificial Immune System

```
L number_of_antigens
all data is normalized ∈ 0, 1
L antigens seed ARB and memory cell (MC) pools
for each training examplar do
    MC identification and ARB generation
    Competition for resources and development of a candidate MC
    MC introduction
end for
for each test case do
    apply kNN algorithm for $K$>0 to determine local neighbourhood of MCs
    classify test case by majority vote
end for
```

As can be seen from the steps above, once the set of memory cells (MC) has been created the classifier works as a nearest neighbours algorithm where the memory cells act as the raw data would in the k-Nearest Neighbours (kNN) algorithm.

2.1.10 Learning Classifier Systems

In this section a moderately detailed description on a Learning Classifier System (LCS) is provided. Although LCSs are not used in the thesis, an algorithm developed in chapter 7 is based on them and some background on LCSs will aid in its explanation. A learning classifier system is a population based adaptive system which combines genetic algorithms (GA) and reinforcement learning (RL). In many ways it is similar to an AIS where a population of classifiers are maintained that become active and involved in classification when they are stimulated by input data from an environment. Modelling complex systems with an LCS was first proposed by John Holland in [52] and later made a practical supervised learning approach by Wilson [136]. Wilson's eXtended Classifier System (XCS) was the first classifier system based on accuracy that segmented the update of the GA population away from the other learning parameters. Without going into too much detail, the XCS algorithm changed how LCS were used and it's framework still forms the basis of many LCSs developed today. In figure 2.3 we have the general framework for a LCS and the typical steps for a given iteration of the algorithm.

The 10 steps depicted in figure 2.3 comprise the three main components of a LCS: discovery, performance and reinforcement. In the discovery component (steps 4 and

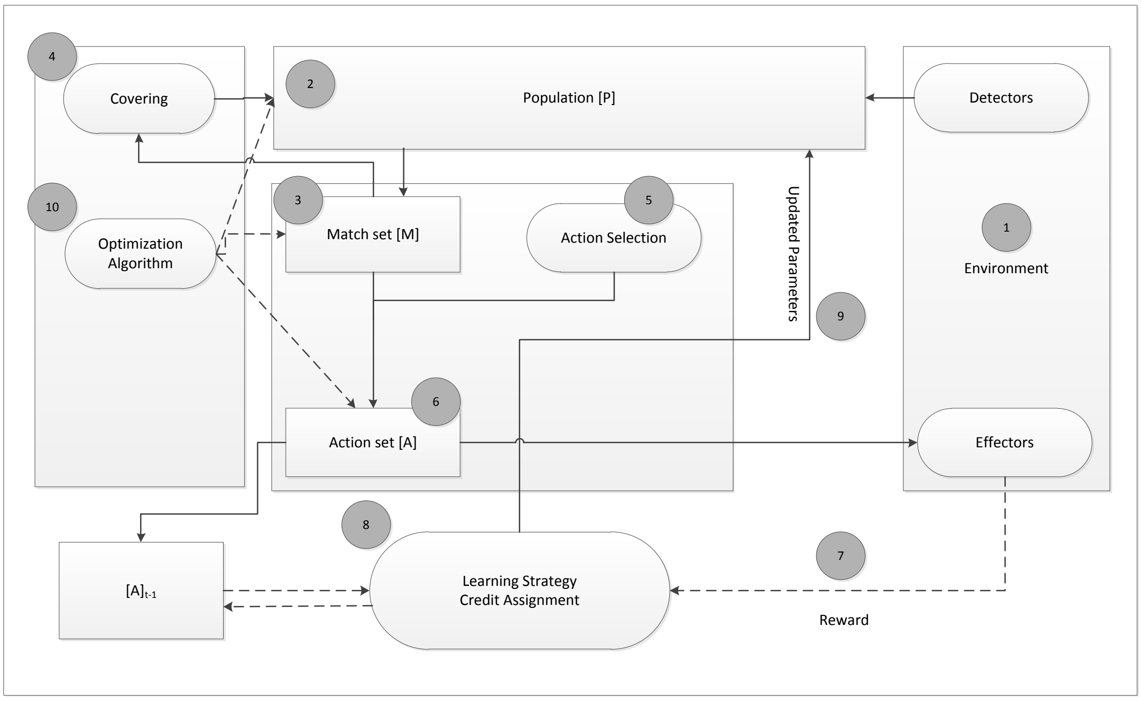

Figure 2.3: A typical LCS framework.

10) the system, guided by the genetic algorithm, creates classification rules of an "*IF condition THEN action*″ format. The rules are learned based on the input data received from the environment (step 1) that is in the standard form for training supervised learning algorithms. The population [P] can be randomly generated then developed online as data is received or an initial batch learning stage can be performed to seed the initial population of the LCS with classifiers. The covering mechanism is used to create new classifiers for [P] when the current data does not satisfy any of the "*IF condition*″s of the current population.

The performance component is responsible for evaluating the classifiers in [P], forming a match set [M] (step 3) and then an action set [A] (step 6). From [M] an action will be chosen (step 5) to be executed by the system for the current iteration. The match set contains all the classifiers from [P] which had their "*IF condition*″s satisfied by the current signal received from the environment. The action set contains the classifiers from [M] whoose "*THEN action*″ matched the chosen action. Finally, *action selection* (step 5) can be determined in a variety of ways but canonically it will be a weighted-fitness evaluation where the action chosen has the highest score.

The reinforcement component is responsible for credit assignment and represents the reinforcement learning portion of the LCS. Based on a reward received from the environment, the parameters of the classifiers from an action set are updated using the

Widrow-Hoff delta rule, equation 2.18:

$$V_i \leftarrow V_i + \beta(\hat{V}_i - V_i) \tag{2.18}$$

where, $0 < \beta \leq 1$ is the learning rate, V_i is the current value of some parameter of the i^{th} classifier and $\hat{V}_i$ is the current estimate of said parameter from the most recent inclusion in [A]. Using equation 2.18 the parameters of the classifiers in [P] can be updated online as new information is received. As we can observe in figure 2.3 the credit assignment step is applied to an action set of time t_{-1} which implies a delayed credit assignment format.

2.1.11 Symbolic Aggregate Approximation

In time series analysis research there is a strong interest in discrete representations of real valued data streams. The discretization process offers several desirable properties such as numerosity/dimensionality reduction, the removal of excess noise and the access to numerous algorithms that typically require discrete data. As a result there is a wealth of literature describing a variety of techniques to facilitate this transformation. These methods can be as simple as equal-width/equal-frequency binning or more sophisticated approaches that are based on clustering [28] and information theory. One approach that emerged over a decade ago and is still (along with its successors) considered state-of-the-art is the Symbolic Aggregate approXimation (SAX) algorithm proposed in Lin et al. [87] [88]. This discretization algorithm was the first symbolic approach that mapped a real-valued time series to a symbolic representation that was guaranteed to lower-bound Euclidean distance. This discovery lead to an explosion of application areas for the SAX algorithm, such as, telemedicine [25], robotics [40], computational biology [1], environmental science [100], network traffic [144] and pattern matching in general [128] [103]. The interest in SAX is also motivated by its easy implementation and intuitive approach to discretization based on the assumption of Gaussian distributions. The SAX algorithm maps a real-valued time series of length n to a symbolic representation of length m where $m < n$ and often $m << n$. In this context, m represents the number of segments the time series is divided up into and the ratio of n/m could be regarded as the compression rate. The three main steps of the algorithm are as follows:

Algorithm 5 Symbolic Aggregate Approximation

```
  Normalize the time series to have $\mu = 0$ and $\sigma = 1$
  convert the time series to PAA
  substitute the PAA segments for symbols based on the standard normal curve
```

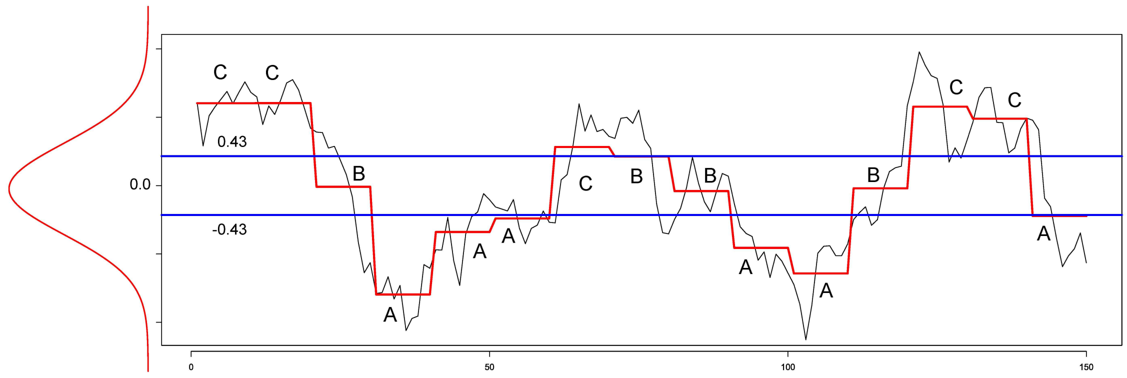

Figure 2.4: A time series of 150 data points converted to a SAX representation of length 15, using an alphabet size of 3. The lines at ±0.43 represent the partitions of the standardized normal curve which yield 3 equally probable regions. The original time series is transformed into the SAX "word" CCBAAACBBAABCCA.

The Piecewise aggregate approximation (PAA) step requires that a time series C of length n is replaced by a vector $\bar{C}$ of length m where each element i of $\bar{C}$ is calculated using the following:

$$\bar{c}_i = \frac{m}{n} \sum_{j=\frac{n}{m}(i-1)+1}^{\frac{n}{m}i} c_j \tag{2.19}$$

Equation 2.19 states that a time-series is divided into m equal size segments, where each segment is then represented by its mean. This representation can also be considered as an attempt to approximate the original time-series with a linear combination of box basis functions [87]. Figure 2.4 illustrates the transformation from a real-valued time-series to it's symbolic representation. From the graph we can observe that a series of original length 150 is mapped into a symbolic sequence of 15 letters which form a "word" [87] from an alphabet of cardinality 3. The distance measure, defined on the symbolic sequence, lower bounds the Euclidean Distance of its real-valued counterpart. For example, for two SAX representations P and Q defined over the same alphabet the distance between those sequences is calculated as follows:

$$MINDIST(\hat{P}, \hat{Q}) \equiv \sqrt{\frac{n}{m}} \sqrt{\sum\nolimits_{i=1}^{m} (dist(\hat{p}_i, \hat{q}_i))^2} \tag{2.20}$$

where the sub function $dist()$ is determined by a lookup table as shown in table 2.1 for an alphabet of cardinality 4.

Table 2.1: A lookup table used by the MINDIST function for an alphabet of cardinality 4.

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

2.1.12 State space models and the Kalman Filter

State space models (SSM) sometimes referred to as dynamic linear models were first developed in the 1960s for the aerospace industry by Kalman and Bucy [64]. SSMs were developed as a means to determine an optimal estimate of some unknown quantity that can only be measured by a less than perfect instrument. Using this suboptimal instrument introduces a degree of measurement error into the observation and therefore the “true” value of the unknown quantity cannot be determined exactly. However, if assumptions are made about the error associated with the measurement then possibly with enough observations a near optimal estimation can be achieved. If the unknown quantity was a single constant variable then we could obtain the maximum likelihood estimate (MLE) by simply averaging the observations, however if the unknown variable is dynamic a more sophisticated approach is required. State space models are commonly used in several areas of computational intelligence and have achieved the greatest recognition in computational linguistics where hidden markov models (HMM) are considered one of the most robust algorithms for part-of-speech tagging and sequence analysis. The HMM is a discrete representation of a SSM where the emission and transition probabilities (between states) would be determined using the forward-backwards algorithm, which is a special case of the Expectation-Maximization (EM) algorithm. For a continuous process such as a time series (the type of process considered in the thesis) a SSM could be represented by two equations: an observation equation (equation 2.21) and a state equation (equation 2.22). For a process $\{Y_t\}_1^n$ for $1 \leq t \leq n$ that is generated by a SSM ($\{Y_t\} \leftarrow$ SSM), the observation equation is:

$$Y_t = Z_t\alpha_t + G_t u_t \tag{2.21}$$

where $u_t \sim (0, \sigma^2 I)$ with $\sigma^2 > 0$. If Y_t is scalar and α_t is a p x 1 vector then Z_t is 1 x p vector that defines the interaction of the state with the observations and G_t is a p x 1 input vector. The state equation for α_{t+1} is:

$$\alpha_{t+1} = T_t\alpha_t + H_t u_t \tag{2.22}$$

where u_t has the same definition as above, $\alpha_0 = 0$ and T_t is the transition matrix. All of the matrices (Z, G, T and H) are non-random and as the subscript implies can be time-varying. The parameters of the SSM of $\{Y_t\}$ can be efficiently estimated using the Kalman filter (KF) and Kalman smoother (KS), which work as a recursive procedure through a set of training data. The KF can be defined by a set of equations for filtering and forecasting the process $\{Y_t\}$. One of the main advantages of the KF is that it is continuously updatable as new observations become available without having to re-train (re-fit) the model with the entire data set. The set of recursion equations defining the KF are:

$$e_t = y_t - Z_t a_t \tag{2.23}$$

$$D_t = Z_t P_t Z_t' + G_t G_t' \tag{2.24}$$

$$K_t = (T_t P_t Z_t' + H_t G_t') D_t^{-1} \tag{2.25}$$

$$a_{t+1} = T_t a_t + K_t E_t \tag{2.26}$$

$$P_{t+1} = (P_t - K_t Z_t) P_t T_t' + (H_t - K_t G_t) H_t' \tag{2.27}$$

where a_t is a predictor of the state α_t and a_t has the variance-covariance matrix P_t, which is an estimate of the error covariance. Equation 2.25 is called the Kalman gain and represents a gain or blending factor that minimizes the *a posterior* error covariance. The matrix D_t can be thought of as a projection of the error covariance ahead and then P_t is the update of the error covariance taking into account the Kalman gain. The Kalman smoother provides an efficient method for smoothing the state estimates, α_t, of the observed series $\{Y_t\}$, where essentially α_t can be predicted using the entire series of observations rather than just those up to time t. Thus smoothing using the KS can be accomplished using the recursion:

$$N_{t-1} = Z_t' D_t^{-1} E_t + L_t' N_t \tag{2.28}$$

$$R_{t-1} = Z_t' D_t^{-1} Z_t + L_t' R_t L_t \tag{2.29}$$

$$L_t = T_t - K_t Z_t \tag{2.30}$$

where $N_n = 0$, $R_n = 0$ and all other quantities are defined in the KF equations above. The KS works in reverse to the KF where it begins a time n and works backwards to t=0. The smoother is not used for making real-time predictions of the state or the observations but allows for a better estimate. However, once the KS has been applied the improved estimates of the state can be used for forecasting.

2.2 Market Hypotheses

In this section a description of the three main stream market hypothesis will be provided that includes related work from the financial literature.

2.2.1 Efficient Markets and Random Walks

Probably the most cited paper in the field of finance and economics is the thesis of Eugene Fama which examines the theory of efficient financial markets [35]. His research formally developed the idea of the efficient market hypothesis (EMH) a ubiquitous concept in modern finance. The majority of research concerning financial forecasting will make reference to this work and to some degree their results will offer evidence in support or against this theory. In a general sense the EMH states that the markets are informationally efficient and that current market prices fully reflect all available information about the underlying asset. Models of this type can be simplified to expected return theories where the following equation describes the efficient market:

$$E(\tilde{p}_{j,t+1} \mid \phi_t) = [1 + E(\tilde{r}_{j,t+1} \mid \phi_t)]p_{jt} \tag{2.31}$$

where E is the expected value operator, p_{jt} is the price of asset j at time t, $r_{j,t+1}$ is the one period percentage return, ϕ_t is the information set and the $\sim$ signifies that the variables are random. Examining equation 2.31 we observe that it defines a relationship where the expected price of a stock at time $t + 1$ given a set of information at time t is equal to 1 plus the return of the stock at time $t + 1$ multiplied by the price of the stock at time t. Since the price and return variables are random this implies that the future value of any stock given the set of available information is not predictable. This expression is the basis for many modern financial models and, if it is valid, also calls into question financial forecasting models and the need for *active portfolio management*. Active portfolio management is an investment strategy where a collection of assets are regularly adjusted to take into account current trends. This relationship also implies that all expected excess returns for a portfolio of assets in relation to the *market portfolio* are equal to zero. The market portfolio is a representation of all assets that trade on a given market, for example, in the United Kingdom the market portfolio is the FTSE 100 index.

Even the casual observer of the stock market will know that in the long-run markets drift and tend to drift upwards. To demonstrate this fact, figure 2.5 is a time series plot of six market indexes from around the world spanning a time period of over 20 years. It is clear from all of these markets that they are drifting, with the majority, towards higher prices. We represent this mathematically as a first order autoregressive (AR) process with a drift or intercept term, as in equation 2.32:

$$y_t = a_0 + a_1 y_{t-1} + \epsilon_t \tag{2.32}$$

where y_t is the observation of the series at time t, a_0 is an intercept or drift term, a_1 is a constant, and ϵ_t is white noise. Based on this formulation the drift term a_0 will be ≥ 0 and therefore the long-run behaviour of the process will be a positive drift. If $a_1 = 1$ then the equation simplifies to the well known stochastic process the random walk. Considering this evidence Fama further refines the EMH into two special cases: (i) a sub-martingale model and (ii) the random walk. A martingale process is equivalent to a gambler flipping a fair-coin where the probability of winning and losing are equal; this would be a "fair game" situation. However, when the coin is weighted in favour of one side we have a sub or super martingale process depending on who receives the advantage. If the gambler was to receive a higher expected return than we have a sub-martingale process, formally defined as:

$$E(\tilde{p}_{j,t+1} \mid \phi_t) \geq p_{jt} \;\; or \;\; E(\tilde{r}_{j,t+1} \mid \phi_t) \geq 0 \tag{2.33}$$

The relationship defined by either form of equation 2.33 would be favourable to investors as one would always be guaranteed to retain at least your initial investment. However, the EMH is still valid as the expected returns are not in excess to the market portfolio and are still random variables.

The other special case that the market follows a random walk is the focus of several studies in the literature and often these separate theories (efficient markets and random walks) are used interchangeably. Despite sometimes being confused with one another, informationally efficient markets and markets following a random walk are not equal. Under a random walk there are two assumptions which must hold; the first that successive price changes are independent and the second that the returns are also identically distributed, which means random and *iid*. The model expression is:

$$f(r_{j,t+1} \mid \phi_t) = f(r_{j,t+1}) \tag{2.34}$$

this relationship states that the marginal and conditional probability distributions of an independent random variable are identical. Studying the notation we see that equation 2.34 is stricter than what we have in equation 2.31 which was only concerned with expected returns from a stochastic process. The random walk model requires that the entire distribution is independent at ϕ_t. In Lo et al. [95] the authors reject the random walk

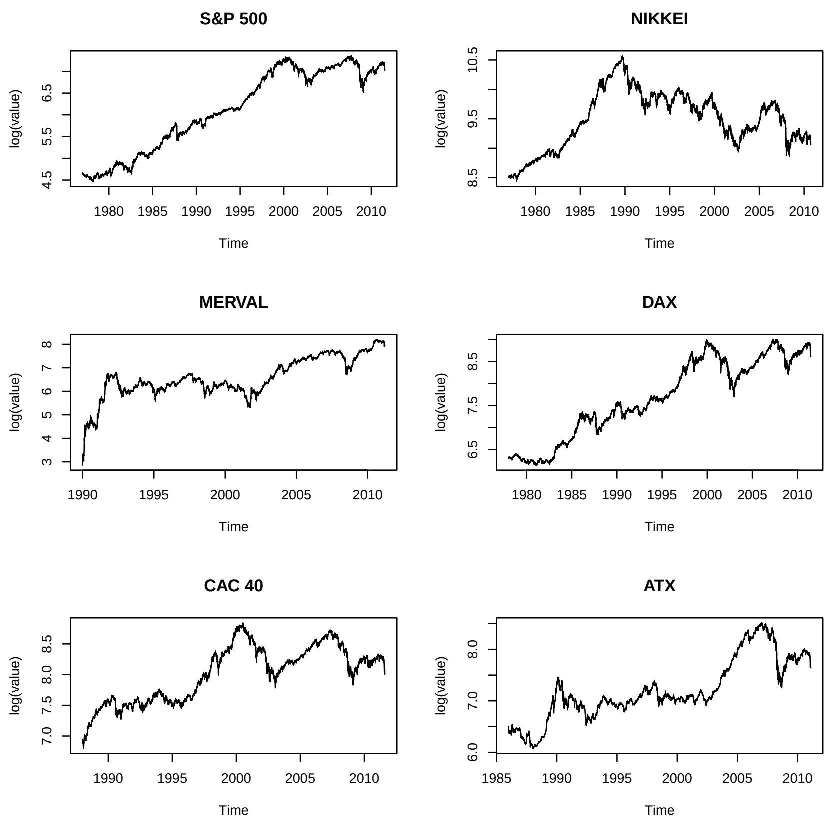

Figure 2.5: Time series plots of 6 market indexes on the logarithmic scale.

hypothesis, but conclude that their results do not necessarily imply that the stock market is not "information efficient". Fama points out that the random walk hypothesis initially came about mainly from observation of market movements [69] and [138] and provided little in the way of economic rationale.

Further to these special cases, there are three forms of the EMH based on the coverage of the information set ϕ: weak, semi-strong and strong. The distinction was created to test different levels of market efficiency and more accurately determine where the hypothesis breaks down. Under the weak form, ϕ_t only contains historical prices and states that by considering only these prices a trading model cannot produce sustainable excess return to that of expected returns in the market. Under the semi-strong form, ϕ_t contains all publicly available information such as news articles and financial reports, etc. The final form is strong and is the least popular as it would be the least plausible and the most difficult to test. Under this assumption the ϕ_t also contains all "insider" information which would mean that individuals which have an intimate knowledge of the company would not have an advantage over other market participants. Given the restrictions and control over insider trading it is unlikely that such a condition would hold true and in [36] Fama does concede that there is evidence (thought not a substantial amount) contradicting this form.

2.2.2 Behavioural Finance

The term behavioural finance (BF) defines a field of study that makes allowances for market inefficiencies and attempts to understand market behaviour from a psychological perspective. The first occurrence of this idea pre-dates that of the EMH by several decades, where in 1912 Selden [121] wrote *The Psychology of the Stock Market*, where he argues that the changes in market prices are a result of mental attitudes and biases of market participants and not a reflection of the underlying value of some asset. Several studies have appeared in the financial literature discussing this field, however for brevity, only a few of the more influential papers will be discussed.

In Kahneman et al. [63] the authors present a critique of utility theory (a theory essential for informationally efficient markets) and offer an alternative description of how investors make decisions. They argue that "prospect theory" offers a more accurate account of investor behaviour, where people value gains and losses relative to a reference point and that they are risk-adverse with the possibility of a certain gain and risk-seeking when faced with a possible loss. Prospect theory is able to explain a well documented behavioural bias called the disposition effect, which is an individuals tendency to sell when in a positive return investment and hold on to an investment that is in a negative position. Thus, the utility functions for the same investor depend on the state of the investment itself, as depicted in figure 2.6. Weber et al. [134] confirmed that the disposition effect, which is sometimes referred to as the reference point effect, does exist. The authors also discovered that the investment behaviour of their subjects was inconsistent with portfolio theory, as proposed in Markowitz [98]. The results showed that the subjects tended to have a more diversified portfolio and traded more often than what would be optimal. The authors also found that the subjects anticipated mean-reversion in the market where they expected losers to become winners and vice-versa. This behavioural trait of being loss-averse is also observed in Coval et al. [27] where day traders on the Chicago Board of Trade regularly assumed above-average afternoon risk to recover morning losses, which had an effect on short-term prices.

Another well known market phenomenon that has received attention from BF is the apparent over and under reaction of investors to new information which is contrary to how efficient markets would operate. A model of investor sentiment is offered in [4] as explanation for this behaviour. In underreaction the evidence shows that over horizons of less than a year stock prices underreact to news, which would mean that current good news would have predictive power of positive stock returns in the future. For overreaction the evidence shows that in time-horizons of 3-5 years, stock prices will

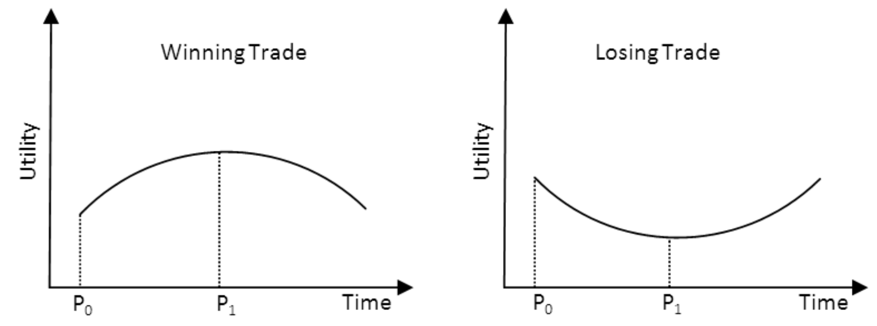

Figure 2.6: The utility functions for individual investors which show the disposition effect in the stock market. Where investors are more likely to sell winning trades and hold-on to losing ones. P_0 is the initial buying into the market and P_1 is decision point to sell or hold onto the investment. When in a winning trade (left) the expected utility goes down over time so an investor is more likely to sell. When in a losing trade (right) the utility curve is convex and the expected utility increase with time and therefore the investor is more likely to hold.

overact (become overvalued/undervalued) to a series of good/bad news and subsequently the returns will be lower/higher over the coming time horizon. The authors draw on two psychological traits known as representativeness and conservatism to explain this behaviour. Representativeness is the tendency for people to see a situation as typical or representative of some group of behaviour and ignore the laws of probability in the process. In the market this would lead to overreaction for a company that has had considerable growth over the last few years. It would become overvalued as investors are too optimistic about its future growth even though it is more likely to slow down rather than keep expanding forever. If we were to speak of this behaviour in terms of expected returns we have:

$$E(r_{t+1} \mid z_t = G, z_{t-1} = G, ..., z_{t-j} = G) < E(r_{t+1} \mid z_t = B, z_{t-1} = B, ..., z_{t-j} = B) \quad (2.35)$$

Where z_t is a form of new information at time t, G and B designate if it is good or bad respectively and $j = 1$ and probably much greater than 1. Equation 2.35 states a relationship where the expected return from a stock which has had a series of good news is lower than for a stock with a series of bad news. The second psychological trait is conservatism, which is the tendency of people to be slow in updating their models in the face of new evidence. This leads to the underreaction to news and the slow integration of it into stock prices, once again stated in terms of expected returns we have:

$$E(r_{t+1} \mid z_t = G) > E(r_{t+1} \mid z_t = B) \quad (2.36)$$

The implications of equations 2.35 and 2.36 are a clear violation of the EMH where a sophisticated investor would be able to create sustainable excess returns in the market by exploiting these investor tendencies.

2.2.3 Adaptive Market Hypothesis

The adaptive market hypothesis (AMH) is an attempt by Lo [94] [93] to combine several competing theories of market behaviour, such as the distinct views presented in the previous sections. The AMH takes an evolutionary perspective of the financial markets, where the market is seen as an evolving entity. One of the greatest insights from this work is that market efficiency is not a static characteristic of the market but rather variable and cyclical. Variable efficiency, that is also cyclical, allows for both the EMH and BF to coexist. There will be times when markets are perfectly efficient and excess returns above the market portfolio are equal to 0 and then there will also be periods of inefficiency where trading models and active portfolio management will be rewarded with higher *risk-adjusted returns* . Risk-adjusted returns and returns from an asset or collection of assets that are adjusted for the amount of risk required to achieve those returns. Lo makes the claim that many of the behavioural traits discussed in BF (overreaction, underreaction, loss aversion) could be explained by an evolutionary model of individuals adapting to a changing environment. The primary components of the AMH are:

1. Individuals act on their own self interest.
2. Individuals make mistakes.
3. Individuals learn and adapt.
4. Competition drives adaptation and innovation.
5. Natural selection shapes market ecology.
6. Evolution determines market dynamics.

The first item on the list is the same as a core assumption of the EMH and is central to modern economics. It was first discussed by Adam Smith in his book *The Wealth of Nations* [125] as the underlying driver of capital markets. However, the rest of the items are unique to the AMH and are based on work from behavioural finance and evolutionary biology. The AMH has received a considerable amount of attention from the academic community since it was first proposed and in chapter 3 a detailed description on the related work on the validity of the AMH is presented. For now, the implications of the AMH fundamental to the validity of modelling financial time series with computational intelligence or any technique not based on random walks are highlighted. The three implications are:

1. Variable market efficiency,
2. The waxing and waning of investment strategies, and
3. A time-varying and path dependent equity risk premium

These implications change the interpretation of the information set ϕ and allow for tactical asset allocation, dynamic risk assessment and active trading strategies. Tactical asset allocation is another term for active portfolio management where the collection or portfolio of assets are regularly adjusted.

2.3 Chapter Summary

In this chapter we presented an overview of methods and hypotheses utilized and discussed in the thesis. We started with an overview of several algorithms and machine learning paradigms from the literature, with particular focus on the specific algorithms implemented for the thesis. We continued this chapter by describing three major stock market hypotheses from the financial literature and how they differnetiated themselves from another. This included a description of the AMH which is referenced to throughout the thesis.

Chapter 3

Related Work

This chapter provides a detailed description of the related work that is vital to the contributions made in the thesis. As such this chapter is divided up in to sections that reflect the three parts of the thesis: validity, implications and innovations. The validity section discusses the related work from econometrics that initially lead to the determination that the AMH would be used as a reasonable characterization of the financial market. The implications section discusses related work on computational intelligence and financial analysis that concerns the financial time-series characteristics under study. Finally the innovations section also covers the overlap of computational intelligence and financial analysis but related to technical analysis and forecasting, as well as, prior attempts to create a discretization algorithm specific to financial time series.

3.1 Validity

There exists an entire spectrum of views on what governs the behaviour of financial markets, as was discussed in 2.2. Despite its popularity in academia the EMH is not widely held in the financial industry [48] and it seems illogical that all the time and effort spent by financial institutions in modelling financial markets would be an exercise in futility. In fact many academics believe that markets need to be inefficient for them to be liquid and therefore to exist [45] [46]. From this point there has been a growing body of literature that suggests the rigid and static nature of efficient markets is not valid and that an alternative based on a more dynamic model of market characteristics, i.e. the AMH is a more reasonable alternative. This includes the work by Neely et al. [105] that showed technical trading rules used in the foreign exchange markets were able to produce sustainable excess returns. From a computational intelligence perspective the most important aspect of a hypothesis is not necessarily that it can be proven beyond a doubt but whether or not it is offers any insight for modelling and forecasting with heuristic algorithms. Thus what follows in this chapter is a discussion of the literature pertaining to the AMH that provides quantifiable characteristics of the market.

3.1.1 Variable Stationarity

There is no previous literature that we are aware of that exactly discusses the possibility that a financial time series could be considered stationary during some periods and non-stationary during others. However the idea that, what we labelled *variable stationarity*, could exist is based on the results of two studies. The first by Lo et al. [95], where he rejects the hypothesis that financial markets follow a random walk based on variance ratio tests that were proposed in the same paper. An interesting implication of these results is that if the markets do not follow a random walk then there are several (not necessarily mutually exclusive) reasons for this conclusion. Once such possibility is that the series does not contain a unit-root which is a necessary but not sufficient condition for a random walk. A time series with a unit-root is non-stationary which has serious implications for modelling said time series. In the second study by Diebold and Kilian [29] the authors investigate if unit-root tests are useful for selecting forecasting models (a unit-root test is a statistical test for stationarity explained in section 4.3.2). Though [29] does not specifically state that stationarity can be variable their results demonstrate that choosing a forecasting model based on the outcome of a unit-root test will improve the forecasting accuracy of an autoregressive (AR) model. Thus, a unit-root test is useful for selecting a forecasting model and because the experiments are performed on a single time-series then this implies that the characteristic of stationarity is time varying. If it was static then pre-testing a time series with a unit root test would not offer any advantages.

An initial experiment to determine if there are periodic departures from a random walk was conducted using the variance ratio test (see Appendix A) applied iteratively using a sliding window. The experiment was motivated by the results from [95] where the random walk hypothesis could not be rejected for all sub periods of the US markets. The results from this study for the US market index are shown in figure 3.1, where z and $z*$ are the test statistics for the variance ratios that assume the variance is homoskedastic and heteroskedastic respectively. Given that market variance is proven to be heteroskedastic [30] the $z*$ statistic is considered more robust. The increasing values of q indicate the number of lagged values of the series used for the test. The results demonstrate that over a 10 year period from 2000-2009 there are several and prolonged departures from a random walk which implies that this characteristic is also episodic. Whether a unit root is present during these departures will be explored in chapter 4.3.

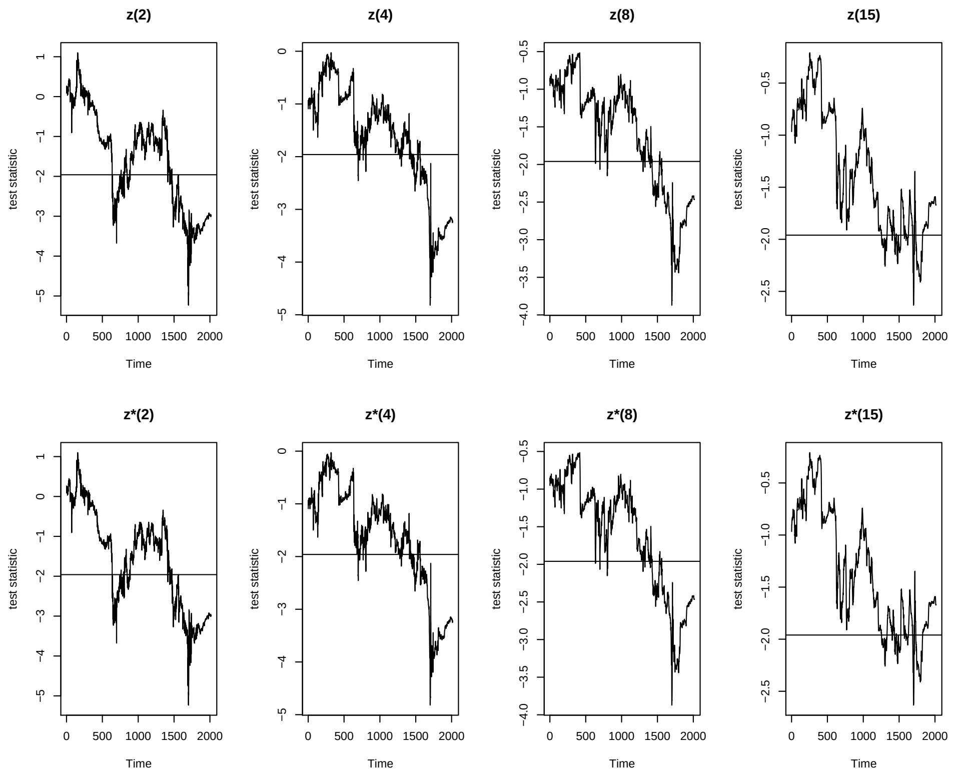

Figure 3.1: A plot of the test statistics from the z and z variance ratios. The solid line represents the significance level of -1.96.*

3.1.2 Variable Efficiency

The AMH was first published in 2004 and since then, there have been several studies which examined variable efficiency using a variety of tests. There is of course related literature submitted/published prior to the AMH that does support it but obviously does not make reference to it. This includes work by Cajueiro et al. [15] (which is the first of a series of papers published on the subject by the authors) which examines the long range dependence in Asian equity markets. Their test was based on the Hurst exponent calculated using the original R/S statistic as proposed in [55]. R/S analysis is a popular technique for detecting long range dependence in a time series and can detect episodic or cyclical dependencies by applying it iteratively with a sliding window. The findings of the study were that all of the Asian markets considered (China, Hong Kong and Singapore) displayed long range dependence and that based on the sliding window results these dependencies were episodic. The authors do not mention the AMH in their conclusions but their findings which show statistically significant dependencies in financial time series do support the AMH implication of variable efficiency. A caveat to this work is that the classical R/S statistic was used and not the modified R/S version as proposed by Lo et al. [92] that is more robust. Therefore it is possible that some of the reported findings are false positives and could impact the final results. In the same year the authors also published a study [16] that considered more financial markets from around the world that also examined long range dependence. An improvement for this work is that it used the modified version of the R/S analysis that took into account the effect of short range dependencies. Using the sliding window approach the findings of the study are consistent with their previous work in that the majority of the markets displayed episodic long range dependence. A third instalment by these authors was made in 2005 [17], where once again the modified R/S analysis was applied to several financial markets using a sliding window. The main distinction for this work is that the authors were examining squared and absolute returns as a means to focus on volatility; the findings are consistent with the previous two studies.

The next study by Lim [85] in 2007 uses a different approach to measure market inefficiency. The author used a test proposed in Hinich [49] and Hinich and Patterson [50] called the Portmanteau bicorrelation test statistic or the H-statistic. The details of the calculation will be supplied in chapter 4 but essentially the test is for non-linear dependence in a time series. The reason the H-statistic was chosen is because the existence of non-linear dependence in a financial time series would invalidate the EMH. Lim's experiment framework was based on [15] and therefore applied the H-statistic iteratively using a sliding window. The results from the experiments revealed that non-linear dependence existed in all the financial markets considered and that is was episodic. The results also suggested that emerging markets were less efficient than developed but that both exhibited a cyclical nature to market inefficiency rather than a convergence.

The strategy in Ito et al. [57] to measure time varying market efficiency was based on the estimated autocorrelation of stock returns. The authors avoided using a sliding window by fitting an autoregressive (AR) model to the returns data using the Kalman filter (KF) (as described in section 2.1.12). The benefit of using a state space model with the KF is that the sampling bias introduced by using a sliding window of a static size is removed and the model can be continuously updated as new data becomes available. The results of the study, which examined the US stock market, found that time-varying market inefficiency exists and that the most inefficient period was in the 1980's.

The final two studies reported in this section were written by the same set of authors in 2009 and 2010. The first ([146]) examined market inefficiency based on forbidden patterns and permutation entropy. Their results showed that inefficiencies exist in markets around the world and that market efficiency is also time varying. The second study ([145]) was based on the complexity-entropy causality plane, a tool for discriminating Gaussian from non-Gaussian process and different degrees of correlations. In accordance with their previous studies the results demonstrate that the markets under study can be characterized by time varying market inefficiencies.

3.1.3 Time Varying Risk

Another important implication of the AMH is that it allows for a time varying risk to reward relationship between a financial asset (such as a stock or commodity) and the market as a whole. This allowance is supported by a growing number of studies ([101] [141] [139] [26] [44] [33] [117] [9] [120]) that have demonstrated that the risk of an asset in relation to the market is not static. Formally we can describe this relationship by the capital asset pricing model (CAPM) [122] [89], equation 3.1:

$$E(R_{i,t} - R_f) = \beta_i E(R_{m,t} - R_f), \tag{3.1}$$

R_f is the (fixed) return of a risk free asset, $R_{i,t}$ and $R_{m,t}$ are the returns of asset i and the market portfolio respectively. β_i is the measure of systematic risk or the proportion of the assets expected excess return described by the market. The linear regression model:

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i(R_{m,t} - R_{f,t}) + \epsilon, \quad t = 1, ..., T \tag{3.2}$$

where ϵ is assumed to be iid zero mean and constant variance disturbances, which allows the estimation of β_a as the least squares estimate of β (the ratio defined in equation 3.3):

$$\beta_i = \frac{cov(R_i, R_m)}{var(R_m)} \tag{3.3}$$

where a $\beta = 1$ would have the same risk as the market and a value less than 1 would indicate a less risky asset. Under an efficient market α would be equal to 0 and β would be constant. The relationship (3.1) and the estimating method that uses (3.2) are valid under the crucial assumption of an efficient market.

Well before the incarnation of the AMH it was recognized that the CAPM in its canonical form was not adequately explaining the observation data. Work as early as 1975 by

Blume [8] and 1976 by Black and Fischer [7] highlighted the presence of heteroskedasity is portfolio volatilities and that changes in volatilities in relation to the market portfolio were not homogenous. These observations were indicative of a time varying β and therefore an asset may become more or less risky as the economic environment changes. This idea was followed up some years later by Bollerslev et al. [9] and in Schwert et al. [120] using regression models to estimate time varying monthly variances in stock portfolios. Both studies confirmed the results of Black and Fischer that betas are time varying based on sample sizes of returns as large as 50 years. Since 1990 there have been several studies which confirmed that the β value of a particular asset can be time varying, however the short coming of these studies is the approach. The dynamic behaviour of a time varying β cannot be described by a linear regression model. To accommodate this dynamic behaviour state-space representations of the CAPM have been proposed [139] [26] that allow the parameters to be stochastic and therefore if β is time-varying its trajectory through time can be estimated without the unnecessary constraints of a linear model. Though several techniques for estimating a time varying β exist the results from Brooke et al. [10] and Choudhry et al. [26] demonstrated that the preferred technique was the Kalman filter (section 2.1.12). The exact methodology for fitting 3.1 with a SSM and the KF will be described in chapter 4 but this methodology has been used in [26] [101] [139] and [44] all of which demonstrated that β was time varying.

3.2 Implications

The adaptive market hypothesis contains several implications that will influence financial modelling decisions. The thesis is concerned with two implications in particular and their impact on modelling and forecasting with computational intelligence algorithms. The first implication can be generalized as variable characteristics of the market, specifically stationarity and efficiency. The second implication is the waxing and waning of investment models and how their effectiveness is dependent on the current market dynamics.

3.2.1 Variable Stationarity

Whether or not a time-series is stationary or non-stationary has consequences for modelling said time-series for many statistical techniques. Financial time series is considered non-stationary [75] [106] and this characteristic is also considered to be static. In chapter 4 this generally held belief is challenged and experimental results demonstrate that at certain time periods markets would be more appropriately described as trend-stationary and thus stationarity could be considered time varying. Given that this characteristic was historically considered static there is no directly related literature on the effect of “variable stationarity” (VS) on computational intelligence.

Although VS has not been previously explored in the computational intelligence literature there is a study which focuses on how to model non-stationary data with an artificial neural network (ANN). Due to the complications of non-stationary data (explained in chapter 4) it can negatively impact the robustness of models learned by ANNs. In Kim et al [72] the authors investigate the preferred procedure for training ANNs for non-stationary time series where the KOPSI market index from Japan serves as the system of interest. Unfortunately the authors ignore the statistical properties of the data and the wealth of literature from the forecasting domain and make the assertion that the preferred way to deal with non-stationary data is to overfit the models. This is achieved by removing a validation set and simply training on all the data. The results analysis considered the residual squared error and if the residuals contained autocorrelation. This is an incomplete analysis of the results and the size of the forecasting errors should have been compared to standard preprocessing techniques. Overfitting is a major obstacle for any modelling technique and the conclusions of [72] are unsound. Thus as part of the chapter on the implications of VS the experiments will also refute this work and offer a statistically sound procedure for modelling non-stationary data with ANNs.

3.2.2 Variable Efficiency

The previous studies on variable efficiency (VE) from section 3.1.2 were concerned with indentifying how often a financial market is inefficient and then ranking the markets in terms of efficiency. None of these studies considered if the periods of market inefficiencies offered any forecasting advantages. Two studies by Todea et al. [130] [129] examined the profitability of moving average strategies during periods of market inefficiencies across several markets in Asia and Europe. A moving average (MA) trading strategy generates signals through the interaction of two or more moving averages and can be mathematically represented by equations 3.4 and 3.5:

$$Buy: \quad MA\_st_{t-1} < MA\_lg_{t-1} \ \& \ MA\_st_t > MA\_lg_t \tag{3.4}$$

$$Sell: \quad MA\_st_{t-1} > MA\_lg_{t-1} \ \& \ MA\_st_t < MA\_lg_t \tag{3.5}$$

where MA_{st} is a short period moving average and MA_{lg} is a long period moving average. From equations 3.4–3.5 we see that when a shorter MA (for example a 5 day MA) crosses a longer MA (for example 100 day MA) from below it generates a buying signal and when it crosses from above we have a sell signal. Figure 3.2 displays a moving average trading strategy and the signals it produced for the Merval index from Argentina. Periods of market inefficiencies in [130] and [129] were identified using the

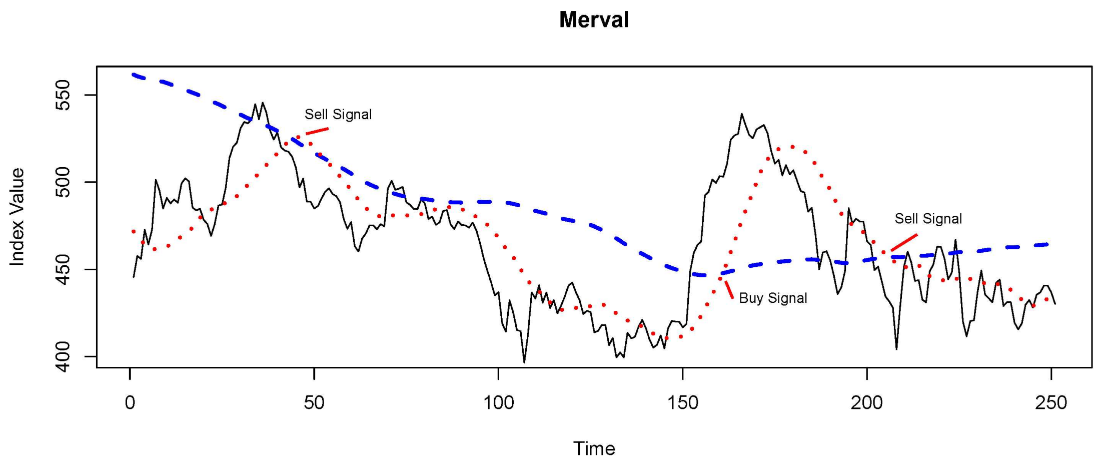

Figure 3.2: A depiction of 250 days of trading for the Merval index with a 100-day (blue dashes) and 20-day (red dots) moving averages. A buy signal is created when the 20-day MA crosses the 100-day MA from below and a sell signal is generated when it crosses from above.

H-statistic (see chapter 4) and its linear companion the C-statistic (see [130]). For each market an optimal moving average strategy was chosen from 15000 alternatives based on profitability and then the average returns for periods of efficiency and inefficiencies calculated and compared. The results of both studies suggested that the excess returns in relation to the market of the moving average strategies were higher during periods of inefficiencies and thus were in accordance with the AMH. There are however a few shortcomings of the studies in terms of the experiment methodology and the results analysis. The experiment methodology only considers an optimal trading strategy and therefore the results are very specific to this one strategy, given that over 15000 were evaluated it would have been more meaningful to report the distribution results where we could have seen on average if any given strategy was more likely to be profitable. This would be preferred as the “optimal” strategy is not likely to be known a priori and therefore most practitioners would be using sub-optimal strategies. Additionally the results analysis does not consider risk, so it is very likely that during periods of inefficiencies there is more variance in the daily returns of the market and therefore more risk. Thus the returns of the trading strategies during these time periods would be expected to be higher. Without taking into account risk and presenting risk-adjusted returns it is difficult to quantify how much of an advantage was offered by these market inefficiencies. To avoid having to account for risk altogether the authors could have reported accuracy or the percentage of trades that were profitable rather than profit itself. Since accuracy and profitability are strongly correlated then an expected increase in accuracy implies a high probability that returns will also increase.

The related literature on the effects of VE on use of computational intelligence is not based on the various statistical tests previously discussed but comes from research into predictability filters for genetic programming (GP). GP [74] is an evolutionary algorithm that learns a computer program which satisfies some objective based on a set of training data. GP has been extensively used in financial modelling ([22] [105] [56] [112] [62]) and offers the advantage of being a "white box" technique where the evolved program can be viewed and is potentially human readable. The first occurrence of predictability filters was in Kaboudan [61] where the η measure is developed which is a predictability score ranging between 0 (completely unpredictable) to 100 (completely predictable). The measure is a proportion of sums of squared errors (SSE) between a GP run on a normal time series (SSE_y) and a GP run on a shuffled version (SSE_s) of the same time series. The idea behind the measure is that if there is a pattern in the time series than the errors from the original should be lower than the (assumed to be random) shuffled version. The η measure calculation is as follows, equation 3.6:

$$\eta = \begin{cases} 0 & \text{if} \left(\frac{\overline{SSE_y}}{\overline{SSE_s}}\right) > 1 \\ 100 * \left(1 - \left(\frac{\overline{SSE_y}}{\overline{SSE_s}}, t\right)\right) & \text{otherwise} \end{cases} \tag{3.6}$$

where

$$\overline{SSE_y} = \frac{SSE_y}{k} \ and \ \overline{SSE_s} = \frac{SSE_s}{k} \tag{3.7}$$

and k is the number of GP trials sampled. The results from [61] demonstrated that by using the η measure as a filter to decide when to make predictions the overall prediction accuracy of the GP models were improved. Other work which used the η measure include Chen et al. [24] where the authors extended the measure to include a random search of equal intensity to compare against the GP models. This added layer was intended to distinguish between time periods when there were no interesting patterns to model and when the η measure said the series was predictable but GP was unable to capitalize. The results from applying the pre-tests to nine financial markets provided preliminary evidence that the filters were able to improve the forecasting performance of the GP models and thus the filters were effective at detecting patterns in the time series. More recent work concerning filter approaches is Wilson et al [135] that developed two novel pre-tests for

GP. The first filter was based on volatility called “Price Frequency” and second based on information theory called “Information Theoretic”. The price frequency filter was used to detect higher then desirable volatility in a time series that would negatively impact the forecast performance of GP. The information theoretic filter was intended to distinguish between time periods of random behaviour and periods where an pattern exists. The experiment results suggest that the price frequency filter was more effective then the information theoretic where the performance of GP was significantly improved when the price frequency filter was pretesting the time series.

The research into predictability filters has demonstrated their effectiveness. However, none have reported the percentage of periods that were considered to be predictable but the results for each demonstrate that periodically patterns exist in financial time series. A discernable pattern that continues past the time period where it was detected is evidence of a market inefficiency and thus these results not only demonstrate variable efficiency but also that when inefficiencies are detected it is reasonable to assume an increase in forecasting performance of GP and most likely computational intelligence algorithms in general, as stated in [24].

3.2.3 Cyclical Effectiveness

Previous work related to the investigation of the waxing and waning of investment strategies comes for research into the Dinosaur (DH) [3] and Market Fraction (MFH) Hypotheses [65] [23]. The DH could be summarized as the need to continually evolve new strategies as the market changes and any previous strategy will become obsolete after the time it was created. This is in contrast to the AMH which states that market behaviour is more cyclical and that previous strategies will become useful again. The initial published results from testing the DH in [66] conclude the DH is not valid and that “dinosaurs” can return and hence the market behaviour is cyclical. These results reflect an experiment methodology based on the average fitness of a population of GP strategies being tested on out of sample data. The fitness function is based on accuracy, which is strongly correlated with profitability. However, subsequent work by the same authors [67] comes to the opposite conclusion and now their results suggest that the DH is valid and that previously effective trading models become “dinosaurs” i.e. become extinct or obsolete as time progresses. A major shortcoming of this work is that the profitability or fitness of these trading strategies is not used as the determining factor as to whether or not the strategy is obsolete. Rather the authors have opted for a new metric based on a dissatisfaction rates derived from clustering trading strategies using self organizing maps (SOM). This approach is based on the belief that the structure of a GP evolved decision tree is correlated with the effectiveness of this trading strategy. Thus when the structures of different populations of GP models are compared the conclusion that DH is valid is based on the result that the structures on average are never the same. Whether or not these models are profitable or outperform the market again is never considered and therefore the results cannot be considered contrary to the AMH. Another short coming of the approach is that the GP models are not controlled for over-fitting and their complexity is not constrained. As a result the GP trading strategies are most likely modelling noise and whether or not the signals or patterns in the market are repeating will be difficult to determine. These results are also considered by the authors of [67] to be supportive of the MFH which states that the micro-structure of the market (the composite of trading strategies in the market) continually evolves and that trading models also have to continually evolve to be useful. This however is not direct evidence as to whether or not the effectiveness of a trading strategy is cyclical.

The previous work on the subject has missed the main point that Lo is making when he discusses the waxing and waning of the effectiveness of trading strategies. The most important metric for a trading model is how it performs in real currency (i.e. \$ or £) terms. Without an investigation into the profitability or accuracy of investment models on out of sample data, the effect of the cyclical nature of market behaviour cannot be determined. In what follows in chapter 5 is investigation into the cyclical effectiveness of computational intelligence models in relation to a relevant benchmark.

3.3 Innovations

This section begins with a description of the related work pertaining to computational intelligence approaches to technical analysis; this includes optimization of technical indicators as well as meta learning approaches. This section then goes on to discuss the literature on discretizing a time-series and previous attempts to create a discretization algorithm specific to financial time series.

3.3.1 Improving Technical Analysis

There have been several attempts to improve technical indicators using population based optimization from computational intelligence. This includes work using genetic algorithms (GA) (see chapter 2.1.2) to fit a technical indicator's parameters to a set of training data. The majority of the attention has been given to optimizing MA strategies [39] [38] [109] [76] (as discussed in section 3.2.2) but other indicators such as filter rules [86] and Bollinger bands [13] (see chapter 5) are also represented. In Fernandez-Rodriguez et al. [39] [38] the authors were optimizing a general moving average (GMA) rule with a genetic algorithm (GA). A GMA can be stated as equation 3.8:

$$S(\Theta)_t = MA(\theta_1)_t - (1 + (1 - 2S_{t-1})\theta_3)MA(\theta_2)_t \tag{3.8}$$

where $\Theta = [\theta_1, \theta_2, \theta_3]$ denotes the parameters associated with the GMA rule and MA(θ) is a MA indicator. θ_1 is the short-period MA, θ_2 is the long-period MA and θ_3 is a filter parameter included to reduce the number of false buy/sell signals. The result of equation 3.8 is either positive or negative which translates into a buy or sell signal respectively. The authors evaluated this equation for each trading day to decide if the model would buy into the market on that day or invest in a risk-free rate (R_f). The GA was optimizing the three parameters using the following fitness function, equation 3.9:

$$r_{tr} = \sum_{t=1}^{N} S_{t-1} rm_t + \sum_{t=1}^{N} (1 - S_{t-1})R_f - T \times c \tag{3.9}$$

where T is the number of transactions, c is the transaction cost per trade, and S_{t-1} is either {0,1}. In these experiments the transaction cost is kept static which is different from [13] where the transaction costs were a function of the amount invested. To assist in evaluating the performance of the GA evolved solutions a benchmark portfolio was created which was a risk-adjusted buy and hold approach, equation 3.10.

$$r_{bh} = \alpha \sum_{t=1}^{N} +(1 - \alpha) \sum_{t=1}^{N} rm_t - 2c \tag{3.10}$$

Where α is the proportion of trading days the model is invested in a risk free rate and rm_t is the market return at time t. The results show that the models developed from the GAs were able to outperform the risk-adjusted buy and hold strategy, however the parameter α is never revealed so the proportion of days spent at risk in the market it not known. If α is quite large it is not surprising that the GA models performed better. It would have been useful to see the GA evolved model compared to the canonical setting of a MA to evaluate if the extra effort to evolve the MA indicator was justified. The other work that also considered using GAs for optimizing the MA technical indicator [76] [109] had similar experiment frameworks and results. A strength of this approach, where a technical indicator is optimized in its canonical form, is that the traditional interpretation of the indicator is still valid and therefore the signal is interpretable by a human. However, these studies along with the others based on different indicators [13] [86] all share the same shortcoming that only one rule is being used in the out of sample testing data, a common approach to technical analysis is to have *confirmation*, which is when two or more indicators produce the same signal and in theory produce more profitable trading solutions [78] [77]. The desire to combine signals leads to a meta-learning approach where the computational intelligence models are taking into account multiple indicators or combining signals from a population of agents based on technical indicators.

Related work pertaining to combined signal analysis in computational intelligence comes from several areas including reinforcement learning (RL), biologically inspired optimization, genetic programming (GP) and learning classifier systems (LCS). The related work could also be segmented into single agent and multi-agent approaches. Under single agent approaches the technical indicators serve as inputs to the algorithm [127] [97] that is generally learning in a supervised context for classification of stock movements or investment decisions. One of the most well known examples of this approach is EDDIE (Evolutionary Dynamic Data Investment Evaluator) [83] and its subsequent extensions [68] which is a tree-based GP algorithm for classification that evolves decision trees based on technical indicators. The original version of EDDIE has static window sizes for the technical indicator parameters but in the latest version these window sizes can also be determined during training. EDDIE has been extensively tested and shown to produce trading strategies which can outperform a passive buy and hold approach. The multi-agent approaches often involve a meta learning agent that interacts with a population of single agents that are generating trading signals. This would include [108] which is a combination of ANNs and RL and the various studies concerning learning classifier systems. As described in chapter 2.1.10 an LCS would also have a meta agent that is based on RL. The population of agents in the LCS could be any type of classification algorithm but a popular theme amongst LCS for financial forecasting is to have a population consist of technical indicator rules evolved using a GA [84] [126] [20] [137] [113] [90] [114] [132] [119] [21].

The main distinction between the algorithm developed in chapter 6 and the previous work is that in all of these approaches the technical indicators are not being used as they are traditionally intended. Most technical indicators do not make a *next-tick* prediction rather they are forecasting future trends over some time horizon. Additionally, technical indicators generate signals from the interactions of components within the indicator (figure 3.2) not by separate criteria being satisfied (such as $MA_{5-day} > MA_{10-day}$). This problem has been highlighted before in [113] but their solution was still not compliant with how a technical indicator is typically used in the real world. The proposed approach is an online adaptive algorithm that optimizes and combines a population of technical indicators whilst still maintaining their intended interpretation.

3.3.2 Discretization

In time series analysis research there is a strong interest in discrete representations of real valued data streams. The discretization process offers several desirable properties such as numerosity/dimensionality reduction, the removal of excess noise and the access to numerous algorithms that typically require discrete data. As a result there is a wealth of literature describing a variety of techniques to facilitate this transformation. Although several discretization algorithms exist the most commonly used are equal width or equal frequency histograms. The majority of studies concerned with sequence analysis of financial time series have used a form of equal width or equal frequency binning on a log normalized first differenced series [118] [140] [11] [19] [111]. The first differenced series, which would be the single period returns, are used as a means to counteract the non-stationarity of financial time series. This creates an extra pre-processing step before the symbolic mapping that is applied indiscriminately without regard to the specific properties of time series during a given period, such as trend and unit-roots. Also by first differencing, information is lost such as where the stock is trading in relation to current highs and lows over some meaning interval, such as the previous 52 weeks. Another method used in the literature (on its own or in conjunction with first differences) to counteract non-stationarity is the sliding window approach, which is also a popular technique for modelling financial data of a continuous nature [12]. In the general area of time series analysis, there have been several algorithms developed to transform the actual observations. Mapping the real valued observations facilitates the extraction of primitive shapes from the series for comparing segments within a time series or between them. These techniques preserve the temporal information as well as the more global position of a data point in relation to the series as a whole.

Two of the most well know techniques from discretizing time-series data for extracting primitive shapes are Symbolic Aggregate aproXimation (SAX) [87] [88] and clustering based methods (CBM) [28]. SAX has already been described in chapter 2.1.11 but to reiterate it is an unsupervised algorithm that is based on PAA and the normal distribution. CBMs transform a real valued time series to a discrete representation segmenting the time series using a sliding window and then clustering the set of segments. Each cluster is assigned a symbol and then each segment is then represented in the original time series by the symbol corresponding to its cluster. In Das et al. [28] this method was used for financial data and the rules were constructed in a manner similar to association rule mining. The effectiveness of the symbolic mapping was quantified by rule *informativeness* based on the J-measure for rule ranking. Aside from forecasting endeavours clustering of financial time series has been very popular in portfolio index tracking where several studies have been conducted. They include, inter alia, Fu et al. [42] which concerned self-organizing maps, Dose et al. [31] which used hierarchal clustering and Musetti [104] which compared several techniques for analysing companies from the S&P 100 index. However CBMs have not been as popular for forecasting and sequence analysis which may be a result of the finding of a study by Keogh et al. [71]. This study concluded that clustering time series subsequences lead to meaningless results based on the commonly used methodologies at that time, which includes the work of [28]. The authors demonstrated that the results obtained from the CBM were the same as if a sine wave had been the underlying series. These results most likely have impacted the research and discretization methods of the previous literature on financial analysis.

Aside from the studies using the equal width/frequency methods for discretization there has been some interest in using the more sophisticated algorithms from time series analysis, such as SAX. However financial time series data has some unique properties that will make it problematic, particularly that it is non-Gaussian and non-stationary. The authors of [88] have addressed the problem of non-Gaussian data and demonstrate that their algorithm is still complete but not as efficient under these conditions. The main problem with non-Gaussian data is that the symbolic mapping based on partitions of the standard normal distribution no longer guarantee an equal probability of each symbol occurring. The problem with non-stationary data is that primitive shapes can be difficult to extract as the characteristics of the series change, as depicted in figure 7.8. Two previous studies have acknowledged that SAX in its canonical form is not appropriate for financial data and have attempted to extend SAX to this application area. The first paper to consider this problem was by Lkhagva et al. [91] where they proposed a method called Extended SAX (ESAX) that attempted to find better matches within the financial time-series by embedding the max and min within in a time-window along with the mean into the symbolic representation. The authors state that the original SAX method will miss important patterns because it only uses the mean to represent a segment of the series, figure 3.3 illustrates the intuition behind the approach. With the ESAX the authors are able to maintain the original SAX symbolic nature and distance function that lower bounds the Euclidean distance. In effect what's happening is that for a given level of dimensionality reduction in terms of the number of partitions w the Extended SAX will be three times larger than SAX as a result of each partition having 3 symbols represent it rather than 1. The results show that for a given reference sequence ESAX matches

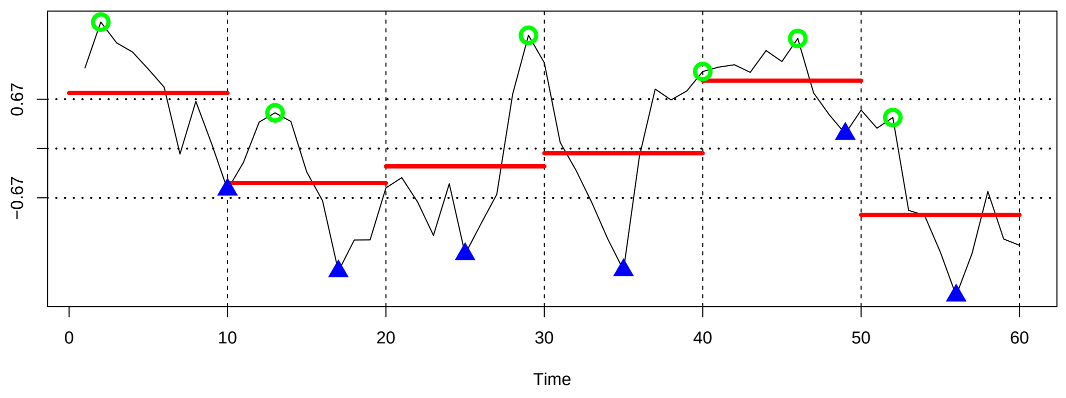

Figure 3.3: An example plot of a randomly generated time series showing the mapping of the real-valued time series to a SAX representation using a 4 letter alphabet with a PAA window size of 10. The green circles and blue triangle represent the min and max within a PAA segment respectively. The authors of ESAX highlight the need to capture this information in order to improve pattern matching using SAX. The horizontal red lines represent the means within each PAA segment.

fewer segments on average then SAX, which implies that closer matches are being found. However, the comparison is not fair because ESAX is working with more dimensions so a true comparison would be against SAX with a equal number of dimensions, because as the number of partitions increases the average matches for SAX would also decrease. Regardless of the inappropriate comparison between ESAX and SAX the proposed method does not alleviate the problems of non-stationarity and therefore primitive shapes are still being missed.

The second attempt to improve performance of SAX on financial time-series was by Hung et al. [54] where they proposed combining SAX with Piecewise Linear Approximation (PLA). PLA is also a dimensionality reduction technique where a partition of data is represented by a best-of-fit straight line that would generally be determined by a regression . In this work the authors propose using PLA as a post-processing step as a way to prune the matching results of SAX to yield superior quality matches. Using PLA to determine the final matching patterns means that the symbolic representation of SAX is changed and with it the loss of the lower-bounding distance function. The distance function used for PLA is based on slope trends and each slope trend of a segment corresponds to a symbol. The distance between symbols is then based on a tree-like hierarchy (as depicted in figure 3.4) which represents a suitable set of slope trends. The

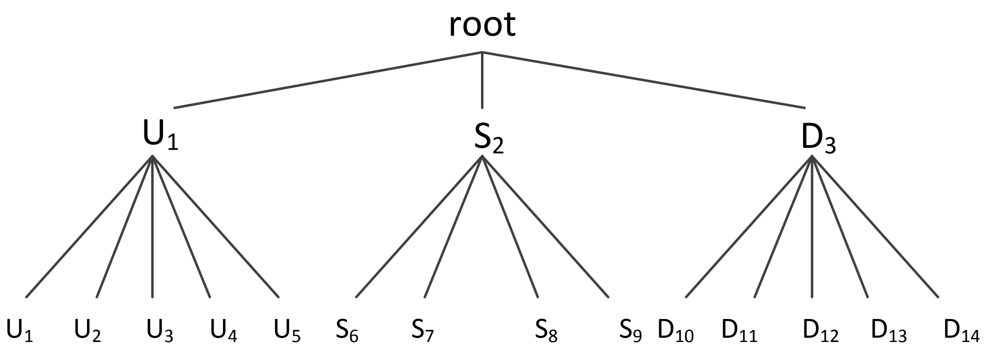

Figure 3.4: The shape definition hierarchy tree for the PLA extension of SAX.

distance between symbols (slope states x_i and y_j) would then be calculated as:

$$dist(x_i, y_j) = 2^{max(0,|k-t|-1)} * max(0, |j - i| - 1) \tag{3.11}$$

where the parent nodes of x_i, y_j are A_k and B_t respectively. The reason for using SAX initially rather than just using PLA is not discussed but the proposed method simply attempts to find closer matches based on the original matches that SAX returns. This approach is still hampered by the shortcomings of SAX with non-stationary data and therefore will still miss important patterns in different regions of the time series.

It is clear that both previous studies have missed the major problems of matching sequence patterns using SAX and that a new method has to be considered. In chapter 7, a new algorithm will be proposed called alSAX or adaptive local SAX that alleviates the problems of non-Gaussian and non-stationarity.

Part II

Thesis Contributions

Chapter 4

Validity

In the first contribution chapter the research which extends the body of work pertaining to the validity of the AMH is presented. This includes the reproduction of variable efficiency results as a means for code validation, the results from a SSM representation of the CAPM (equation 3.2), which reveals the time varying nature of the CAPM parameters and finally the test for and demonstration of variable stationarity in a financial time series.

4.1 Variable Efficiency

The presence of variable efficiency has already been established based on a review of the previous literature in section 3.1.2. In this section a portion of this work will be reproduced as a means to validate the code implementation of the portmanteau bicorrelation test statistic (H-statistic). The H-statistic, described in the next section, has been utilized in several studies (see section 3.1.2) but there is not a widely available library for calculating it. Thus for the experiments performed in chapter 5 the H-statistic has been implemented in the R statistical language. To validate the implementation, the results from running the code will be compared with the results published in Lim [85] and Todea et al. [130] for various market indexes.

4.1.1 H-Statistic

Prior to calculating the H-statistic the data undergoes two stages of pre-processing as outlined in [85]. First, the series $\{Y_t\}_1^T$ is *assumed*[1] to be a non-stationary stochastic process. To aid with the analysis, the series is transformed to stationary by converting the series as follows:

$$r_t = log(y_t/y_{t-1}) * 100 \tag{4.1}$$

where r_t is the continuously compounded percentage return for time t. The second step is to standardize the data to have a sample mean of zero and a sample standard deviation of one, as follows:

$$Z_t = \frac{r_t - m_R}{\sigma_R} \tag{4.2}$$

where $\{Z_t\}$ is the standardized series, m_R is the sample mean and σ_R is the sample standard deviation. The null hypothesis of the test is that $\{Z_t\}$ is a realization of a white noise process with null bi-correlations. The test for non-linear correlations is calculated as follows:

$$H = \sum_{s=2}^{L} \sum_{r=1}^{s-1} G^2(r, s) \tag{4.3}$$

where,

$$G(r, s) = (n - s)^{1/2} C_{RRR}(r, s) \tag{4.4}$$

and,

$$C_{RRR}(r, s) = (n - s)^{-1} \sum_{t=1}^{n-s} Z(t)Z(t + r)Z(t + s) \tag{4.5}$$

where r and s satisfy $0 < r < s < L$. The H statistic is distributed according to a χ^2 law of probability with $(L\text{-}1)(L/2)$ degrees of freedom. The number of lags, L, is specified as L = n^b, with $0 < b < 0.5$ and n is the window size. Previous work by Hinch and Patterson [50] recommend a value of 0.4 for b. In this implementation, when L = n^b does not yield an integer value, the value of L is rounded down prior to calculating the degrees of freedom for the χ^2 critical value.

In addition to the pre-processing performed above, the series $\{Z_t\}$ undergoes one additional step of pre-whitening before calculating the H-statistic. The pre-whitening step entails filtering away the linear component and therefore any autocorrelation structure of $\{Z_t\}$ by means of an autoregressive $AR(p)$ fit. When performing an $AR(p)$ fit an $AR(p)$ model is fitted to the data using ordinary least squares and the residuals are retained for further analysis. The order $p \in \{0, 10\}$ is the smallest value for which the Ljung-Box Q(10)

[1]Later in this section this assumption is tested where the results demonstrate that departures from non-stationarity exist in financial time series.

statistic is insignificant at the 10% level. In this implementation, 20 lags are used for calculating the Q-statistic and the $AR(p)$ model contains an intercept term.

4.1.2 Code Validation Results

The H-statistic has been implemented based on the details provided in [85] and [130], where the results to be reproduced were chosen based on data availability. The first set of results are published in Lim [85] and concern the US (S&P 500), Japan (Nikkei 225) and Argentinean (Merval) market indexes. The results from running the thesis implementation over the time frames for each market are presented in table 4.1. It is worth noting that the data for the thesis and the data used in the previous work are from different sources, the data in the thesis is from Yahoo! Finance and both previous studies used Datastream. From the results it is apparent that the number of samples created based on a 50-day sliding window[2] are not consistent with the reported time frame, for example, in the US market Lim had 3602 samples whereas the thesis only had 3479. This is quite a large difference that is most likely from incorrect reporting of the time-period for data collection. For each market index the time-frame was extended backwards and then forwards to accommodate the number of samples. We can see from lines 3 and 7 from table 4.1 that extending the time-frame backwards yields almost identical results. For the Argentinean market the results are comparable but the thesis implementation still labels fewer windows as having non-linear dependence.

The second set of results reproduced are from Todea et al. [130] and concern the Hong Kong (Hang Seng) and Malaysian (KLSE) market indexes. The results of running the thesis implementation using the same experiment details are displayed in table 4.2. From these results we observe that the number of samples is almost identical and therefore the time frames do not need to be adjusted. The number and percentage of rejections are also very similar with the thesis implementation again labelling fewer samples as having non-linear dependence.

[2]A sliding window approach is a method for segmenting data that yields over-lapping samples. For example, given a time series $\{Y_t\}$ and a window of size d an initial sub-sample is created consisting of observations $\{Y_{1,2..,d}\}$, the appropriate tests are run and then the window shifts by one day to cover $\{Y_{2,3..,d+1}\}$ and so forth until the end of the sample.

Table 4.1: The results comparison between the implemented H-statistic and the results reported in [85] (shown in bold) for the US (US), Japanese (JPN) and Argentinean (ARG) market indexes.

Result	Code	Index	Period	samples	# of Rej.	% of Rej.
1	**Lim**	**US**	**01/01/1992 - 31/12/2005**	**3602**	**189**	**5.25**
2	Thesis	US	01/01/1992 - 31/12/2005	3479	177	5.09
3	Thesis	US	09/07/1991 - 31/12/2005	3602	188	5.22
4	Thesis	US	01/01/1992 - 28/06/2006	3602	177	5.09
5	**Lim**	**JPN**	**01/01/1992 - 31/12/2005**	**3602**	**312**	**8.66**
6	Thesis	JPN	01/01/1992 - 31/12/2005	3396	304	8.95
7	Thesis	JPN	05/03/1991 - 31/12/2005	3602	312	8.66
8	Thesis	JPN	01/01/1992 - 30/10/2006	3602	308	8.55
9	**Lim**	**ARG**	**02/08/1993 - 31/12/2005**	**3189**	**541**	**16.96**
10	Thesis	ARG	02/08/1993 - 31/12/2005	3037	415	13.66
11	Thesis	ARG	22/12/1992 - 31/12/2005	3189	475	14.89
12	Thesis	ARG	02/08/1993 - 09/08/2006	3189	429	13.45

Table 4.2: The results comparison between the implemented H-statistic and the results reported in [130] (shown in bold) for the Hong Kong (HK) and Malaysian (MAL) market indexes.

Result	Code	Index	Period	samples	# of Rej.	% of Rej.
1	**Todea**	**HK**	**14/04/1998 - 14/04/2008**	**2279**	**362**	**15.88**
2	Thesis	HK	14/04/1998 - 14/04/2008	2270	339	14.93
3	**Todea**	**MAL**	**23/09/1997 - 14/04/2008**	**2398**	**1363**	**56.84**
4	Thesis	MAL	23/09/1997 - 14/04/2008	2401	1283	53.44

4.1.3 Code Validation Discussion

The output from the thesis implementation of the H-statistic is comparable to the previous literature and in most cases the results were identical once certain assumptions were made. There is not enough detail concerning the experiments and the H-statistic from the previous work and although discrepancies could have been caused by differences in the data sources and normal implementation bias, there are three aspects of this work that were not reported. The first concerns the calculation of the lag (L) parameter from equation 4.3 and how the values are handled when the calculation n^b does not yield an integer. Are the values rounded and, if so, is this rounding performed before or after the calculation of the degrees of freedom for the χ^2 distribution? Secondly, what type of AR(p) model is fit to the data in the pre-whitening step? Does the equation contain an intercept term? Finally the third detail not discussed is the lag parameter for the Ljung-Box Q statistic and how many lags are used for detecting autocorrelation? These details will affect the outcome of the H-statistic and are most likely a source of the small variations in the output. Nonetheless the results demonstrate that the H-statistic has been implemented correctly and that in all markets considered there exists variable efficiency.

4.2 Time Varying CAPM

The work presented in this section is taken from a collaborative paper, "Dynamic risk analysis with incomplete information in an adaptive market", written by Dr. Sonia Mazzi and Matthew Butler, under review at the Journal of Business and Economic Statistics at the time of writing the thesis [99]. The theoretical model used in the study (defined by equations 4.6-4.8) was developed and implemented by Dr. Mazzi along with preprocessing scripts. Matthew Butler was responsible for initial data preprocessing, preparing the scripts for the experiments, performing the experiments and the creation of the various plots. The dicussion and results analysis was jointly contributed.

The AMH makes allowances for a time varying risk to reward relationship between financial assets and the market. From the discussion in chapter 3.1.3 the detection and modelling of this time varying relationship was revealed through the assumption of a dynamic β. Based on this literature the AMH assumption of a time varying risk to reward relationship is valid. The distinguishing features of this work are (1) that in addition to considering β to be time varying we also analyze α and explore if it too is dynamic and how the co-evolution of α and β can affect investment decisions, (2) we fit the model using the diffuse Kalman filter (DKF) and (3) we explore if a time varying CAPM holds for metals. The modelling technique allows inference on a crucial but unobserved component, namely factors influencing risk of an asset not accounted for by the market (such as behavioral biases or other exogenous shocks). The result is an adaptive tool which could be used for active portfolio management and because the approach is applicable to a range of time horizons, the analysis is relevant for several techniques of asset allocation. However, despite the evidence for a time varying β, to achieve this objective we argue that in the short-term the CAPM holds and thus model this relationship with a first-order approximation in state space by the following:

$$
\begin{aligned}
R_{a,t} - R_{f,t} &= \alpha_t + \beta_t(R_{m,t} - R_{f,t}) + \epsilon_t, \quad t = 1, ..., n && (4.6)\\
\alpha_{t+1} &= \alpha_t + \eta_t && (4.7)\\
\beta_{t+1} &= \beta_t + \nu_t && (4.8)
\end{aligned}
$$

where equation 4.6 is the observation equation and equations 4.7 and 4.8 are the state equations. The variables $R_{a,t}$, $R_{f,t}$ and $R_{m,t}$ are returns for the an asset a, a risk-free rate and the market respectively. The processes $\{\epsilon_t\}$, $\{\eta_t\}$ and $\{\nu_t\}$ are mutually independent white noise with variances σ_ϵ^2, σ_η^2 and σ_ν^2 respectively. This model representation can be thought of as a generalization, where the canonical CAPM (equation 3.1) is a special case where $\sigma_\eta = \sigma_\nu = 0$, i.e. the variances of these parameters are zero and therefore they are constant. We also denote the random processes $A = \{\alpha_t\}$ and $B = \{\beta_t\}$. As stated, in this work we also allow α to be time varying and this is facilitated by assuming it follows a random walk (equation 4.7). Also, because we are interested in interpreting A, it is mandatory to include $R_{f,t}$, otherwise the error terms in equations 4.7 and 4.8 will be contemporaneously correlated and α_t would have a different interpretation [99]. By contemporaneously correlated we mean correlation between two error terms in simultaneous equations.

Our state space representation still allows for the traditional interpretations of the CAPM, with the only difference being that these processes are potentially time-varying. After the white noise is removed from the excess return data, the process B represents the part of the excess returns of the asset explained by excess returns of the market and the process A represents excess returns of the asset not accounted for by the excess returns of the market.

Our proposed model (equations 4.6-4.8) is a state-space model and can be fitted using the diffuse Kalman Filter (DKF) and smoother (DKS) of [60]. De Jong's paper deals with unobservable initial conditions, hyperparameter estimation via maximum likelihood, and smoothing, which allows for predictions of $E(\alpha_t|y_1, \ldots, y_n)$ and $E(\beta_t|y_1, \ldots, y_n)$, $t = 1, \ldots, n$, together with their mean squared errors. If only the KF was used in the analysis then the predictions would not incorporate all the information available, rather for any time, t, $E(\alpha_t|y_1, \ldots, y_{t-1})$ and $E(\beta_t|y_1, \ldots, y_{t-1})$. Smoothing the output of the KF is an integral component of state space modelling.

To demonstrate the different outcomes from fitting this model figure 4.1 displays the 4 possible combinations of σ_η and σ_ν from four stocks from the S&P 500: BA has $\sigma_\eta = \sigma_\nu = 0$, WMT has $\sigma_\eta, \sigma_\nu > 0$, XOM has $\sigma_\eta = 0$, $\sigma_\nu > 0$, BEAM has $\sigma_\eta > 0$ and $\sigma_\nu = 0$. Additionally in figure 4.2 we have the α-β scatter plots, where the trajectories of their interactions are displayed. Different line colours along with their ending symbols are used for different time intervals but all time intervals are of the same length (five years in all our examples). In this way longer stretches denote acceleration and shorter stretches denote less change in the time period.

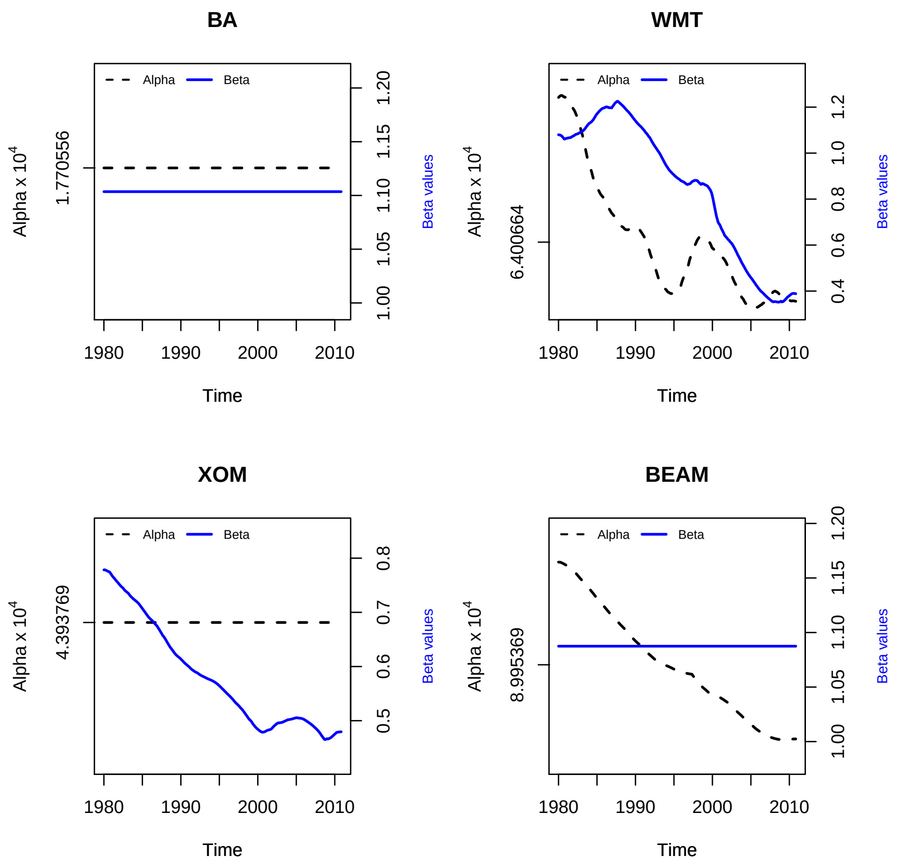

Figure 4.1: Predicted α_t and β_t values, left axis for predictions of α_t, right axis for predictions of β_t, for four stocks from the S&P 500.

4.2.1 Fitting the Model

As stated, we fit our proposed model using the diffuse Kalman filter (DKF) and smoother (DKS) of De Jong [60]. The DKF is a derivation of the original KF that allows for better approximations to real world data when the initial conditions are considered diffuse or nearly diffuse. Modelling a system under the assumption of a diffuse process means that little to no information is available concerning the initial conditions. In this case the unobservable initial conditions are α_1 and β_1 but we do not use the fully diffuse form of the DKF as we estimate the initial conditions using the canonical OLS regression (equation 3.2) on a small sample of the data. The diffuse Kalman Filter for this model is the basic KF (see chapter 2.1.12) with new equations for e_t and a_t given by the following:

$$E_t = (0, y_t) - Z_t A_t \tag{4.9}$$

$$A_{t+1} = A_t + K_t E_t \tag{4.10}$$

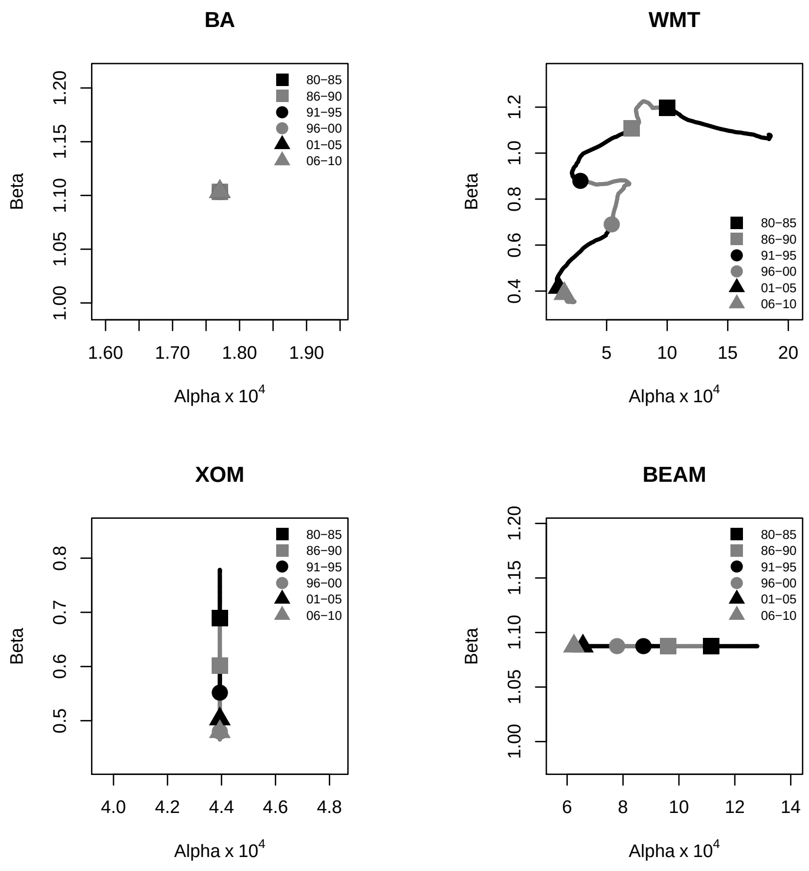

Figure 4.2: Alpha-Beta trajectories for four stocks from the S&P 500.

where the notation $(0, y_t)$ signifies a combination of column vectors side-by-side. We model the initial condition of the state vector $(\alpha_1, \beta_1)' = c' + \gamma + H_0\zeta$, where c denotes the 2×1 vector with entries given by the intercept and slope of the ordinary regression of $R_{a,t}$ on $R_{m,t}$, for $t = 1, \ldots, n_0$, where $n_0 < n/4$, γ is a random vector with unknown mean and variance, $\zeta \sim (0, 1)$ and H_0 is a diagonal matrix with diagonal entries given by the standard errors of the components of c, also generated from the OLS regression. Let $P_1 = H_0H_0'$, $A_1 = (-I_2, c)$, $Z_t = (1, x_t)$, $Q_1 = 0_3$, where I_k denotes the $k \times k$ identity matrix, and 0_k denotes a $k \times k$ matrix of zeroes.

The SSM representation is as parsimonious as the simple linear regression model where it has the same number of unknown parameters as the canonical CAPM. Other extensions of the CAPM model have been proposed which increase the number of unknown parameters, these extension are known collectively as multi-factor models (MFM) (such as the multifactor CAPM [37]). These MFM increase the complexity without possibly taking into account all non-trivial influences on asset returns. For example the work presented on behavioural finance (section 2.2.2) discusses factors that would be difficult to quantify by

Asset	Classification	Time Period
gold	precious	01/1980 - 12/2010
palladium	precious	01/1994 - 12/2010
silver	precious	01/1980 - 12/2010
copper	industrial	01/1987 - 12/2010
lead	industrial	01/1987 - 12/2010
nickel	industrial	01/1987 - 12/2010
tin	industrial	01/1990 - 12/2010
aluminium	industrial	01/1988 - 12/2010
aluminium alloy	industrial	01/1993 - 12/2010
S&P 500	market index	01/1980 - 12/2010
1 month T-bill	risk-free	01/1980 - 12/2010

Table 4.3: Summary of metals data analysed.

covariates in a regression model such as the disposition effect and sentiment. Additionally, the relationship between these factors and risk is most likely to be variable as well, and therefore a linear model would not be appropriate. However the proposed model allows for these unknown effects to be modelled dynamically and because it is flexible, additional covariates could be added if required. For example if one was invested in several international markets then it may be of interest to quantify the non-systematic risk on an asset in relation to two or more market indexes.

4.2.2 Metals Case Study

In this section we present the results from fitting the proposed model (equations 4.6-4.8) in an effort to test if the time varying risk-reward relationship holds for precious and industrial metals. Table 4.3 provides details of the metals included in the study as well as the market benchmark and the risk-free rate. All metal data was obtained from Bloomberg[3]. The time-period covered varies from metal to metal but at a minimum all time periods are at least 17 years, with the majority covering over twenty years. The data consists of daily observations of the closing prices where values for the time aggregates used for the analysis (quarterly, monthly and semi-monthly) are derived by averaging the daily returns at each time granularity considered. This is in contrast to other studies which would use returns over these time-periods, i.e. a monthly return rather than a monthly average of daily returns. This data pre-processing method was utilized as a means of capturing most of the information in daily returns.

The results from fitting the model using the DKF and DKS are presented in table 4.4 and

[3]Obtained from Bloomberg, last access was August 23, 2011

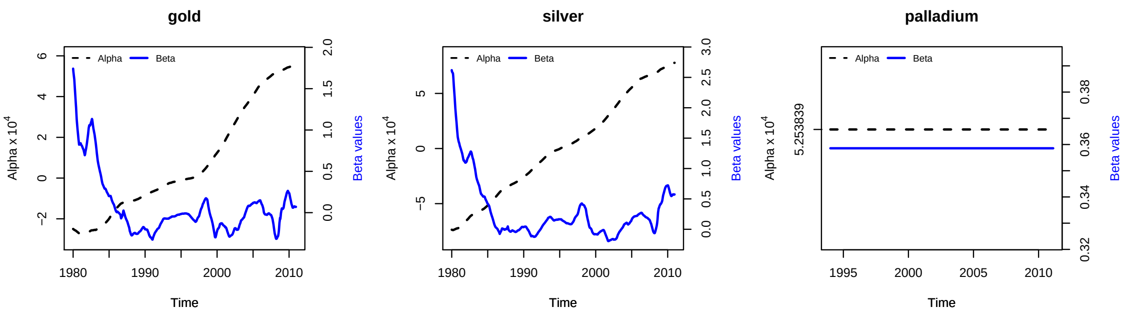

Figure 4.3: Plots of α_t and β_t for the precious metals.

plots of the semi-monthly time granularity are displayed in figures 4.3 and 4.4. Reported in table 4.4 are the maximum likelihood estimates of the model parameters σ_ϵ, σ_η and σ_ν which indicate if there is time-variation in α_t or β_t, where positive values for σ_η and σ_ν indicate time variation.

If we begin the discussion with the precious metals (gold, silver and palladium) we observe two different types of behaviour. Gold and silver have time variation in both α_t and β_t as shown in figure 4.3, whereas the canonical CAPM basically holds for Palladium (figure 4.3), where a constant but slightly positive α_t indicates that a portion of the excess returns are not accounted for by the market. The only industrial metal to have time variation in α_t is tin where both σ_η and σ_ν are > 0. The unique behaviour of tin is most likely influenced by the EU policy change that required that all solder contain 97.5% tin instead of the previous requirement of 40% by July 2006. This policy change increased the demand for tin and thus with a limited supply an increase in the price and investment returns. The other industrial metals all exhibit similar behaviour (figure 4.4) where the parameter estimates of $\sigma_\eta = 0$ and $\sigma_\nu > 0$, indicating constant α_t and time varying β_t.

From these results we can conclude that based on σ_ν there exists a time varying risk to reward relationship between precious/industrial metals and the market. Additionally, the results demonstrated time variation in α, which means that the proportion of an asset's return not accounted for by the market is also dynamic. In terms of risk analysis, our proposed model allows for the estimation of both variables, market and non-market, which could aid in portfolio optimization when an active approach to portfolio management is sought. For more details on the approach please refer to Mazzi and Butler [99].

Metal	Time Granularity								
	Quarterly			Monthly			Biweekly		
	σ_ϵ	σ_η	σ_ν	σ_ϵ	σ_η	σ_ν	σ_ϵ	σ_η	σ_ν
Alum.	0.0016	0.00000	0.1520	0.0032	0.00000	0.0444	0.0045	0.00000	0.0164
Al.all.	0.0016	0.00000	0.1980	0.0025	0.00006	0.0148	0.0034	0.00005	0.0098
Tin	0.0017	0.00008	0.1160	0.0028	0.00006	0.0148	0.0041	0.00004	0.0098
Lead	0.0021	0.00000	0.1510	0.0038	0.00000	0.0569	0.0059	0.00000	0.0222
Copper	0.0023	0.00000	0.1730	0.0038	0.00000	0.0489	0.0052	0.00000	0.0207
Nickel	0.0039	0.00000	0.1400	0.0058	0.00000	0.0398	0.0076	0.00000	0.0154
Gold	0.0012	0.00006	0.0000	0.0022	0.00003	0.0756	0.0033	0.00002	0.0765
Silver	0.0020	0.00010	0.1420	0.0039	0.00006	0.2290	0.0058	0.00004	0.0975
Palladium	0.0030	0.00000	0.0000	0.0050	0.00000	0.0000	0.0068	0.00000	0.0000

Table 4.4: Maximum likelihood parameter estimates of model (equations 4.6-4.8) for biweekly, monthly and quarterly returns of metals.

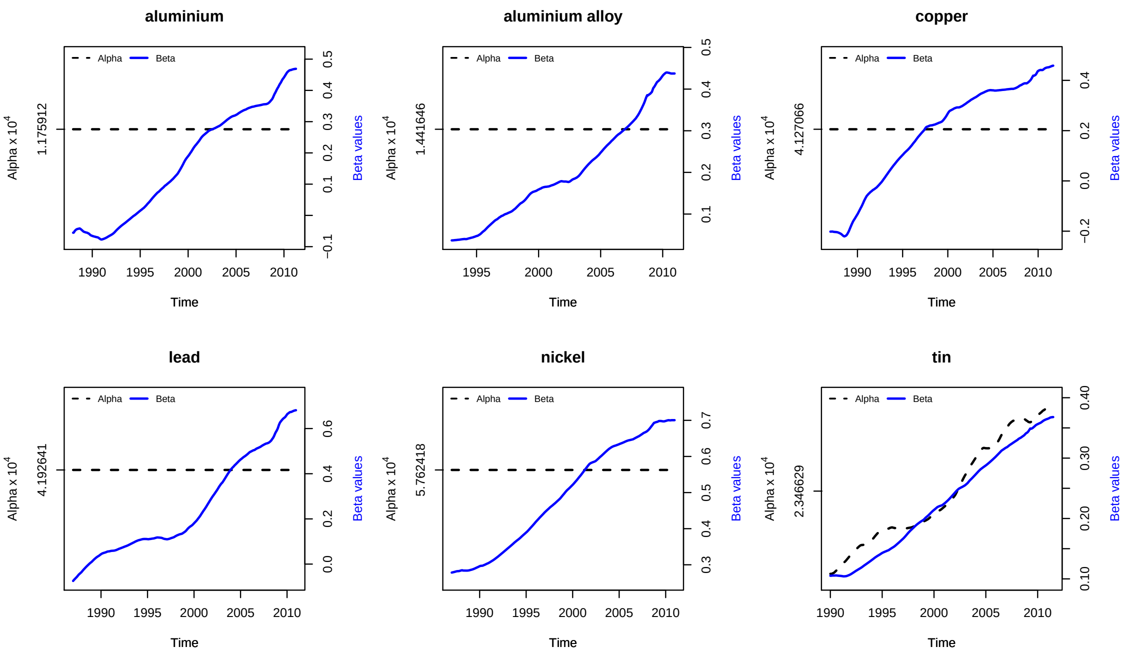

Figure 4.4: Plots of α_t and β_t for the industrial metals.

4.3 Variable Stationarity

Determining if a time series is level or trend stationary or if it contains a unit root is an important task for making informed financial decisions that involve forecasting. The reason stationarity is so important is that the assumption of a stationary series is fundamental to entire classes of modelling techniques from statistics and machine learning. If a time series is stationary, then any external shocks will only impact the series temporarily, however in a non-stationary series, these shocks will have a permanent effect on the moments (i.e. μ and σ^2). Put more formally, consider a time series $\{Y_t\}$ and an AR(1) model with drift:

$$y_t = a_0 + a_1 y_{t-1} + \epsilon_t \tag{4.11}$$

where y_t is the observation of the series at time t, a_0 is an intercept or drift term, a_1 is a constant, and ϵ_t is white noise. The process is stationary *iff* the constant $a_1 < 1$, so as to enable the system to return to a stable trend. If $a_1 = 1$ then the equation simplifies to:

$$y_t = a_0 + y_{t-1} + \epsilon_t \tag{4.12}$$

which is a random walk with drift model; this process is non-stationary and any shocks to the system will have a permanent effect. To observe why shocks to the system have a lasting effect we can solve for y_t given an initial condition y_0, so:

$$y_t = y_0 + a_0 t + \sum_{i=1}^{t} \epsilon_i \tag{4.13}$$

here the value of y_t is governed by a deterministic trend $a_0 t$ which depends solely on time and a stochastic trend Σ_{ϵ_i}, that is an accumulation of all previous shocks to the system from t_0 until time t. The presence of a unit root has important consequences for modelling and forecasting where non-stationary behaviour increases the complexity of the task. For these reasons it is important to determine the characteristics of the time series under study in order to transform it to stationary via differencing or de-trending.

Given the motivating factors from chapter 3.1.1, the investigation into whether or not variable stationarity exists is performed using two different statistical tests. The standard tests are utilized with a technique that is intended to detect time variation in a_1 from equation 4.11. We denote this possibility of time variation by the inclusion of a time subscript t, so we have $a_{1,t}$. The experiment performs a series of unit-root tests using a sliding window with overlapping periods.

4.3.1 Data Description

The data considered for this study are daily observations of the adjusted closing prices from 18 market indices from around the world. The sample spans a 10 year period from the beginning of January 2000 to the end of December 2009. All observations are log normalised as a pre-processing step. Table 4.5 displays the 18 market indexes (and their corresponding country) and the number of observations for each. Although the time span is of equal length, the number of trading days for each market varies slightly. The 18 markets represent 9 developed and 9 emerging markets based on the classification provided by the Morgan Stanley Capital index methodology to define developed and emerging stock markets.

4.3.2 Sliding Window Approach

The experiment framework is similar to the studies investigating variable efficiency where the fraction of windows rejecting the null hypothesis at different significance levels will be reported. To facilitate this objective a sliding window approach is utilised which moves in increments of 1 day at a time. For a time series $\{Y_t\}$ and a window of size d an initial

Table 4.5: The 18 market indexes considered in the study and the number of observations for each. The sample period is from January 2000 to December 2009. An asterisk signifies an emerging market.

Country	Market Index	obser.	Country	Market Index	obser.
US	S&P 500	2515	Indonesia*	Jakarta	2411
Korea*	KOSPI	2464	Malaysia*	KLSE	2464
Taiwan*	TSEC	2465	Argentina*	MerVal	2466
Japan	Nikkei 225	2454	China*	Shanghai Comp.	2579
Singapore	Strait Times	2509	UK	FTSE 100	2526
Hong Kong	Hang Seng	2489	France	CAC 40	2553
Brazil*	Bovespa	2472	Germany	DAX	2543
Mexico*	IPC	2506	Canada	TSX/S&P	2515
India*	BSE 30	2474	Australia	ASX(all ord)	2538

sub-sample is created consisting of observations $\{Y_{1,2..,d}\}$, the appropriate tests are run and then the window shifts by one day to cover $\{Y_{2,3..,d+1}\}$ and so forth until the end of the sample. We will consider sliding windows of size 250 and 500 observations, roughly one and two years of data respectively. The experiment will be performed twice with two different unit-root tests that both assume a null hypothesis that the series contains a unit-root. The first is probably the most widely used, namely, the Augmented Dickey Fuller test (ADF), where the time-series is fitted to a differenced AR(p) process of one of three possible forms:

$$\Delta y_t = \gamma y_{t-1} + \sum_{i=2}^{p} \beta_i \Delta y_{t-i+1} + \epsilon_i \tag{4.14}$$

$$\Delta y_t = a_0 + \gamma y_{t-1} + \sum_{i=2}^{p} \beta_i \Delta y_{t-i+1} + \epsilon_i \tag{4.15}$$

$$\Delta y_t = a_0 + \gamma y_{t-1} + a_2 t + \sum_{i=2}^{p} \beta_i \Delta y_{t-i+1} + \epsilon_i \tag{4.16}$$

If we assume a unit-root process for the difference equations then equation 4.14 is a simple random walk, equation 4.15 is a random walk with drift and equation 4.16 is random walk with drift and a linear time trend. The process to determine if the time series contains a unit root (i.e. $a_1 = 1$) entails estimating the coefficients (a_0, a_2 and β) for one of the equations above using ordinary least squares (OLS). If we let $\gamma = a_1 - 1$ then testing $a_1 = 1$ in the AR(1) model is equivalent to testing $\gamma = 0$ in the difference equations. The estimate of γ along with its standard error (SE) will form a t-statistic which can be evaluated against the critical values in the Dickey-Fuller tables.

The ADF test assumes that the errors are statistically independent with a constant variance, which can potentially be problematic, therefore in [110] the Phillips-Perron (PP) test was proposed to relax some of the assumptions on the error terms. The test statistic can be evaluated for significance using the same table of critical values as the ADF test. The regression equation considered in the PP test is:

$$y_t = \mu + \beta(t - T/2) + \alpha y_{t-1} + u_t \tag{4.17}$$

where μ, β and α are determined by OLS and the unit root test is determined under the assumption that the data are generated by $y_t = y_{t-1} + u_t$. Under the PP test the error terms, u_t, are allowed to be weakly dependent and heterogeneously distributed. From equation 4.17 we see that an advantage of using the canonical form of the PP regression equation is that there is no need to choose the lag. Though, like the Dickey-Fuller tests this equation can be expanded to include additional lags. By default the PP test includes a drift term and linear time trend. For the purpose of this study the ADF test will also be performed with a difference equation that includes a drift term and a linear time trend (equation 4.16).

The effectiveness of the ADF and PP tests depend upon the choice of p the lag parameter. There are two popular methods for choosing the lag parameter, the first is using Akaike's information criterion (AIC), and the other is based on the statistical significance of the estimated coefficients, as suggested by Ng and Perron [107] and denoted the NP method. In [107] the authors found that the use of an information criterion tended to underestimate the correct number of lags in the autoregressive equation and therefore the power of ADF test suffers. Therefore, in the thesis the NP method is used to fit the lag parameter of the unit-root tests. The pseudo code for the NP method is shown in algorithm 6, where this procedure is more effective in the general → specific direction, as in the reverse the number of lags is generally underestimated. This is due to the fact that having a lag k which is significant does not imply that all lags $< k$ will also be significant.

Algorithm 6 NP method for estimating the lag

```
1) calculate maximum lag: $P_{max} = [12 \times (T/100)^{1/4}]$, $T$ is the sample size
2) set order of AR model $lag(p) = P_{max}$
3) Fit AR model to training data
4) Calculate t-statistic for $lag(p) = t_{stat} = \hat{\phi} \,/\, SE(\hat{\phi})$, $\hat{\phi}$ is the estimated coefficient
while lag(P) is not significant do
    set $lag(p) = p$-1
    repeat steps 2-4
end while
set $ADF_{lag} = p$
```

4.3.3 Sliding Window Results

Upon applying the unit root tests using the methodology described above the following results have been obtained. Table 4.6 displays the results for the ADF and PP tests, respectively, performed with each window size. Reported are the fraction of windows which rejected the null hypothesis of a unit root at the 5% and 10% significance levels. The results from both unit-root tests indicate that across all markets there are periods where the null of a unit-root is rejected and thus the series can be considered trend stationary. However the number of rejections tends to be quite low and thus the majority of the periods could be considered non-stationary. This observation is not surprising given that the markets are generally held to be non-stationary . In figure 4.5 we have plots of the p-values from the unit-root tests for the US and Singapore market indexes using a sliding window of 500 observations. The dashed and solid lines indicate the 5% and 10% significance levels and clearly show the time-periods were the null is rejected. It is interesting that in both markets and according to both tests there were periods of trend stationarity around 2006, a couple of years prior to the most recent market crash of 2008. Additionally we observe that the patterns of the p-values are very similar between the unit-root tests. A short coming of this study is that the results are dependent on the window size, type of unit-root test and the choice of the lag parameter.

Table 4.6: The Augmented Dickey-Fuller and Philips-Perron test results. Displayed is the fraction of windows rejecting the null hypothesis for each market and window size at the 5% and 10% significance levels.

	ADF				PP			
	250		500		250		500	
Country	5%	10%	5%	10%	5%	10%	5%	10%
US	0.006	0.02	0.04	0.113	0.023	0.065	0.096	0.198
Korea	0.021	0.049	0.013	0.017	0.047	0.087	0.014	0.02
Taiwan	0.023	0.045	0.014	0.035	0.029	0.05	0.022	0.074
Japan	0.03	0.061	0.03	0.057	0.058	0.111	0.05	0.063
Singapore	0.036	0.078	0.01	0.036	0.092	0.161	0.068	0.133
HongKong	0.013	0.029	0.032	0.068	0.016	0.043	0.064	0.112
Brasil	0.031	0.067	0.022	0.059	0.067	0.101	0.02	0.052
Mexico	0.031	0.051	0.007	0.017	0.049	0.087	0.005	0.019
India	0.018	0.04	0.018	0.072	0.013	0.042	0.004	0.018
Indonesia	0.01	0.024	0.021	0.081	0.013	0.033	0.017	0.07
Malaysia	0.008	0.013	0.008	0.039	0.009	0.033	0.006	0.021
Argentina	0.028	0.046	0.046	0.101	0.031	0.079	0.03	0.097
China	0.018	0.035	0.008	0.029	0.028	0.066	0.009	0.046
UK	0.044	0.077	0.026	0.095	0.121	0.207	0.157	0.307
France	0.031	0.078	0.016	0.041	0.107	0.196	0.058	0.164
Germany	0.034	0.079	0.007	0.025	0.068	0.16	0.018	0.046
Canada	0.032	0.064	0.016	0.03	0.064	0.105	0.022	0.061
Australia	0.029	0.072	0.013	0.037	0.046	0.092	0.012	0.053

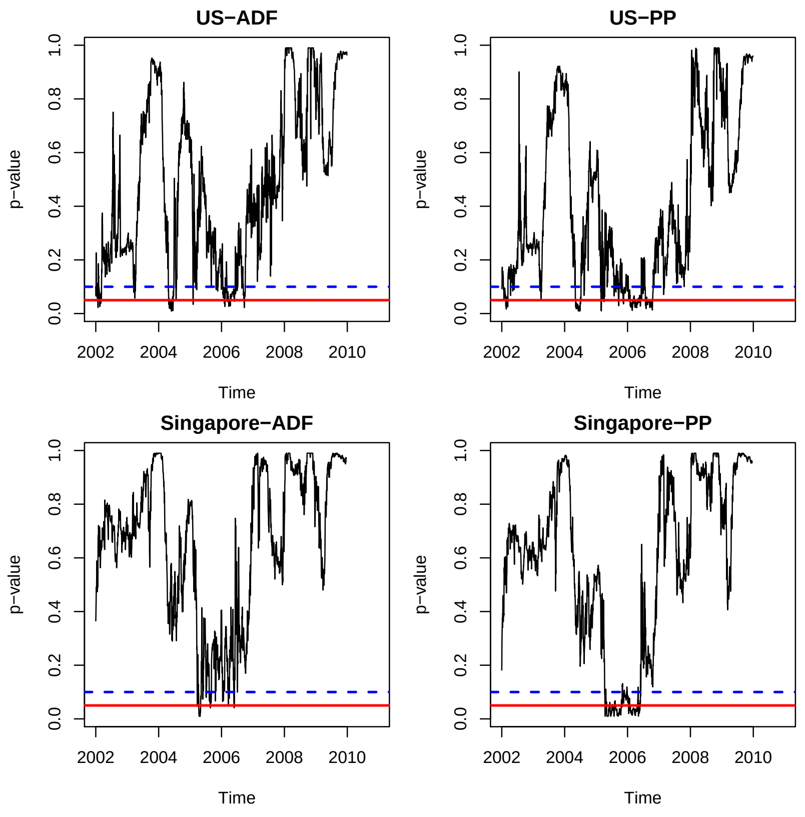

Figure 4.5: Plots of the p-values for the US and Singapore market indexes for the ADF and PP tests using a sliding window of 500 observations. The red (solid) and blue (dashed) lines are the 5% and 10% significance levels respectively.

4.4 Chapter Summary

In this first contribution chapter we have provided further evidence in support of the AMH in the form of three studies which analyse the dynamic nature of the financial markets. The first study reproduced results from the econometrics literature on variable efficiency; this not only provided evidence of reproducibility but also served as a means of code validation for subsequent experiments to be performed. Next we considered the risk-to-reward relationship for precious and industrial metals, where we demonstrated that for the majority of the metals included in the study the risk-to-reward relationship was time-varying as implied by the AMH. Finally, the characteristic of non-stationarity which is generally held to be a static property of financial time-series was analyzed. The results established that in all markets considered there were departures from non-stationarity and that time periods existed where the markets were in fact trend stationary. This result further demonstrates the dynamic behaviour of the financial markets and that departures from random walk behaviour could, in part, be caused by the periodic disappearance of the unit-root in financial time-series.

Chapter 5

Implications

Given the previous discussion, the AMH has been demonstrated as a reasonable characterization of market behaviour. Thus, this chapter explores some of the specific implications of the AMH and their effect on computational intelligence algorithms. This begins with an analysis of variable efficiency and how it affects forecasting with supervised learning algorithms. Secondly, the effects of departures from non-stationarity on the level estimation task of ANN are explored and, finally, we investigate the effect of the waxing and waning of investment strategies.

5.1 Variable Efficiency

From sections 3.1.2 and 4.1 we can conclude on the basis of the statistical metrics considered that there exists periodic non-linear dependence in the financial markets. However there still remains the question whether or not active trading strategies or technical analysis can take advantage of these inefficient market periods. The observation that market efficiency is cyclical is dependent on the robustness of the statistical test. From a forecasting point of view, the most important question, assuming a cyclical nature to market efficiency, is whether or not the implied periods of non-linear dependence can be used to improve forecasting accuracy and therefore lead to more profitable trading models. This is the motivation to determine if the *presence* of non-linear dependencies in a time series offers any benefits to forecasting models developed from machine learning techniques. The word 'presence' is emphasized as the actual data generating process is not known and any dependencies identified are contingent on the robustness of the statistical test.

To analyze the effect of non-linear dependence in a time-series on the forecasting accuracy of supervised learning, a generalized autoregressive conditional heteroskedasticity (GARCH) model is used to simulate a financial time-series. A GARCH model, as the name suggests, allows for conditional variance that is not constant through time, as is commonly observed in financial time series. The form of a $GARCH(p, q)$ process for a series of discrete observations $\{Y_t\}$ is given by the following:

$$Y_t = \sigma_t \epsilon_t \tag{5.1}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2 \tag{5.2}$$

where ϵ_t is standard Gaussian white noise and the condition that $\sum_{i=1}^{p} \alpha_i + \sum_{i=1}^{q} \beta_i < 1$. Equations 5.1 and 5.2 return a white noise process with non-constant conditional variance, where the variance depends on the previous return. Equation 5.2 can be easily extended to include more lags. For the purpose of this study a GARCH(2,2) model was used to simulate the data series. A simulated data series was chosen as means to focus the analysis on the algorithms and their performance as opposed to being unduly influenced by shortcomings of the statistical test. Using the GARCH process allows for a more controlled environment where the influence of non-linear dependence can be meaningfully determined. From a computational intelligence perspective we want to know if non-linear dependence has an effect on classification accuracy; whether or not the test is robust to real market conditions is a question for econometricians. In this study the H-statistic (described in section 4.1.1) is used to identify non-linear dependence.

5.1.1 Supervised Learning

We are interested in the effect, if any, that non-linear correlations have on the forecasting abilities of trading models developed from supervised learning (SL). The advantage of focusing on forecasting accuracy in terms of classification is that the results are easily interpretable. If the investment returns were used as in [130] then they would have to be risk-adjusted to make any meaningful conclusions. There is no shortage of literature on SL techniques being developed and applied to the financial domain. The dynamic and non-linear nature of the financial markets makes them a challenging and attractive system to model using complex methods. This study focuses on six widely available learning algorithms that represent four well established learning paradigms. The algorithms considered are:

1. Multilayer Perceptron (MLP)

2. Support Vector Machine (SVM)
3. Artificial Immune System (AIS)
4. J48 Decision Tree (J48)
5. *k*-Nearest Neighbour (kNN)
6. Naïve Bayes (NB)

where the MLP and SVM are function approximators, AIS and kNN are instance based learners, J48 is a decision tree learner and, finally, NB is a simplified approach to Bayesian learning. In terms of forecasting financial time series, all of the listed algorithms have been previously used in other studies with the exception of AIS. The related work on the overlap of AIS and finance never concerned time series forecasting directly, however a more recent study by Butler et al. [12] demonstrated that in general the practice of using the natural immune system to inspire a learning algorithm is a viable alternative to modelling financial time series when implementing a supervised learning approach. The full details of the study are available in Appendix B.

5.1.2 Experiment Setup

Using the sliding window methodology previously discussed, the simulated GARCH(2,2) series (figure 5.1) is segmented into samples; one sample consisting of sliding windows that contain non-linear dependence and the other consisting of data that adheres to a stochastic random walk. In other words, a sample is a set of window frames. The size of the window is 200 observations and the rejection of the null is determined at the 5% significance level. The forecasting task for each of the algorithms is classification. Each tuple of information supplied to the various SL techniques will have 5 consecutive lagged values of the time series and a class attribute (C_i) where $C_i \in \{0, 1\}$. 0 signifies a market contraction and 1 signifies a market expansion. The SL algorithms are then applied to the separate samples, where 75% is allocated for training and 25% for testing. The parameter settings for the SL algorithms are kept constant between the samples and are presented in table 5.1 (only parameters different from the default settings in WEKA-3-7 are reported). Any changes to the parameters were a result of experimentation on the training data. The experiments are performed in Java using the WEKA libraries [47] for the various SL algorithms. In addition to the experiment code, a purpose built Java class was implemented for pre-processing the data from CSV format to the Attribute Relation File Format (arff) of WEKA-3-7.

Table 5.1: The parameter settings for the various supervised learning algorithms that are different from the default settings in WEKA-3-7.

Algorithm	Parameters
MLP	learning rate = 0.1
SVM	cost parameter for C-SVC = 60
AIS	all default settings used
J48	pruning confidence factor = 0.50
kNN	k = 7
NB	all default settings used

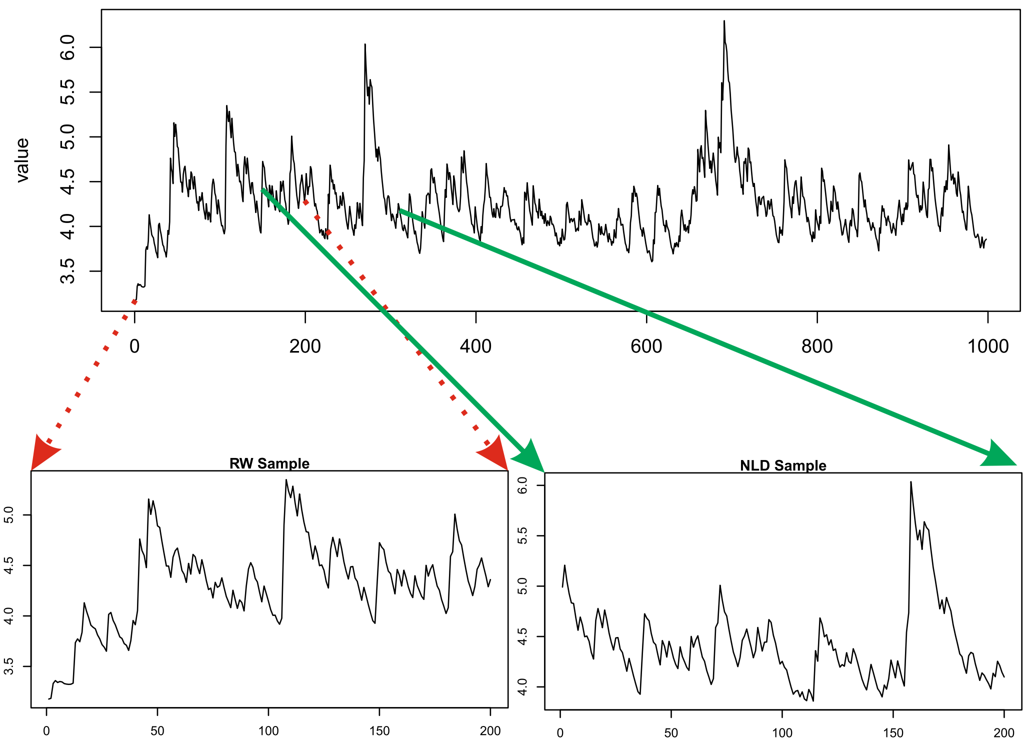

Figure 5.1: (top) A plot of the GARCH series used in the study. (bottom) Example plots of the GARCH process when it is exhibiting non-linear dependence (NLD) (right) and random walk (RW) behaviour (left).

This experiment setup allows us to answer the question: “Keeping everything else constant, do we expect an increase in the classification accuracy of supervised learning algorithms when non-linear dependence is detected?”. Here we use the traditional interpretation of expectation that is, whether or not we expect an increase on the average. Figure 5.1 displays a plot of the GARCH series used in the analysis (top) and examples of subsamples of this series that contain non-linear dependence (right) and random walk behaviour (left).

*Table 5.2: The results from training and testing the SL algorithms on the GARCH subsample data. NLD represents samples with non-linear dependence and RW represents samples adhering to a random walk. *, ** signifies the increase in accuracy is statistically significant at the 5% and 1% levels respectively.*

	RW			NLD		
Algorithm	Acc.	Min.	Max.	Acc.	Min.	Max.
MLP	0.587	0.347	0.755	0.622**	0.367	0.796
SVM	0.625	0.510	0.796	0.656**	0.531	0.775
AIS	0.569	0.327	0.796	0.580*	0.388	0.755
J48	0.617	0.469	0.755	0.656**	0.429	0.755
kNN	0.629	0.428	0.775	0.633	0.367	0.861
NB	0.617	0.429	0.796	0.651**	0.490	0.775

5.1.3 Experiment Results

The results in table 5.2 and figure 5.2 show that all 6 algorithms achieved a higher directional accuracy in the subsamples that exhibited non-linear dependence and in 5 of the 6 cases the increase was statistically significant based on a one-sided t-test. The only exception was the kNN algorithm where only a small incremental gain was realized, however the overall accuracy was comparable to the other algorithms. These results indicate that when non-linear dependence is present the SL algorithms tested were able to take advantage of this deterministic component of the signal. A caveat to be made is that these results are for classification accuracy and not investment returns, and therefore we cannot conclude with absolute confidence that these results indicate higher investment returns. However, based on the results from Leung et al. [81] which showed that investment returns and classification accuracy are highly correlated, we can conclude that this result indicates that the models will most likely (or on average) be more profitable during periods of non-linear dependencies as well.

5.1.4 Variable Efficiency Conclusions

One should put these results in perspective of the related work on filter approaches discussed in 3.2.2. The conclusion that SL algorithms are able to take advantage of non-linear dependence, strengthens the argument that the filter approaches are effective. Additionally it implies that the filters were indirectly indentifying non-linear dependence in the financial time series. The filter approaches focus on forecasting metrics and therefore are unable to make any rigorous statistical inferences as to the dependencies (linear or non-linear) in the time series. The results from this study demonstrate in

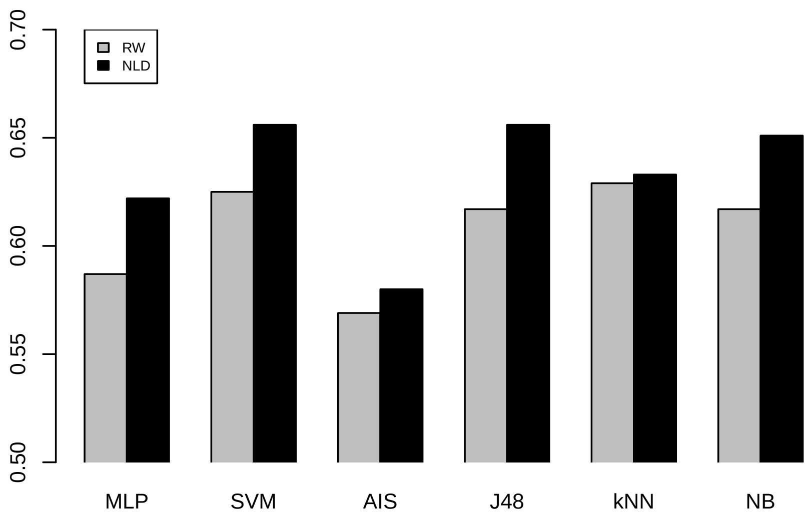

Figure 5.2: A Histogram of the testing accuracy results from the Random Walk (RW) and non-linear dependent (NLD) subsamples.

a quantitatively sound framework that supervised learning algorithms can model non-linear dependence and, if it is reliably detected, a statistically significant increase in classification accuracy can be expected.

With respect to reliably detecting non-linear dependence there is an opportunity to improve this process using the Bonferroni Correction (BC) [32]. The BC is a method to counteract the problem of incorrectly rejecting the null hypothesis (type 1 error) when performing multiple comparisons. In its most naive form, the desired α is divided by the number of tests (n) to be performed to yield a new level of significance, i.e. $\alpha = \alpha/n$. Using this stricter criterion to filter out time-periods with non-linear dependence, in theory, should widen the gap in classification accuracy between the two samples utilized previously.

5.2 Variable Stationarity

This section has two objectives, the first to investigate what effect variable stationarity has on artificial neural networks and, secondly, to determine the preferred methodology for training ANNs with non-stationary data. The latter objective aims to clarify the results reported in [72] and test their correctness.

Variable stationarity would most likely affect any algorithm from CI in time-series prediction, so it seems appropriate from a CI perspective to start the analysis of this phenomenon with arguably the most robust modelling CI technique for forecasting financial time-series, i.e. Artificial Neural Networks [12] [142] [6]. ANNs are considered non-linear function approximators, which makes them attractive for complex time-series. Indeed, studies have shown that modelling non-linear data with ANNs produces more accurate models than their linear counterparts. For this study we will be experimenting with a feed-forward multi-layer perceptron with two hidden layers and the canonical back-propagation for updating connection weights. The topology of the ANN used was shown in figure 2.1, where we have a (5,2,4,1) architecture with 5 inputs representing 5 lagged values of the time-series. The hidden nodes have sigmoid activation functions (equation 2.10) and the output node is linear (equation 2.11) as we require a real-valued output.

5.2.1 Experiment Setup

The main objective of this study is to examine the effects of departures from non-stationarity in a time-series on the forecast errors of ANNs. To facilitate this objective we train ANNs on stationary and non-stationary time series using three different pre-processing steps:

- First differenced (DIF): $log(P_t) - log(P_{t-1})$,
- De-trended by removing a linear time trend (DET), and
- No pre-processing (OBS).

where these three forms represent the preferred pre-processing method for input data when a time-series is difference, trend and level stationary respectively. The first differences of a log normalized series are essentially the one period returns. Thus by comparing the different forecast errors we can determine if suspected departures from non-stationarity impact the forecasts of the ANNs. The experiments will be conducted on data sets of increasing size and time-horizon. The real-world data used for this study is the same as for the test for variable stationarity that used the sliding window approach, see table 4.5. The subsamples from the real-world data will also be collected for this study with a sliding window approach which moves in increments of 1 day at a time. So, for a time series $\{Y_t\}$ and a window of size d an initial sub-sample is created consisting of observations $\{Y_{1,2..,d}\}$. Next the appropriate tests are run and then the window shifts by one day to cover $\{Y_{2,3..,d+1}\}$ and so forth until the end of the sample. If a sample is found to generate a statistically significant p-value at some specified limit then the sample is stored and a linear time trend is evaluated. The algorithm for determining stationary and non-stationary windows is described in algorithm 7, where the $ADF_{test}(L)$ is the ADF test for L lags (described

Algorithm 7 Identify Stationary/Non-stationary Windows

```
input: Data set D
input: $P_{max}$
input: windowSize
input: upper/lower significance levels(sigLevU, sigLevL)
for i← 1:length(D)-windowSize do
    sample(S) ← i:i+windowSize
    apply NG method to S
    return Optimal lag L
    apply $ADF_{test}$(L) to D
    return p-value($p_{val}$)
    if $p_{val}$ < sigLevL then
        store S to SW
        Trend ← regress S on time
        store Trend to ST
    end if
    if $p_{val}$ > sigLevU then
        store S to NS
        Trend ← regress S on time
        store Trend to NST
    end if
end for
return SW, NS, ST, NST
```

in chapter 4.3.2) and the *NGmethod* is the Ng and Perron method [107] for determining the lag of a unit-root test (algorithm 6). SW and NS are matrices of stationary and non-stationary data samples respectively, and ST and NST are vectors of linear time trends for the samples stored in SW and NS respectively. Once the relevant data has been collected, it is split into two equal sets for training and testing. The evaluation of the results will be performed using mean squared errors (MSE), where all the errors are calculated based on the price of the asset. This is vitally important for comparing the results as the difference data would always yield smaller errors (generally magnitudes smaller) since the values of the series are smaller. Thus, to have a fair comparison, the difference data forecast will be added to the previous day price to arrive at a forecast price and the de-trended data will have the trend added back in to arrive at a forecast price as well. So for example, if the difference data produces a forecast of 1% then the forecast price P_t for time t is $1.01 \times P_{t-1}$. This will enable the MSE to be calculated on the price forecasts for all pre-processing steps and therefore allow a truly fair comparison.

In addition to the real-world market data this study also considered a simulated time series as a means to compare the real-world results and to answer the question as to what is the

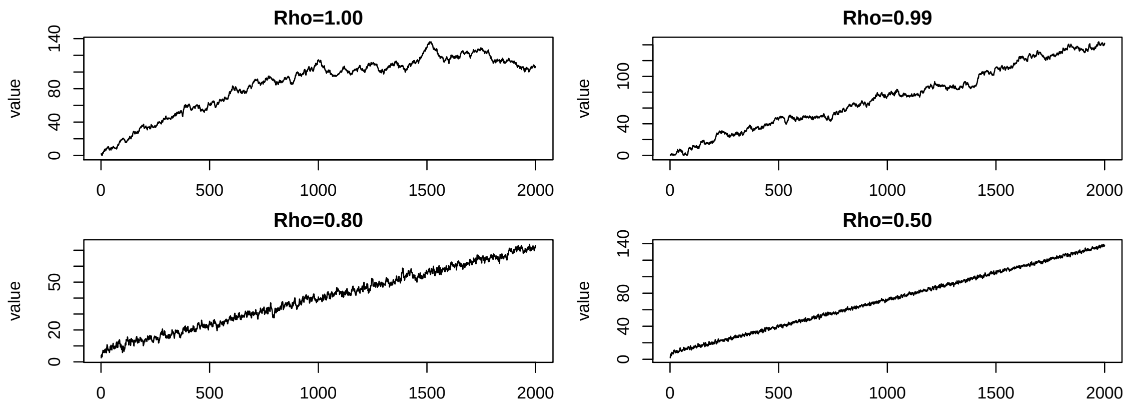

Figure 5.3: A plot of one realization for each of the four levels of persistence generated from $Y_t = (a - a\rho + b\rho) + (b - b\rho)t + \rho\, Y_{t-1} + \epsilon_t$.

preferred method for training ANNs with non-stationary data. For the simulated data experiments a series will be generated that contains 2000 observations with a 50/50 split for training and testing, using equation 5.3 (taken from Diebold et al. [29]):

$$Y_t = (a - a\rho + b\rho) + (b - b\rho)t + \rho Y_{t-1} + \epsilon_t \tag{5.3}$$

where $\epsilon_t \sim$ N(0,1), a = 7.3707 and b = 0.0065. The values for a and b were chosen to mimic post World War II Gross Domestic Product (GDP) data. ρ is a coefficient which determines the influence of past values on the present, the same as a_1 from equation 4.16, where ρ = 1 creates a non-stationary time series. The data is generated using increasing values for ρ that approach and eventually equal unity. Figure 5.3 displays the four simulated realizations of equation 5.3 used in the experiment.

5.2.2 Simulated Data Results

This section will detail the results from training and testing on the simulated data. In tables 5.3-5.5 PP_{step} stands for pre-processing step, OBS, DET and DIF stand for observation, de-trended and differenced data respectively. All simulations are generated with a = 7.3707 and b = 0.0065. The reported results are the MSE from the forecasts using the different pre-processing steps. Table 5.3 displays the testing results from the three different pre-processing steps for each time-horizon and ρ.

From the $AR(p)$ results discussed in Diebold et al. [29], we would expect that when a series is trend-stationary, de-trending would be the preferred pre-processing method. Likewise, when the series is non-stationary we would expect to get the best forecasts from differencing the data. From the plots in figure 5.3 we observe that all of the time

Table 5.3: The MSE results from out-of-sample testing on the simulated results generated from equation 5.3, where $\epsilon_t \sim N(0,1)$, a = 7.3707, b = 0.0065 and ρ = {0.50, 0.80. 0.99, 1.00}.

		Time Horizon					
PP_{step}	ρ	1	5	10	20	50	100
OBS	0.50	1548	1555	1564	1578	1626	1712
DET		1.06	1.36	1.36	1.36	1.38	1.38
DIF		1.35	2.41	2.81	3.84	3.83	13.40
OBS	0.80	1547	1558	1572	1583	1631	1713
DET		2.08	2.85	2.93	3.02	3.15	2.92
DIF		1.26	5.23	5.72	6.02	6.60	8.44
OBS	0.99	1731	1741	1764	1785	1831	1925
DET		2.76	16.71	11.77	20.90	28.98	34.26
DIF		1.13	5.99	22.31	30.04	68.95	150.68
OBS	1.00	3.24	67.60	47.58	148.24	408.69	1569.34
DET		110.59	136.97	126.20	167.33	219.46	308.99
DIF		1.15	6.13	17.20	45.07	155.11	519.84

series contain a trend, and when $\rho < 1$, the series are trend-stationary.

The results in table 5.3 show that the preferred pre-processing step for $\rho = 0.50$ is de-trending and this is true for all time horizons. However, as we increase ρ, the preferred method becomes a function of the time horizon, where at shorter time horizons first differencing the data produces better forecasts. In time series analysis the ρ of a stationary series is referred to as its persistence, where higher values for ρ indicate more persistent processes. It is interesting to note that for the non-stationary series ($\rho = 1$) the preferred pre-processing method at the 100 day time horizon is de-trending rather than first differencing as might be expected. It should also be noted for comparison that using just the observations (as published in [72]) never produced a superior model and the forecast errors were generally magnitudes larger than in the case of differencing or de-trending.

This concludes the experimentation on the simulated data. We have shown that stationarity, or lack thereof, affects the performance of the ANN and that proper consideration for the characteristics of the time series leads to improved forecasting performance. The choice of the proper pre-processing step is a function of forecast-time horizon, persistence of the stationary process and the presence of a linear time trend.

5.2.3 Stock Market Data

This section will detail the results from applying the above procedure to sub-samples of actual stock market index data. The results reported in the tables are not the MSE as before but the percentage of samples that $preferred$ each pre-processing step. By preferred we mean that a given pre-processing step minimized forecasting error for a particular sample based on MSE.

5.2.3.1 P-Values = {0.99, 0.01}

This section reports results from training the ANN on samples that either accepted or rejected the null of the ADF test at the 0.99 and 0.01 levels of significance respectively. This criterion yielded 219 stationary and 1049 non-stationary samples for the 250 day window. For the 500 day window only 25 samples were considered stationary and 1537 samples were non-stationary.

Table 5.4: The results from out-of-sample testing on the stationary and non-stationary samples determined by an ADF test (with p-values ≤ 0.01) using a 250 and 500 observation sliding window.

			Time Horizon					
PP_{step}	Window	Type	1	5	10	20	50	100
OBS	250	S	0.0	0.0	0.009	0.027	0.027	0.023
DET			0.0	0.091	0.356	0.598	0.703	0.831
DIF			1.0	0.909	0.635	0.374	0.270	0.146
OBS		NS	0.0	0.031	0.041	0.160	0.137	0.388
DET			0.0	0.0	0.027	0.182	0.662	0.598
DIF			1.0	0.968	0.932	0.658	0.201	0.014
OBS	500	S	0.0	0.0	0.0	0.0	0.0	0.0
DET			0.0	0.0	0.0	0.24	0.680	0.840
DIF			1.0	1.0	0.1	0.76	0.320	0.160
OBS		NS	0.0	0.005	0.019	0.025	0.184	0.230
DET			0.0	0.0	0.0	0.012	0.269	0.587
DIF			1.0	0.995	0.981	0.963	0.547	0.113

The results from the real-world data (table 5.4) are akin to those obtained with the simulated data. The behaviour of the market is not always well represented by an $AR(1)$ process (the form of the simulated data) and this added complexity makes the benefits of pre-processing less definitive. To help visualize the differences between the pre-processing steps and the stationary/non-stationary samples, figure 5.4 plots the percentage of samples that preferred de-trending at each time horizon and sample type. From these plots we can observe that as the time horizon extended, the percentage of samples preferring de-trending increases, and in the trend-stationary samples this tendency is monotonic. Additionally, at all time-horizons the percentage of samples preferring de-trending was always equal to or greater than that for the trend-stationary samples. Next we relax the rejection level for the ADF test to an α=0.05 and perform the same experiment.

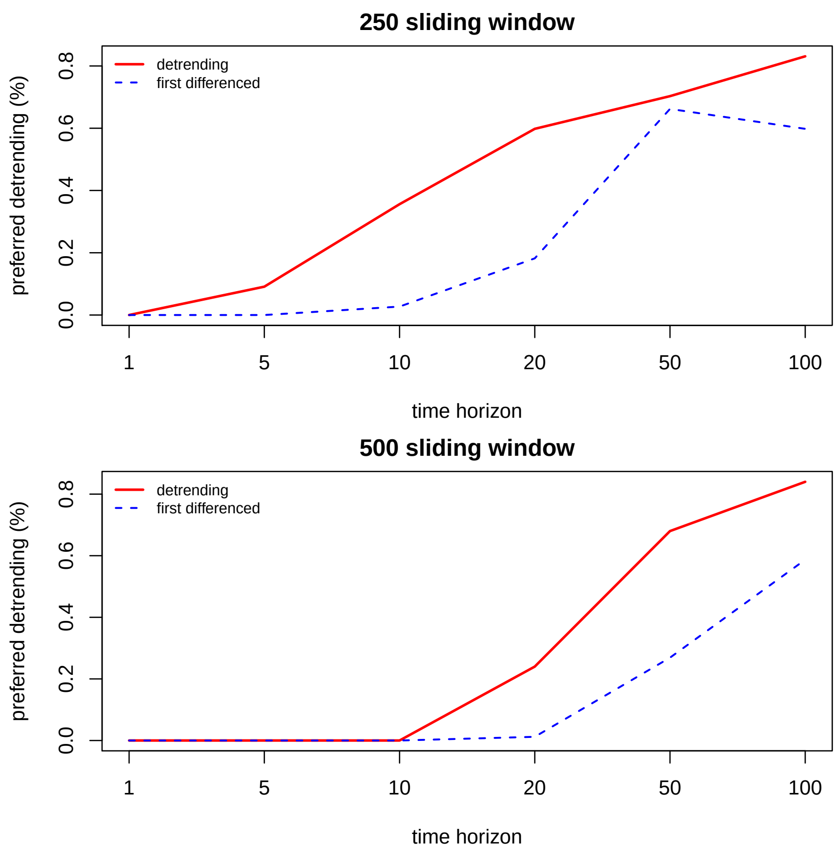

Figure 5.4: Plots of the percentage of stationary samples that preferred de-trending as a pre-processing step for the 250 and 500 sliding windows using the ADF test at $\alpha = 0.01$.

5.2.3.2 P-Values = {0.95, 0.05}

This section reports results from training the ANN on samples that accepted and rejected the null of the ADF test at the 0.95 and 0.05 levels of significance respectively. This criterion yielded 883 stationary and 3332 non-stationary samples for the 250 day window. For the 500 day window, 498 samples were considered stationary and 4879 samples were non-stationary.

The results in table 5.5 and figure 5.5 show similar results to those reported in the previous section. From the plot we can observe that de-trending is preferred more often when the series is considered trend-stationary. At all time horizons, a greater number of samples prefer de-trending when the series is trend stationary (except when these percentages are zero). The change in this percentage is also monotonically increasing with the time horizon for trend stationary samples and at time horizons of 50 points and greater de-trending is the preferred method for the majority of the samples. This result along with those previously discussed provide evidence that taking into account the local characteristics of the market will benefit forecasting financial time series with ANNs.

Table 5.5: The results from out-of-sample testing on the stationary and non-stationary samples determined by an ADF test (with p-values ≤ 0.05) using a 250 and 500 observation sliding window.

			Time Horizon					
PP_{step}	Window	Type	1	5	10	20	50	100
OBS	250	S	0.0	0.0	0.0	0.009	0.050	0.091
DET			0.0	0.046	0.160	0.475	0.740	0.758
DIF			1.0	0.954	0.840	0.516	0.210	0.151
OBS		NS	0.0	0.009	0.073	0.265	0.123	0.525
DET			0.0	0.0	0.005	0.192	0.498	0.447
DIF			1.0	0.991	0.922	0.543	0.123	0.027
OBS	500	S	0.0	0.0	0.004	0.068	0.100	0.100
DET			0.0	0.0	0.022	0.265	0.634	0.773
DIF			1.0	1.0	0.974	0.667	0.265	0.127
OBS		NS	0.0	0.0	0.0	0.022	0.168	0.494
DET			0.0	0.0	0.0	0.012	0.236	0.295
DIF			1.0	1.0	0.999	0.966	0.600	0.210

5.2.4 Non-stationarity and ANN

Our results appear to contradict a result published in [72]. The effects of non-stationary data on stationary models is well documented in the statistical literature and the most common method for modelling non-stationary data is to transform it to stationary. To begin, we show that an ANN can be written in the form of an $AR(p)$ which is widely understood to be a stationary model. If we consider a basic ANN with a linear activation function and some input bias, as depicted in figure 5.6, we can represent it using the following equation:

$$\hat{Y}_t = W_0 X_0 + \sum_{i=t-lag}^{t-1} W_i X_i \tag{5.4}$$

where W_0X_0 is the input bias and W_iX_i the lagged inputs of the series being multiplied by their respective weights. If we set $W_0X_0 = \alpha$ and assume the error of $\hat{Y}_t$ (of a fully trained ANN) to be uncorrelated white noise with $\mu = 0$ and σ^2_ϵ, then we have:

$$\hat{Y}_t = \alpha + \sum_{i=t-lag}^{t-1} W_i X_i + \epsilon_t \tag{5.5}$$

which is the same equation as an AR(p) model. Thus it seems logical that ANNs would most likely benefit from appropriate pre-processing techniques.

The original paper [72] was performing one-step ahead forecasts with daily data. Focusing on the short term forecast results from the simulated data, we can see that using the original observations never produced the optimal forecast of the three models. Since a validation set was not used, these results are directly comparable to each other. The most

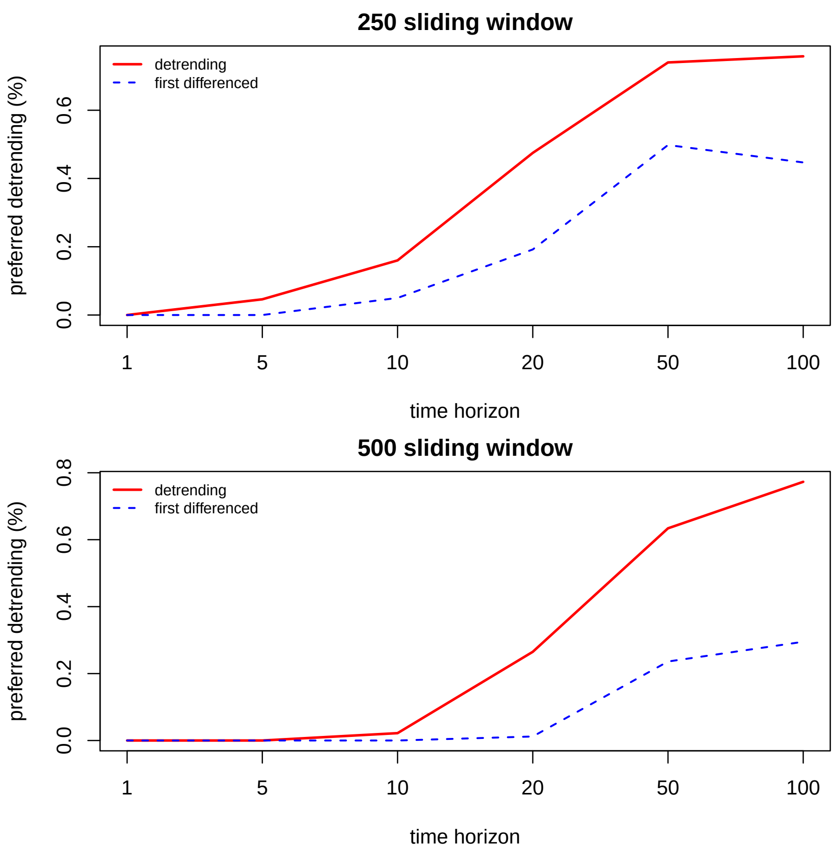

Figure 5.5: Plots of the percentage of stationary samples that preferred de-trending as a pre-processing step for the 250 and 500 sliding windows using the ADF test at $\alpha = 0.05$.

probable reason for the authors of [72] conclusion was that the patterns the ANN was learning were only valid for a short-time period. So without the validation set, there was a small period of time after training where the learned relationship between inputs and output was still valid, leading to slightly better results in the out-of-sample testing. An example of this problem is displayed in figure 5.7, where we have a plot of a non-stationary time series (top) and plots of the forecast errors from training an ANN with and without appropriate pre-processing. As we can observe, the error is similar in the initial period after training but as the series continues to evolve and the moments of the series change the error in the ANN trained without pre-processing begins to grow.

However, despite the results in [72], from the derivation of the ANN in to an $AR(p)$ model and the experimental results we can conclude that the preferred methodology for training ANNs with non-stationary data is to transform the data to stationary using an appropriate pre-processing step.

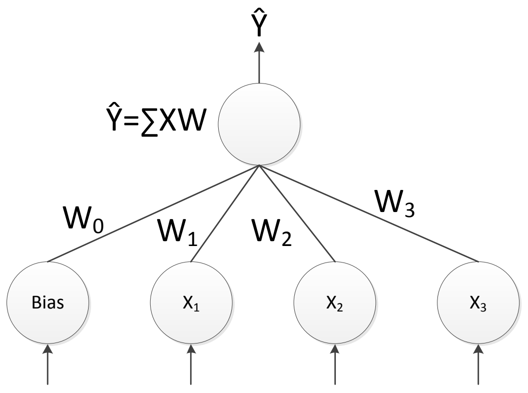

Figure 5.6: A basic artificial neural network structure with a linear activation function an input bias.

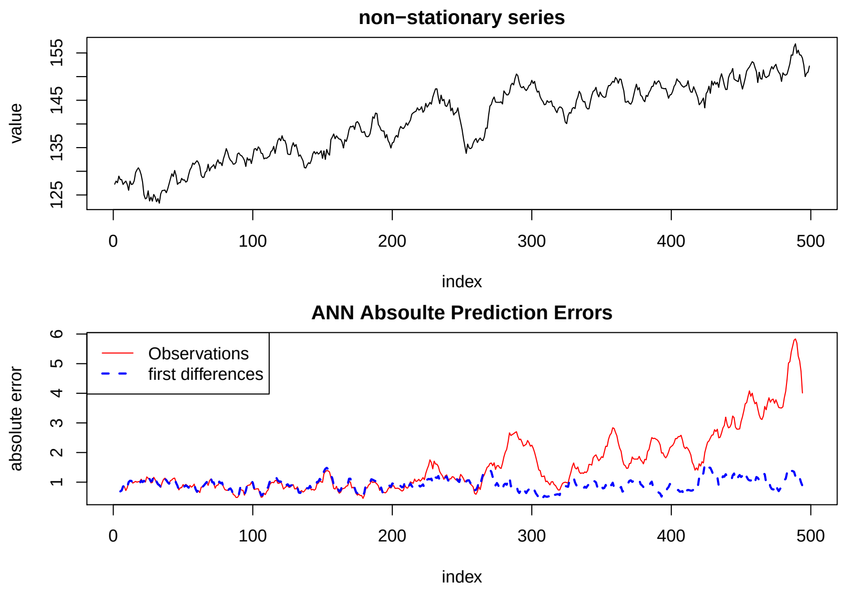

Figure 5.7: The absolute forecast errors for an ANN trained with and without appropriate pre-processing. Plotted values represent a 10-day moving average of the errors.

5.2.5 Variable Stationarity Conclusions

The results from the simulated data confirm that the effects of (non)-stationary of time-series on ANNs are similar to those of $AR(p)$ models that were obtained in [29]. Where de-trending was beneficial in lowly persistent trend stationary processes at any time-horizon and in highly persistent trend-stationary processes, the advantages to using de-trending were a function of the time-horizon and degree of persistence. The results from the real-world data are not as clear as the simulated, but in general, given an appropriate window size for calculating the ADF test, we observe results that roughly adhere to their simulated counter-parts. In the trend-stationary samples de-trending was the dominant pre-processing step at time-horizons of 50 days and above. Also de-trending was more often preferred at any time horizon in the trend-stationary samples as opposed to the non-stationary ones. The inferior performance on the real-world data sets could be a result of Type II errors, where some of the null acceptances could have been from structural breaks or other anomalies in the time-series that the unit-root test was sensitive to. This could possibly be overcome by using a test which allows for structural breaks, such as the Zivot and Andrews unit root test [143].

The results from the real-world data also present conclusive evidence that overfitting the training data is not the preferred method for training ANNs for financial time-series forecasting. Using just the observations was commonly the inferior model for the non-stationary data w.r.t. differencing at forecasting time-horizons of 50-days or less. When the use of the observations was superior to differencing (at the 100-day time-horizon) there was not a conclusive method for producing optimal results. As a consequence, the use of differencing in non-stationary financial data will produce the most accurate models in the majority of the cases up to forecasting time-horizons of at most 50 days. This conclusion is supported by the statistical properties of stationarity and its implications for modelling a time-series.

As noted before in section 5.1.4 the results in this section could be improved by utilizing the Bonferroni Correction, which will yield a more conservative approach to detecting departures from difference stationary in financial time series.

5.3 Cyclical Effectiveness

From the results discussed in chapter 3.2.3 [66] we have seen that trading models developed using genetic programming can be *useful* after the time period they were trained on. By useful we mean that their fitness functions achieved similar values in out-of-sample time periods. Equipped with only the fitness function results the *effectiveness* (redefined below) of the models cannot be determined. Active trading strategies must be able to outperform the market or other passive approaches to investing in order to warrant the extra time and expenses required to administer them. As a result, in this section we extend this work to a more meaningful interpretation that allows us to gauge the effect of the waxing and waning of investment strategies on computational intelligence. We hope to determine if CI models are ever *effective* again after the time they were trained for. To reiterate, we define effectiveness as:

$$\begin{array}{ll} \text{I} & R_t(TM) > R_t(M) \text{ outperforms the market,} \\ \text{II} & R_t(TM) \; > \; 0, \text{ yeilds a postive return, and} \\ \text{III} & T_t > 0 \text{ is actively trading in the market} \end{array}$$

where $R_t(TM)$ and $R_t(M)$ are the returns of the trading model and the market in time period t respectively and T_t is the corresponding number of trades in time period t. The criterion are also represented in figure 5.8 as a Venn diagram, where the region corresponding to the overlap of the three circles contains the models which are effective. As such a trading model is considered effective when these three criterion are satisfied. If trading models exhibit cyclical effectiveness then maintaining and consulting previous models may improve forecasting performance. In essence this would be a passing on of knowledge from older generations to new ones. This positive impact of older generations is seen in the natural world where the emergence of grandparents in human society led to an explosion of sophisticated tools and art [18]. There are various methods which could be explored to test the implication of cyclical effectiveness of trading models developed from computational intelligence. Concerning financial time series analysis, several studies have shown that CI algorithms have been effective at learning and forecasting, producing results suggesting that the markets are not perfectly efficient. From this we have to decide what the primary objectives of the study are and which algorithms can accommodate. The list of primary objectives is provided below:

1. Optimal - able to outperform the market benchmark,
2. Flexible - adapt to changing market conditions, and
3. Interpretable - surmise what the agent is doing and determine market conditions from agent structure

The first of the primary objectives is to ensure that the results are meaningful. Secondly, for a trading model to be profitable in a range of market conditions that model needs to be flexible. Rigid trading rules will not produce above average returns at all times, which is precisely why technical analysis is difficult. Thirdly, the model should be white box. The results from the analysis would be more meaningful if we could interpret what the agent has learned, and if we could surmise what type of market conditions are suitable for a particular agent. For example, can we determine if the market was trending or more volatile based on the agent's structure?

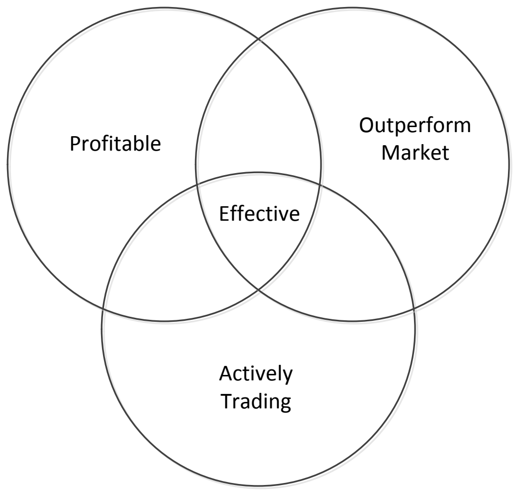

Figure 5.8: A Venn diagram showing that effective models are those that satisfy all the criterion of being profitable, outperforming the market and are actively trading.

Let us start with one of the most popular learning paradigms from CI for time-series analysis, Artificial Neural Networks (ANN), where studies have shown that they are arguably among the most robust [142] [6]. In the context of the three primary objectives we can determine that ANNs are able to outperform the market during training, that they are flexible but represent a black-box model, and that it would be difficult to extract domain knowledge from the topology and connection weights. Support Vector Machines have also become popular in the financial forecasting literature and offer a robust and flexible modelling approach, however, they also suffer from a lack of interpretability just as the ANNs. Evolutionary Computation (EC) is also an active area of research in financial forecasting and encompasses a variety of techniques from genetic algorithms (GA), genetic programming (GP), Artificial Immune Systems (AIS) and hybrid algorithms, to name a few. Once again, in the canonical use of these techniques we can easily accommodate the objectives of flexibility and optimality but the models will generally be black box. Moving back to traditional technical analysis, certain trading rules could be more effective in trending markets (moving averages) and others when the market is moving sideways (Bollinger Bands) and although it is possible to interpret these rules, they are, by construction, static.

With each of these techniques possessing weaknesses with respect to the primary objectives, it is a natural succession to entertain the combination of two or more of them. There has been documented success in combining population based optimization techniques with technical trading models, such as GAs with moving averages. This would entail determining the length of windows for calculating the moving averages via profitability based fitness functions. Another recent study by Butler et al. [13] combined Bollinger Bands with particle swarm optimization (PSO) to tune the parameters to current market conditions. The experiments implied that the effectiveness of the indicator could be enhanced beyond that of just using the default parameters. In the context of the primary objectives the hybrid models are the most suitable. Using an architecture from traditional technical analysis allows for interpretable models; additionally the benefit of flexibility from the CI algorithms is retained, and finally the comparability between models is possible as the technical trading rules have a finite set of attributes, which allows for comparisons in a relatively small n-dimensional space.

For this study the optimal trader for each market segment will be determined using adaptive Bollinger bands (ABB) [13], which are based on a technical indicator created by John Bollinger in the 1980's.

5.3.1 Adaptive Bollinger Bands

The ABBs were initially developed because, despite their popularity, the recent academic literature had shown Bollinger Bands (BB) to be ineffective [79] [80]. However, through PSO-based parameter fine tuning the indicator could be improved and outperform the market index under certain market conditions. The three main components of BBs are:

1. An N-day moving average (MA) for a price series $\{P_i\}$, which creates the middle band, equation 5.6,
$$MA_n(t) = \frac{\sum_{i=t-N+1}^{t} P_i}{N} \tag{5.6}$$
2. an upper band, which is the MA plus k times the standard deviation of the middle band, and
3. a lower band, which is the MA minus k times the standard deviation of the middle band.

The default settings for using BBs are a moving average window of 20 days and a value of k equal to 2 for both the upper and lower bands. When the price of the stock is trading above the upper band, it is considered to be overbought, and conversely, an asset which is trading under the lower band is oversold. The trading rules that can be generated from using this indicator are given by equations 5.7–5.8:

$$Buy : P_i(t-1) < BB_n^{low}(t-1) \& P_i(t) > BB_n^{low}(t) \tag{5.7}$$

$$Sell : P_i(t-1) > BB_n^{up}(t-1) \& P_i(t) < BB_n^{up}(t) \tag{5.8}$$

Essentially, the above rules state that a buy signal is initialized when the price (P_i) crosses the lower band from below, and a sell signal when the price crosses the upper band from above. Using the BBs in their canonical form, in both cases the trade can be closed out when the price crosses the middle band. As such, a trader will be taking long/short positions in the market; a long/short position is a trading technique which profits from increasing/decreasing asset prices.

To allow for efficient online optimization of the BBs we define two new forms of the traditional indicator, running and exponential BBs, that make use of estimates of the 1^{st} and 2^{nd} moments of the time series (i.e. μ and σ^2).

5.3.1.1 Running and Exponential Bollinger Bands

We define a BB as:

$$BB = MA_n \pm k \times \sigma(n_{period}) \tag{5.9}$$

where MA_n is an n-day moving average and σ is the standard deviation. Then a Running Bollinger Band that makes use of estimates of the 1^{st} and 2^{nd} moments is:

$$BB = A_n \pm k \times J_n(B_n - A_n^2)^{1/2} \tag{5.10}$$

where,

$$A_n = \frac{1}{n}\sum_{i=1}^{n} Y_i \quad , \quad B_n = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 \tag{5.11}$$

$$J_n = \frac{n}{n-1}^{1/2} \tag{5.12}$$

where the normalization factor J_n allows for an unbiased estimate of the σ and Y_i is i^{th} data point. From this, recursive updates of the BBs can be performed as follows:

$$A_n = \frac{1}{n}Y_n + \frac{n-1}{n}A_{n-1} \tag{5.13}$$

$$B_n = \frac{1}{n}Y_n^2 + \frac{n-1}{n}B_{n-1} \tag{5.14}$$

Table 5.6: The parameters that the PSO algorithm optimized. MA stands for moving average. The particles are the number of particles from each individual in the swarm allocated for that parameter.

Description	Particles
The value for calculating the upper/lower band.	2/2
Window size for the upper/lower band MA.	5/5
The type of ABB to use for upper/lower band.	1/1
The stop loss for short-sells/buys.	2/2

For the exponential form we define the BB on a time scale η^{-1}. Where incremental updates of the estimates are:

$$A_n = \eta Y_i + (1-\eta)A_{n-1} \tag{5.15}$$

$$B_n = \eta Y_i^2 + (1-\eta)B_{n-1} \tag{5.16}$$

and the normalization factor becomes:

$$J_n = \frac{1-\eta/2}{1-\eta} \tag{5.17}$$

This implementation of the ABBs was written in JAVA and optimizes eight parameters, as displayed in table 5.6. A result from [13] concluded that BBs are ineffective at generating profits when the market is trending. This shortcoming of the BBs was mainly due to the exiting of profitable trades prematurely. To counteract this consequence of using the middle band (the N day moving average) to initiate the closing out of a trade, this implementation uses trailing stop-losses to determine exit points. A trailing stop-loss is a popular trading technique that essentially allows a set amount to be lost from the maximum profit achieved.

An additional advantage to using BBs as the underlying technical analysis tool is that we are able to tap into a common heuristic used by active traders of identifying turning points in stock movements. The identification of an overbought or oversold security signals a correction and therefore a change in directional movement. However, choosing a turning point is very difficult as a trader will be taking positions that are contrary to the current market trend.

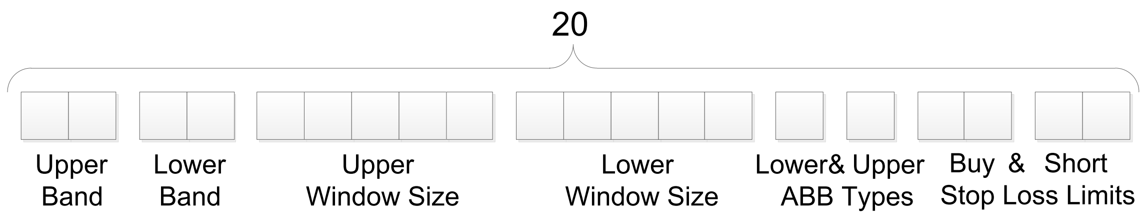

Figure 5.9: The mapping of a particle from the 20-dimensional solution space to the 8-dimensional parameters space for the ABBs.

5.3.2 Training Adaptive Bollinger Bands

The new versions of the Bollinger bands allow for efficient online optimization which is performed using particle swarm optimization (PSO). More specifically the type of PSO used is dynamic Heterogeneous Particle Swarm Optimization (dHPSO) as described in section 2.1.1. Each particle in the swarm will define a mapping from the n-dimensional solution space (particle space) to the m-dimensional parameter space, where m<n. In these experiments each particle will have 20 dimensions that map to the 8 dimensions of the ABB as displayed in table 5.6 and depicted in figure 5.9.

In addition to the PSO representation above the algorithm also requires a metric to gauge the performance of the individual particles in the swarm. In a genetic algorithm this is referred to as a fitness function and although PSO is not derived from the principles of evolution we will also refer to this performance metric as a fitness function. Several metrics have been proposed in the literatures that vary from general classifier outputs, such as accuracy and precision, to more application based such as profit and risk-adjusted returns. The previous literature on optimizing technical analysis (discussed in section 3.3.1) only considered profitability for assessing the fitness of an individual, however there are several documented cases of using accuracy (as discussed in chapter 3) and risk-adjusted returns [102] to train CI algorithms where these metrics have outperformed purely profit maximization approaches. As such, this section will perform a preliminary study to determine "what is the most appropriate fitness function for training ABBs given the three primary objectives listed above? The first is based on profit maximization that also takes into account transaction costs, equation 5.18:

$$fitness_i = \sum_{i=1}^{T} capital_t \times \frac{(P_{1,t} - P_{0,t})}{P_{0,t}} - (\tau \times capital_t) \tag{5.18}$$

where $fitness_i$ is the fitness of the i^{th} particle in the swarm, τ represents the transaction costs, and P_0 and P_1 are the entering and exiting price for the underlying asset. The profit for each trade is the rate of return multiplied by the capital invested minus the transaction cost which is also a function of the amount of capital invested. The next fitness function is the Sharpe ratio, a metric which considers returns and risk, shown in equation 5.19:

$$S = \frac{E[R - R_f]}{\sqrt{var[R - R_f]}} \tag{5.19}$$

where R is the return on the asset and R_f is a risk-free rate and E is the expected returns operator. Using the Sharpe ratio allows for particles which efficiently use risk to be assigned higher fitness with the intention that these models will perform better in the out-of-sample test periods. The third and final fitness function to be considered is based on accuracy and though a technical indicator is not a classifier we can base accuracy on the number on profitable trades divided by the total number of trades executed, equation 5.20

$$fitness_i = \frac{\#returns > 0}{\#returns > 0 \ + \ \#returns \leq 0} \tag{5.20}$$

where *returns* is a scalar representing the return of a single trade.

To investigate the preferred fitness functions we simulate a time series using the following:

$$\begin{aligned} p_t &= p_{t-1} + \beta_{t-1} + \kappa\epsilon_t \qquad (5.21) \\ \beta_t &= \alpha\beta_{t-1} + \nu_t \qquad (5.22) \end{aligned}$$

where α and κ are constants, and ϵ_t and ν_t are normal random variables with zero mean and unit variance. The simulated price process is:

$$z_t = exp\left(\frac{p_t}{R}\right) \tag{5.23}$$

where R is the range of p_t: $max(p_t) - min(p_t)$ over a simulation of 10,000 samples. Equations 5.21-5.23 define a random walk with short-term autoregressive trend properties as discussed in Moody et al. [102]. Using a realization from these equations the ABBs will be trained and tested based on a 50/50 split, which yields 5000 points for training and 5000 for testing. Due to the fact that a degree of randomness is associated with PSO, 100 runs will be performed for each fitness function and reported results will be averages over those runs. Reported are the percentage returns and number of trades for the training and testing periods, shown in table 5.7.

The results suggest that accuracy is not appropriate for training an ABB and across all periods and metrics it produced inferior results. This is not as surprising as it may seem given that accuracy is generally associated with classifiers and technical indicators are more closely aligned with reinforcement learning, in that their signals tend

Table 5.7: The average results from 100 runs for training and testing the ABBs using the various fitness functions.

	Train		Test	
Function	Return (%)	# of Trades	Return (%)	# of Trades
Profit	0.6844	93.14	0.0710	18.92
Sharpe	0.6798	95.79	-0.0355	15.6
Accuracy	0.6055	12.53	-0.0243	7.86

to be associated with delayed rather than immediate rewards. Thus, given that profit maximization and the Sharpe ratio have been implemented successfully in reinforcement learning it is not surprising that these metrics produced superior results. However, it is a straight forward decision that profit maximization is the preferred method as it produced higher returns (more likely to produce positive returns) and more trades (more likely to be actively trading) in the testing period. This result is in agreement with previous results published in Butler et al. [13]. Therefore in the cyclical effectiveness experiments to come the profit maximization fitness function will be used to train the ABBs.

5.3.3 Experiment Setup

The data used for testing cyclical effectiveness were the daily closing prices for the S&P 500 for a 10 year time period spanning 2001-2010. The first 5 years were allocated for training the ABBs with the remaining 5 years for testing. A benefit of using BBs (as well as other technical analysis techniques) is that no pre-processing of the data is required as the indicators do not make any assumptions of normality or stationarity.

To allow for a range of investment policies we analyze the optimal traders at different levels of granularity. Thus the experiments are conducted for an increasing number of data points within the sliding window. The use of the sliding window is the same as described in section 5.2.1. Table 5.8 displays the parameters and number of agents created for each of the experiment setups. To assess the effectiveness of the agents, the experiments are carried out with an initial starting capital of £1000.00 and a transaction rate (applied when entering and exiting the market) of 0.25%, i.e., a quarter of a percent of the amount of capital invested. We assume no transaction costs for investing in the risk-free rate (R_f) which is accrued daily and has AER of 2%. In this implementation the ABB fully invests all capital each day and whilst in a trade no other positions can be taken.

The parameters for the PSO algorithm have the same settings for each experiment and are displayed in table 5.9. In order to maximize the number of time-periods where an agent

Table 5.8: The parameters for the various experiment setups.

Case	Window	# of Agents	≈ time	# of test periods
1	125	1132	6 mths	1133
2	250	1007	1 yr	1008
3	500	757	2 yrs	758
4	1000	257	4 yrs	258

is identified that outperforms the market, the PSO algorithm will initially train for 100 epochs. If at that time an optimal agent is not found the algorithm is allowed to continue up to a maximum of 1000 epochs. The dimensions are a sum of the number of particles in each position vector allocated to each of the ABB parameters.

Table 5.9: The parameter settings for the PSO algorithm.

Parameter	Value	Parameter	Value
initial epochs	100	max epochs	1000
c_1	2	c_2	2
Particles	30	Dimensions	20

5.3.4 Cyclical Effectiveness Results

The results presented in this section are the average performance results for the ABBs over all test periods. There are three metrics considered: (1) the average number of ABBs that outperform the market (OM), (2) the average number of ABBs that produce a positive return (PR), and (3) the average number of ABBs that are effective (EF), where effective implies, from the above definition, that the ABB was profitable, active and outperformed the market index. The following sections will present tables and box plots of the results as well as a discussion.

5.3.4.1 Case 1 through Case 4

The results from training and testing the ABBs using sliding windows of 125, 250, 500 and 1000 days are presented in tables 5.10-5.13 and figures 5.11-5.14 are box plots of the metric distributions.

5.3.5 Cyclical Effectiveness Conclusions

The results presented in tables 5.10 through 5.13 reveal that at each level of granularity there were ABBs that were effective in the out-of-sample test data. Figure 5.10 plots the

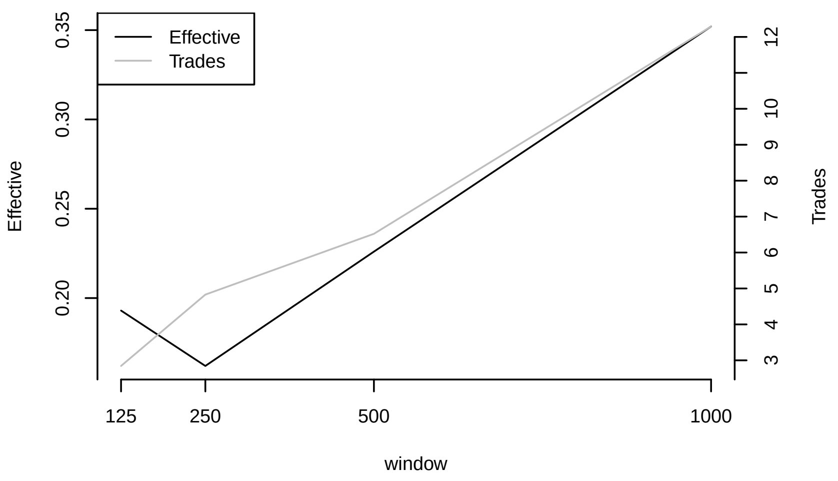

Figure 5.10: Plots of the average number of trades for effective ABBs and the average percentage of ABBs that were effective.

Table 5.10: The results from training and testing the ABBs with a sliding window of 125 days. ??$_{\bar{tr}}$ stands for the average number of trades by the ABBs that satisfied the concerned metric.

	OM	$OM_{\bar{tr}}$	PR	$PR_{\bar{tr}}$	EF	$EF_{\bar{tr}}$
Average	0.444	2.841	0.463	2.409	0.193	2.844
Min	0.116	0.000	0.000	0.000	0.000	1.000
Max	0.627	21.270	1.000	19.723	0.462	19.914
Median	0.449	1.941	0.434	1.549	0.192	1.984

Table 5.11: The results from training and testing the ABBs with a sliding window of 250 days. ??$_{\bar{tr}}$ stands for the average number of trades by the ABBs that satisfied the concerned metric.

	OM	$OM_{\bar{tr}}$	PR	$PR_{\bar{tr}}$	EF	$EF_{\bar{tr}}$
Average	0.438	5.128	0.371	4.588	0.162	4.829
Min	0.056	0.000	0.013	0.000	0.000	1.000
Max	0.678	41.695	1.000	38.571	0.422	38.147
Median	0.438	4.015	0.343	3.669	0.157	3.765

Table 5.12: The results from training and testing the ABBs with a sliding window of 500 days. ??$_{\bar{tr}}$ stands for the average number of trades by the ABBs that satisfied the concerned metric.

	OM	$OM_{\bar{tr}}$	PR	$PR_{\bar{tr}}$	EF	$EF_{\bar{tr}}$
Average	0.601	8.558	0.311	6.689	0.226	6.521
Min	0.001	0.062	0.000	0.065	0.000	1.000
Max	0.959	92.201	0.997	92.028	0.858	92.028
median	0.617	7.258	0.289	5.463	0.219	5.123

average number of trades and the percentage of effective ABBs against the window size. We see an increase in the percentage of the ABBs that are effective as the window size increases. This is due to overfitting, where the ABBs tuned to smaller amounts of data

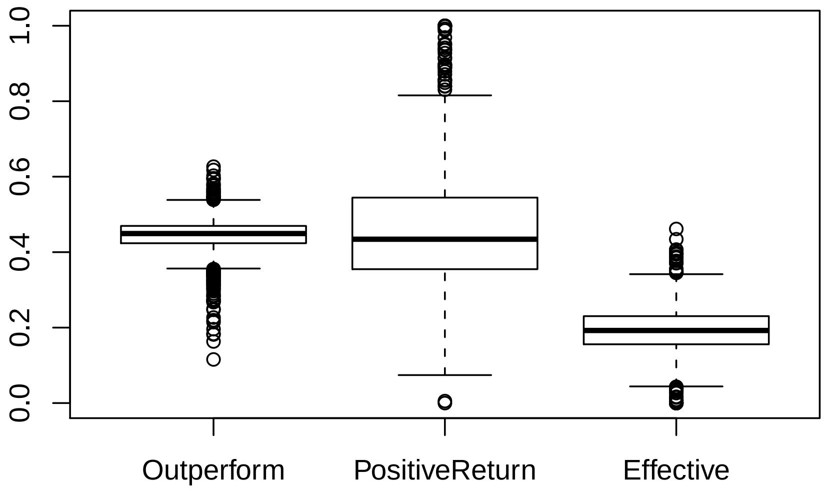

Figure 5.11: Box plots of the distributions of the Outperform the Market (OM), Positive Return (PR) and EFfective (EF) metrics for case 1.

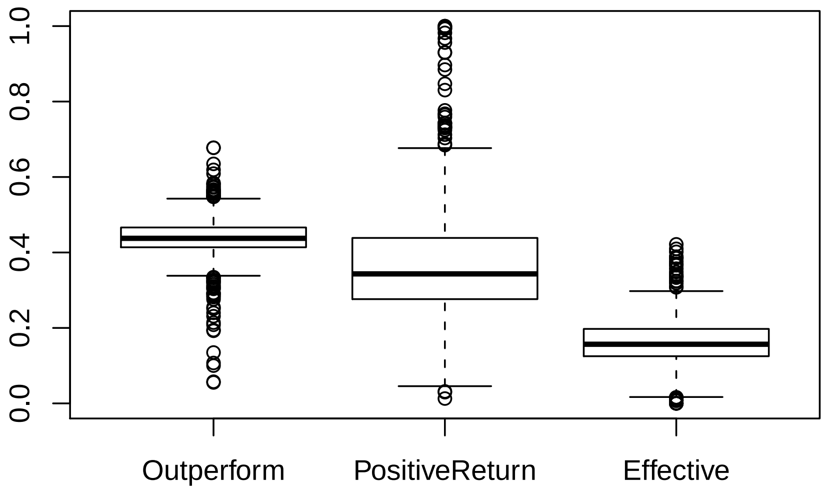

Figure 5.12: Box plots of the distributions of the Outperform the Market (OM), Positive Return (PR) and EFfective (EF) metrics for case 2.

Table 5.13: The results from training and testing the ABBs with a sliding window of 1000 days. ??$_{\bar{tr}}$ stands for the average number of trades by the ABBs that satisfied the concerned metric.

	OM	$OM_{\bar{tr}}$	PR	$PR_{\bar{tr}}$	EF	$EF_{\bar{tr}}$
Average	0.611	12.108	0.391	11.871	0.352	12.289
Min	0.000	0.000	0.000	0.000	0.000	1.000
Max	1.000	51.171	1.000	48.952	1.000	48.952
Median	0.624	13.593	0.399	13.296	0.329	13.589

are more likely to become over-fit and therefore not generalize as well. We also observe a monotonic increase in the average number of trades executed by the ABBs as the window size increases. From the OM and PR metrics we can observe that ABBs are not always

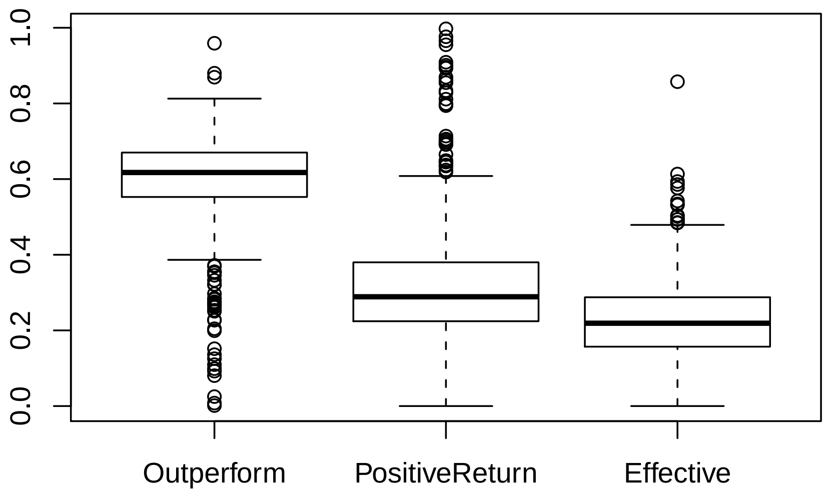

Figure 5.13: Box plots of the distributions of the Outperform the Market (OM), Positive Return (PR) and EFfective (EF) metrics for case 3.

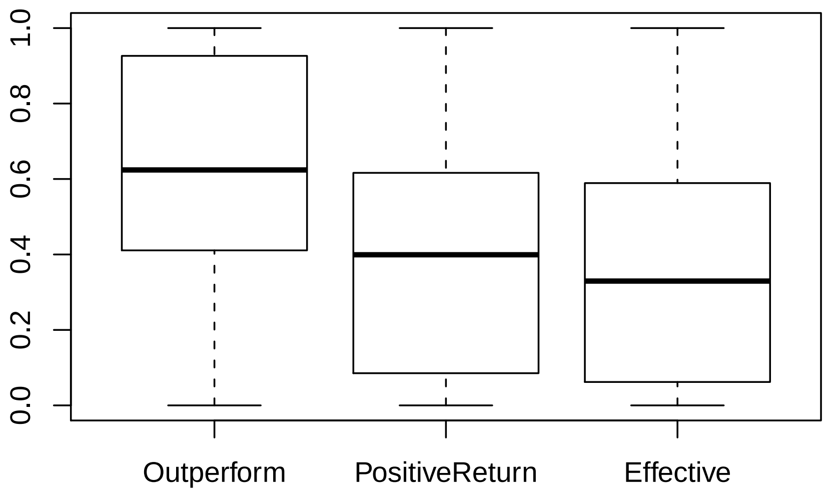

Figure 5.14: Box plots of the distributions of the Outperform the Market (OM), Positive Return (PR) and EFfective (EF) metrics for case 4.

active in the market and that the parameters which are optimal in one time period can lead to a technical indicator that does not execute any trades when the market environment is quite different. This is partly the reason for higher percentages of the ABBs producing positive returns but not being able to outperform the market. On average the ABBs made a trade every 3 to 4 months when they were effective, though there were instances where the ABBs were effective and extremely active in executing trades. In case 3 where the window size was 500 days we observe a maximum average trading activity of 92.028, which translates to about 4 trades a month. This is quite active for a technical indicator that is identifying turning points in stocks price behaviour.

The box plots reveal that none of the metric distributions are normal (all rejected the null of normal from the Jarque-Bera test [58]) and that for the majority of the plots there are several outliers beyond the 1^{st} and 3^{rd} quartiles. With the exception of case 4 (1000 day window) all of the boxes are quite small indicating that 50% of the data is within close range of the median. This narrow interquartile range coincides with the large amount of outliers or suspected outliers.

5.4 Chapter Summary

In this chapter we presented work which investigated the effect of certain implications of the AMH on computational intelligence algorithms. Commencing with a study into variable efficiency and if the presence of non-linear dependence in a financial time series offered any advantages for supervised learning algorithms. The results demonstrated that if non-linear dependence could be reliably detected then a statistically significant increase in classification accuracy could be expected. Secondly we considered variable stationarity and how it impacted the optimal pre-processing step for training ANNs for level estimation. From the results we concluded that the optimal pre-processing step was a function of stationarity, forecast time horizon and persistence of the stationary process. This conclusion also lead to the rejection of a previously published result, that concluded that overfitting an ANN was the optimal way for modelling non-stationary data. Our results, which are also theoretically sound, demonstrated that differencing the data was the optimal pre-processing step for short-term forecasting. Finally the effect of the waxing and waning of investment strategies was investigated. This section included the proposal and development of a novel modelling tool called an adaptive Bollinger band, which was used for testing cyclical effectiveness. The results demonstrated that ABBs exhibited cyclical effectiveness and that models developed in one time period were effective again in future time periods. This concludes the research on the implications of the AMH and in the next chapter we explore innovative approaches to accommodate these implications and the complexity of the market in general.

Chapter 6

Innovations in Technical Analysis

Technical analysis is based on the premise that history tends to repeat itself and therefore past patterns can be used for predictive purposes [82]. This section introduces a novel forecasting algorithm that is a blend of micro and macro modelling perspectives when using computational intelligence techniques. The micro component concerns the fine-tuning of technical indicators with population based optimization algorithms. This entails learning a set of parameters that optimize some economically desirable fitness function so as to create a dynamic signal processor which adapts to changing market environments. The macro component concerns combining the heterogeneous set of signals produced from a population of optimized technical indicators. The combined signal is loosely based on a Learning Classifier System (LCS) framework that combines population based optimization and reinforcement learning (RL). This research is motivated by the results in chapter 5.3 that demonstrated the cyclical effectiveness of trading models derived from computational intelligence. As discussed, the non-stationary nature of the stock market denotes a signal which has moments (such as the μ and σ^2) that are not constant in time. This could imply that trading models will have to continually adapt to changing market conditions to remain profitable. However, cyclical effectiveness implies that the performance of trading models constructed from historical information will wax and wane with the evolving market conditions. Thus, experienced trading models will still contain useful information for forecasting. These two properties are not necessarily in contradiction but they do highlight the need for adaptation and creation of new models, while synchronously being able to consult others which were previously effective. What follows in this chapter is the proposal, implementation and testing of an algorithm called LATIS (Learning Adaptive Technical Indicator System) which can account for this complexity in the financial markets, whilst remaining true to the canonical form of technical indicators. The name has a double meaning, it is not only an acronym but since the framework is based on several interconnected woven components it is also *lattice* like in structure.

6.1 How to generate signals?

The previous work discussed in section 3.3.1 highlighted that the majority of the research in the overlap of computational intelligence and technical analysis concerned supervised learning for classification of price movements. In the studies discussed, the technical indicators were used as input attributes to the various algorithms and therefore the information contained in the technical indicators was being utilized for classification. This of course is not how technical indicators are intended to be used and their straight forward interpretation is also compromised. Although many of the approaches have been "white box" this does not mean that an intuition as to **why signals are being created** can be gained by examining those models. However, the mechanics behind signal creation in technical indicators is well understood by technical analysts and these approaches have endured due to their intuitive nature. Thus it would be beneficial to create an algorithm that has the sophistication of computational intelligence but also the intuitive and familiar interpretation of technical indicators.

The results from Butler et al. [13] on optimizing a single Bollinger band, and the results from the previous chapter on cyclical effectiveness, demonstrate that technical indicators can be successfully combined with population based optimization algorithms and that models created in one time-period will be effective again in another. However the questions of when a technical indicator will become effective and how long it will be effective for, still remain. These questions contribute to the motivating factors behind the development of LATIS. Since it cannot be known a priori when a model will be effective again and for how long, there needs to be a mechanism which can learn this. If we consider the answers to these questions to be time varying meta parameters of the indictors then an appropriate algorithm can learn them from a set of training data. The LATIS framework allows for online estimation of these parameters using reinforcement learning. Therefore signals generated from a population of optimized technical indicators can be evaluated by a meta agent where actions of the agent are based on a set of signals and meta information associated with the indicators producing the signals. In summary then, the indicators are not used as inputs to a classifier but are preserved in their original form and the, highly desirable, combined signal approach is achieved through population based optimization and reinforcement learning.

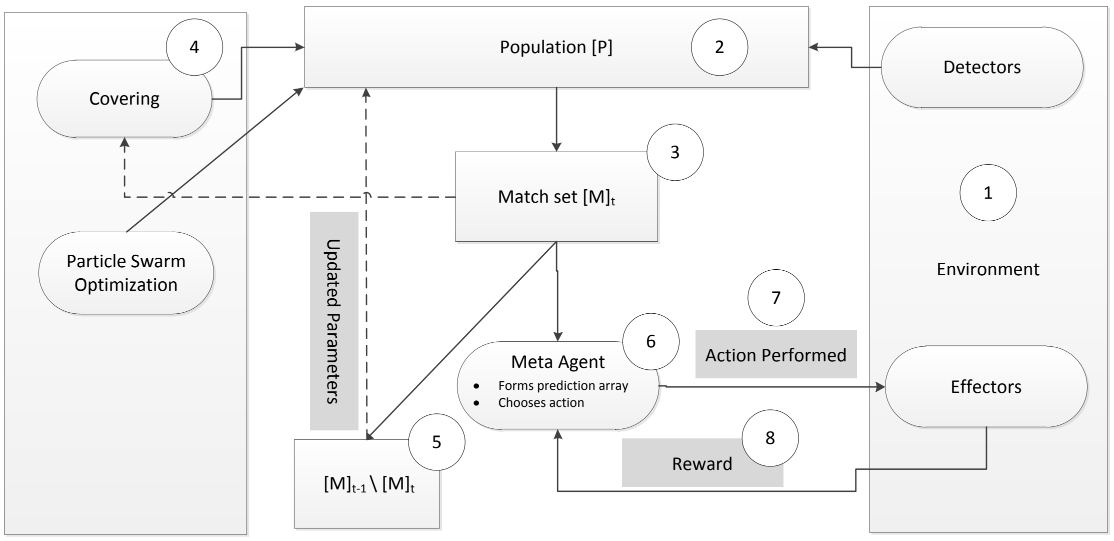

Figure 6.1: LATIS framework - the values 1-8 indicate the typical steps included in a single learning iteration of the algorithm. Dashed lines indicate steps that may not occur every iteration.

6.2 Description of LATIS

An overview of a typical iteration of the LATIS algorithm is supplied in figure 6.1. We observe from figure 6.1 that LATIS interacts with its environment through a set of detectors, for receiving signals, and a set of effectors, which allow LATIS to make changes or alter its environment. The eight steps comprise the three main components of the LATIS system: performance, reinforcement and discovery, which will all be described in detail in the following sections. In what follows we denote the following parameters and data structures. LATIS contains a population, [P], of technical indicators of size N, where the initial population, $[P]_0$ is created using a set of training data and an optimization technique. The indicators in [P] are referred to as individuals, I, where technical indicators in the population will be denoted as $I \in [P]$. Associated with each $I \in [P]$ are three meta parameters, *reputation*, *effectiveness threshold* and *return* (R, κ and θ respectively) that are used by the meta agent to choose actions. R determines how much influence an individual has on the meta agent and is an expectation of how long an individual will provide reliable information, κ is the performance required to be introduced into [M] (the match set) and θ is the expected return while included in [M].

At each time step, t, LATIS will choose an action, a, from a set of possible actions, A, to perform. The typical actions available to LATIS are long, short or risk-free positions in the environment. A long position requires that the financial asset under consideration is bought at market price (the current price in the market). For a short position the financial asset is short-sold (sold before acquiring), and the risk-free position requires that LATIS invests in a risk-free alternative. To reiterate, a long position means an investment increases in value when the asset **increases** in value and a short position means the investment increases in value as the asset **decreases** in value.

The flow of information through the system is depicted in figure 6.2, where the forward propagation of a signal is shown from the environment through to the generation of an action to be performed. At each time step a signal, S_t, is generated by the environment and serves, as depicted, a dual purpose. Firstly, the signal is transmitted to the population, $[\mathrm{P}]_t$, which contains technical indicators. The time subscript on [P] denotes that the components of [P] are time-varying. The signal is then used by $[\mathrm{P}]_t$ to form a match set, $[\mathrm{M}]_t$, which is a combination of a subset of the match set a time $t-1$ and the additions to $[\mathrm{M}]_t$ from $[\mathrm{P}]_t$. The match set then transmits the *opinions* of the of $I \in [\mathrm{M}]$ to the meta agent which are then combined with their respective meta parameters to form an action. This action is then performed in the environment by the meta agent. The opinions of the $I \in [\mathrm{M}]$ are their current positions in the environment. For example, if an I is currently in a long position, it would transmit a "buy" opinion to the meta agent and if it was in a short position it would transmit a "sell" opinion.

Secondly, the same signal, S_t is also transmitted back to the meta agent as a reward for its previous action. The reward is a function of the signal and the action performed at time $t-1$, where for the reward calculation, the actions long, short and risk-free are repesented as {1, -1 and 0} respectively. The reward calculation is as follows:

$$reward \leftarrow \begin{cases} (S_t - S_{t-1})/S_{t-1} * a_{t-1} & \text{if } sgn(a_{t-1}) \neq 0 \\ r_{rf} & \text{otherwise} \end{cases} \tag{6.1}$$

where a_{t-1} is the most recent action, $sgn()$ is the signum function which extracts the sign of a real number and r_{rf} is the risk-free rate of return. For example, if the meta agent's action at $t-1$ was to take a long position and $S_t > S_{t-1}$, then the agent would receive a positive reward.

It is worth noting that the creation of $[P]_0$ does not have to be data driven and a population of technical indicators with random parameter settings could be used. However, since the implication of cyclical effectiveness was demonstrated to be valid for CI models, it is desirable to initialize $[P]_0$ with some training data representative of the target environment. In general, the environment will be represented by time series data from a financial asset, such as a stock or market index, and the signals generated are recent prices sampled from that environment. The exact details of how the technical indicators

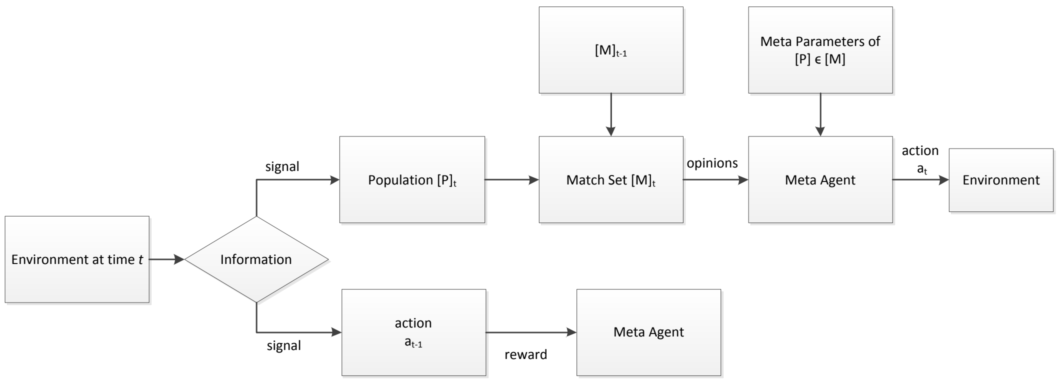

Figure 6.2: The propagation of information from the environment through the LATIS framework. Each signal generated from the environment serves two purposes, the first to create the match set and generate a signal, and secondly, to provide a reward to the meta agent for the most recent action performed in the environment.

are trained for inclusion in $[P]_0$ is up to the data modeller but an example approach will be supplied in section 6.4, where we discuss a specific implementation of the LATIS framework.

The two questions which motivated the LATIS framework, (i) as to when a technical indicator will be effective and (ii) for how long? are answered by the meta parameters of the $I \in$ [P]. Firstly, to know when a technical indictor (not currently included in [M]) is going to be effective its recent performance is gauged by the meta agent against the original performance from training. The performance required is determined by the κ meta parameter, which is a scalar ≥ 0. If the current performance satisfies this minimum criterion then it is included in [M]. Secondly, once an individual is a member of [M], its reputation represents an expectation of how long it will remain effective after inclusion. Reputation thus provides an estimate of how reliable the signal is. Assuming that we have a dynamic environment, we would want these meta parameters to be adaptive, and thus these values are determined and updated in an online learning fashion. Once the algorithm goes online the R, κ and θ meta parameters are updated each time an individual participated in [M] (this process is described in section 6.2.2).

6.2.1 Performance Component

Given an input at time t from the environment, a match set $[M]_t$ is formed from individuals in $[P]_t$ that satisfy the performance criteria set by the meta agent. This is another difference between LATIS and an LCS (Learning Classifier System) where in an LCS the classifiers from [P] that form [M] would have rules with antecedents that matched the signal. From [M] a *prediction array* is constructed which determines an action $a_i \in A$ (the set of possible actions) to be performed by the system. To construct the *prediction array* the system forms a *system prediction* $P(a_i)$ for each a_i represented in [M]. The calculation of $P(a_i)$ is based on a reputation weighted score of the individuals advocating a_i. This is similar to the fitness-weighted method described by Wilson [136]. Once the prediction array is constructed, the a_i to be selected can be based on a variety of methods. For example, deterministic action selection, which selects the a_i with the largest $P(a_i)$, other possibilities could be roulette wheel or random. The three possible actions a typical LATIS implementation can perform are: long, short and risk-free. Once an action is chosen the meta agent performs it and receives a reward from the environment (see equation 6.1), which is used to update the trading position of the meta agent. The equation for calculating $P(a_i)$ is:

$$P(a_i) = \left[\underbrace{\sum_{j=0}^{m} r_j \left(\frac{Rem_j}{\sum_{j=0}^{m} Rem_j} \right)}_{reputation-weighted\ return} \times \underbrace{\frac{\psi_j}{\psi_{avg}}}_{relative\ accuracy} \right] \tag{6.2}$$

where the first component of the equation is a reputation weighted return. m is the number of individuals advocating the action a_i in [M], r_j is the current return of j^{th} individual and Rem_j is the *remainder* or difference between an individuals reputation and its current duration, D_j, in [M], so:

$$Rem_j \leftarrow \begin{cases} R_j - D_j & \text{if } R_j \text{ - } D_j > 1 \\ 1 & \text{otherwise} \end{cases} \tag{6.3}$$

Reducing the reputation by the current duration takes into account the confidence the system has for a particular individual given that it will only be effective for a given duration based on cyclical effectiveness . The second component is an accuracy multiplier, where ψ_j (equation 5.20) is the directional accuracy of the j^{th} individual and ψ_{avg} is the average accuracy in [M]. The time-period used to determine the directional accuracy is the same time period for determining inclusion in [M]. The accuracy multiplier allows for an extra dimension to the system prediction that considers accuracy as well as return.

To aid in the explanation of this step, figure 6.3 shows the interaction between the meta agent and [M], where the action selection mechanism is based on information transmitted to the meta agent from [M] and calculated using equation 6.2. The individuals (I_j) in [M] transmit to the meta agent their action, reputation (R) and current return (r) to be used to calculate the system predictions.

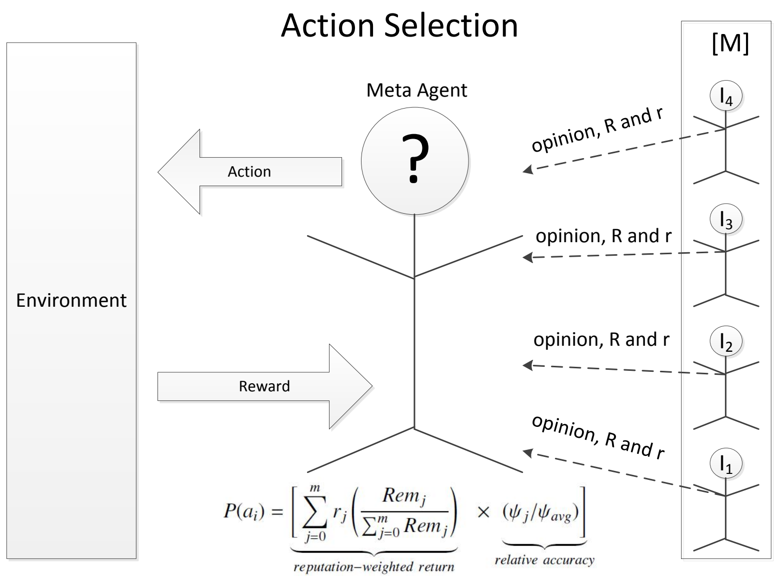
Action Selection
[M]
Meta Agent
?
Action
Environment
Reward
opinion, R and r
opinion, R and r
opinion, R and r
opinion, R and r
I_4
I_3
I_2
I_1
$P(a_i) = \Big[\underbrace{\sum_{j=0}^{m} r_j \Big(\frac{Rem_j}{\sum_{j=0}^{m} Rem_j} \Big)}_{reputation-weighted\ return} \times \underbrace{(\psi_j / \psi_{avg}) \Big]}_{relative\ accuracy}$

Figure 6.3: A diagram of the action selection mechanism of the meta agent. Showing the information that is supplied to the meta agent from [M] and how the rewards from the system are feed back into the meta agent directly.

6.2.2 Reinforcement Component

The reinforcement component of LATIS updates the meta parameters R, κ and θ of the individuals in step 5 of figure 6.1. The box label, $[M]_{t-1} \setminus [M]_t$, denotes the relative complement of $[M]_t$ in $[M]_{t-1}$ which represents the individuals that no longer satisfy the criteria for being included in [M]. The criteria for remaining in [M] is based on the following, where the j^{th} individual, referred to as I_j, may remain in [M] while either:

$$I_j \in [M] \ \text{if} \begin{cases} R_j - D_j > 0, & \text{or} \\ er_j > 0 & \text{and} \\ opinion_j \neq \text{risk-free} & \end{cases} \tag{6.4}$$

where er_j is the excess return of the j^{th} individual since R_j - D_j = 0. Therefore an individual may remain in [M] as long as it remains profitable and is invested in the market. This allows for trades that are useful for longer than the reputation suggests to still be considered by the meta agent. er_j is an excess return so it is calculated as:

$$er_j = r_j^{a..b} - r_j^{b+1..t} \tag{6.5}$$

where r_j is the return of the j^{th} individual during inclusion in [M], a is the time of inclusion in [M], b is the time when R_j - D_j = 0 and t is the current time. Thus er_j is only based on the time after reputation indicates the individual is no longer effective.

If reputation is supposed to represent how long I_j will be effective and we expect this to be time-varying, then we would want to have updates that incorporate recent performance. When reputation is too large and the signals are not as reliable, we want to be able to reduce reputation and when I_j is remaining effective beyond its' reputation value, we want to be able to increase it. Therefore, a learning technique which allows for online learning and the incremental aquistion of new information is appropriate. This type of learning can be acheived using the Widrow-Hoff delta rule, which is commonly used in neural networks and learning classifier systems. Thus, the update to reputation using the learning parameter β is as follows:

$$R_j \leftarrow \begin{cases} R_j + \beta(D_j - R_j) & \text{if } D_j > R_j \\ R_j + (r_j^{a..b} - \theta_j)/\theta_j, & \text{otherwise} \end{cases} \tag{6.6}$$

where all quantities are as previously defined. This equation allows the reputation of I_j to expand and contract in reaction to inclusion in [M]. If $D_j > R_j$ then the update is positive and is based on the difference between the two. If I_j exited [M] when $R_j = 0$, then the return ($r_j^{a..b}$) may have been lower than expected and the difference between the realized return ($r_j^{a..b}$) and the expected return (θ_j) is used to update reputation with a lower value. Since the returns will typically be quite small in comparison to R, there is no need for a discount factor.

The minimum criterion for the initial inclusion of I_j in [M] (κ) is intended to represent when I_j is effective and therefore would be producing reliable signals. Once again if we assume this parameter to be time-varying or at the very least to be unknown a priori we will want an update equation that maximizes the time I_j is in [M], that is, we want to maximize the actionable information in I_j. To facilitate this objective then, κ should be increased when I_j's performance declines and increased when I_j is becoming more effective. Thus an update for κ, based on the change in R, is:

$$\kappa \leftarrow \kappa - log(|\Delta R_j|)/(1/log(|\Delta R_j|) + (100 * sgn(\Delta R_j))) \tag{6.7}$$

where ΔR_j is the change is reputation based on equation 6.6 and sgn is the signum function, as before, which extracts the sign of a real number. This relationship allows for κ to decrease when R is increasing and vice-versa. The inclusion of 100 in the denominator is to control the size of the jumps in κ as ΔR_j increases. This relationship is shown in figure 6.4 where the size of the changes in κ tappers off as $|\Delta R_j|$ increases. When R_j is decreasing it indicates that I_j was added at an inappropriate time and therefore a closer

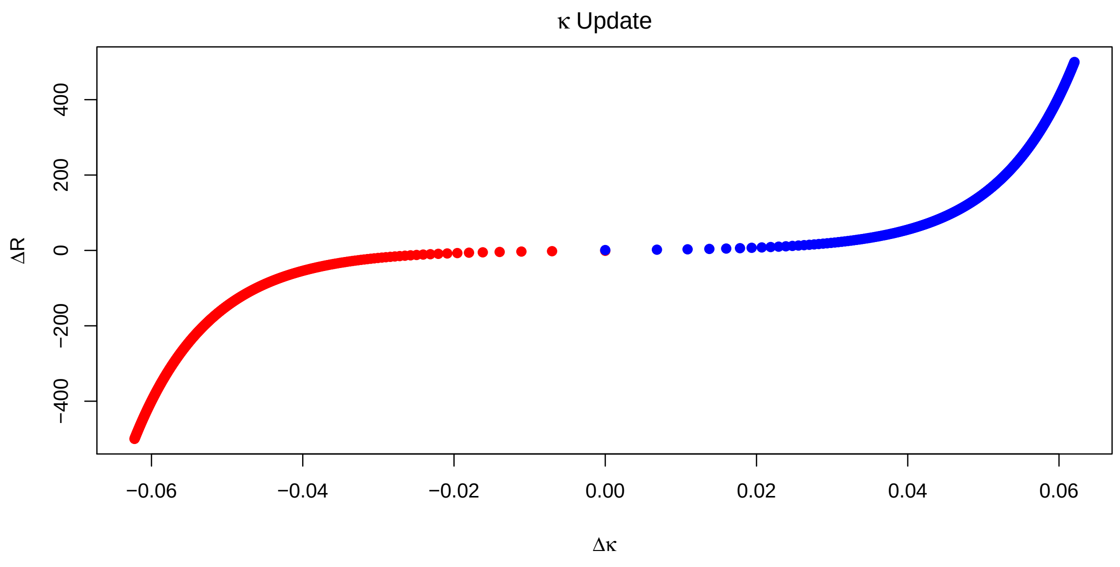

Figure 6.4: A plot of the change in κ for increasing values of ΔR_j, where the acceleration of the change in κ tappers off as ΔR_j approaches extreme values. Negative values are shown in red and positive values are blue.

match is desired in the future. It is worth mentioning, at this time, that there is not a maximum value for κ and values > 1 represent a minimum criteria that is above what was achieved by I_j in training. This of course could indicate the I_j is not very useful and in section 6.2.4 we will see how the meta parameters are used to eliminate individuals from [P].

The final meta parameter to be updated in θ_j which is the expectation of the return of I_j in any given inclusion in [M]. This parameter is updated after each participation in [M] by the following:

$$\theta \leftarrow \theta + \alpha(r_j^{a..b} - \theta) \tag{6.8}$$

where α is the learning rate or discount factor and all other quantities are as previously defined. Equation 6.8 allows θ to increase/decrease when returns are increasing/decreasing in [M] and the use of the learning rate α ensures θ is not overly sensitive to the most recent returns.

6.2.3 Discovery Component

The third and final main component of LATIS is discovery which is used to create new individuals for [P]. Unlike an LCS, in LATIS the population based optimization algorithm does not interact with [M] and only has a connection with [P]. In LATIS the optimization algorithm used is PSO and its two main duties are the following:

1. Create the initial population, $[P]_0$, based on a set of training data supplied from the environment.

2. Introduce new individuals to $[P]_{>0}$ (called the covering mechanism, step 4) when:

   (a) $|[M]| < [M]_{min}$ (the minimum size of the set [M]), or

   (b) when $I \in$ [P] no longer meet the minimum criteria to be a member of [P].

As stated, how the initial population is created is a decision for the data modeller but in the following section an example of this process will be provided. However in general the following is required:

1. An optimization algorithm for fitting the parameters of the technical indicators,

2. A set of technical indicators that have parameters which can be fitted to a set of training data,

3. A set of training data which is representative of the environment being modelled, and

4. A $FitnessFunction$ - a performance function based on some economically desirable metric.

The training data can be partitioned using a sliding window (discussed in previous chapters) and then for each window a version of each technical indicator can be created and added to [P] up to size N, which is the maximum size of the population. Other selection methods can be used for determining which indicator to optimize and the size of the sliding window to partition the data.

The covering mechanism (step 4) is meant to update LATIS with new information from the environment and also in recognition that not all models created in training are going to be useful or may not be useful forever. Removing individuals from [P] that are no longer useful and replacing them with new models from more recent data allows LATIS to incorporate current information. Thus the covering mechanism facilitates the objective of being able to synchronously consult previously found models and models developed in the current environment. If [P] is at its maximum capacity then any additions will have to be accompanied by a deletion from [P]. The determination of removing from [P] is covered in the next section.

6.2.4 Removing from [P]

The removal of individuals from [P] is a sensitive operation. This is because we want to protect the $I \in$ [P] which will be useful again and remove those which will not or will be detrimental to the performance of the meta agent. All of the information required to make this decision is contained in the meta parameters of the individuals. So, if any of the following criteria is satisfied then I_j should be removed from [P] (i.e. $I_j \notin$ [P]):

1. $R_j < 0$ (reputation is negative), or
2. $\theta < 0$ (expected return is negative).

The number of $I \in$ [P] need not be constant. When an individual is selected to be removed, it is not necessary to replace them immediately and the place can be held until the next covering step has be initiated. If I_j has a negative reputation then it has consistently produced lower returns then expected to the point that those returns have been negative. Alternatively, I_j could still have a positive reputation but a negative expected return, which would also make I_j undesirable. This would happen if the initial reputation was estimated to be quite large and only a limited number of inclusions in [M] have taken place. This leads to the final consideration for removing from [P] and that is the frequency at which an individual participates in [M]. If an individual never or rarely is included in [M] then it is not contributing to the meta agent and therefore should be replaced. To facilitate this, a count is maintained for each individual in [P] that indicates the number of times it has been included in [M]. If the covering step produces a new individual for [P] and [P] is at maximum capacity then the following equation can be used to determine the I to be removed:

$$Remove_j = \frac{\theta_j}{R_j} * (TC_j - MC_j) \tag{6.9}$$

where TC_j is a count of the number of time periods I_j has existed, MC_j is the count of the number of times I_j has participated in [M] and $Remove_j$ is a gauge of the contribution of I_j to the meta agent. Where higher scores for $Remove_j$ are more desirable. The use of TC_j is to accommodate the newly created I in [P] that did not have the same number of opportunities to participate in [M]. This removes the bias from LATIS of removing newly created I. Dividing the expected return by the reputation provides the daily contribution for being a member of [M] and then by multiplying this number by the TC_j-MC_j gives an estimate of how useful I_j is to the meta agent.

6.2.5 Adding to [M]

An integral part of forming the system predictions, $P(a_i)$, which are used to select an action, a_i, is the formation of [M]. In this section we provide more detail on how to use κ to select individuals for [M]. κ_j defines the performance criteria for I_j that is intended to signify when I_j is effective in the *cyclical effectiveness* sense of the word. The size of the time period, that the performance of $I_j \in$ [P] is calculated for, is a decision for the data modeller. Since we are evaluating current performance, the time-period should end with the most recent observation. However, there is not a generally preferred value for this variable but in the next section an example for choosing it is provided.

We propose two methods for selecting individuals for [M]. The first is based on performance and since these are trading models the appropriate metrics to consider are return, risk and trading activity. LATIS is meant to be active in the market, so the minimum criteria for trading is simply that I_j has made at least one trade in the period being considered. For the metrics of risk and return the κ_j meta parameter can be used. The metric for measuring risk is the Sharpe ratio as defined in equation 5.19. If we require that both metrics satisfy the κ criteria then we have a corresponding region in 2-d feature value space. In figure 6.5 we have a representation of the feature value space and the region that I_j must occupy to be introduced in to [M]. We do not necessarily want to penalize I_j for achieving a superior performance over the training period and therefore κ does not define a minimum criteria but a region around the training performance, denoted the κ-region, as indicated by the red ellipse in figure 6.5.

The second method is based on the structure of the individuals in [P]. Where the distance between models is determined in parameter space and κ forms a multi-dimensional region around the training structure. For the time-period currently under consideration a technical indicator is fitted, denoted the reference model (M_{ref}), and the resulting structure is compared to the κ-regions of each $I \notin$ [M]. Each I_j that the M_{ref} occupies the κ_j-region is added to [M].

6.2.5.1 Calculating the κ-region

The κ-regions defined above will be either 2 or 3 dimensional spaces depending on the feature space and the technical indicator being considered. We will describe the steps for calculating the κ-region in 2-d using figure 6.6 as a reference. Given a training point and a κ value the corresponding region is as follows:

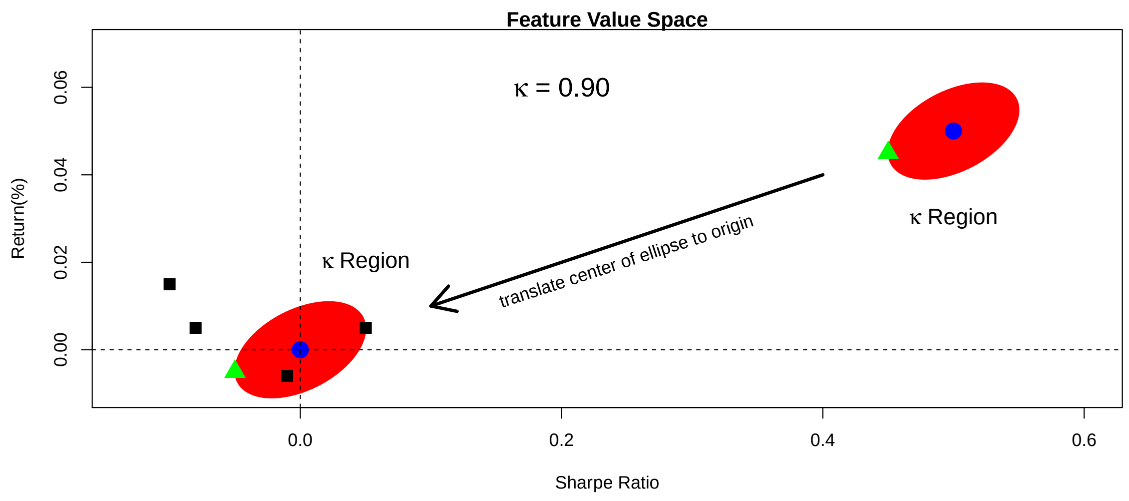

Figure 6.5: A plot of the κ = 0.90 region in 2-D feature space for the risk and return performance metrics. The area covered by the red circle represents the region that I_j must occupy to be introduced in to [M]. To establish that I_j occupies the κ-region the ellipse is translated to the origin and the standard equation of an ellipse (equation 6.11) is used. The squares (black) represent test points, the triangles (green) are the points determined by κ and the circles (blue) are the centres of the ellipses which is the training performance.

1. calculate the length a as $a = (\Delta X^2 + \Delta Y^2)^{-1/2}$
2. calculate the angle A as $A = \arcsin(\Delta Y^2/a)*(180/\pi)$
3. angle A = angle D
4. calculate the length b as $b = \Delta Y^2/\sin(\text{D})$

The values of a and b form the major and minor axes of the ellipse. Equipped with this information the training point at the centre of the ellipse (red point) can be translated to the origin (as shown in figure 6.5) where the standard equation for describing an ellipse:

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} = 1 \qquad (6.10)$$

can be used to determine if a point lies within the κ-region. Substituting the values of X and Y with any point in the 2-d feature value space will determine if the

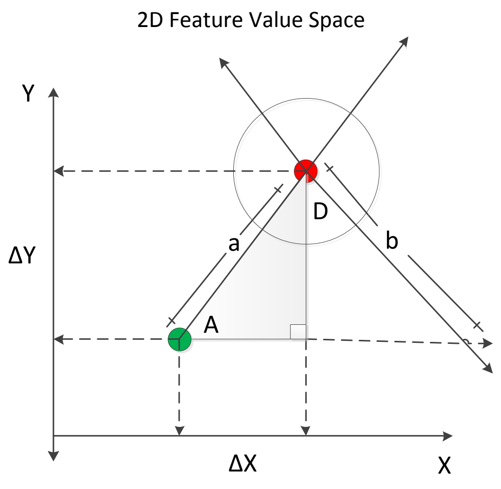

Figure 6.6: An example figure for the κ-region calculaiton.

point lies within the κ-region, where values < 1 indicate points within the region and values > 1 indicate points that are not.

This procedure can easily be extended to 3-d space where the equation for determining if a point lies within the κ-region is:

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} + \frac{Z^2}{c^2} = 1 \tag{6.11}$$

and the additional axis c is determined by simplifying the 3-d (x, y, z) space into 2-d along the (x, z) axis and then the procedure is executed as normal. In the above discussion the x-axis is considered the largest and therefore the major axis for all calculations.

6.3 Parameter List

The foregoing description of LATIS has introduced various parameters. They are summarized below in table 6.2 and some typical values will be seen in the implementation.

Table 6.1: A list of LATIS parameters.

Parameter	Description
N	Population size of [P]
R	Reputation assigned to $I \in$ [P]
κ	Defines the region in parameter space for adding I to [M]
θ	Expected return assigned to $I \in$ [P]
β	Learning rate for reputation
α	Learning rate for expected return
$[M]_{min}$	Minimum size of [M]
TC_j	Number of periods I has existed for
MC_j	Number of times I has participated in [M]
$a_i \in A$	An action from the set of actions available to LATIS
$P(a_i)$	The system prediction for action a_i
D	Duration assigned to $I \in$ [M]
Rem	Difference between R_J and D_j
ψ	Accuracy assigned to $I \in$ [M]
er	Excess return assigned to $I \in$ [M]
M_{ref}	Reference model for current time period
κ-region	Region defined by the κ parameter

6.4 Implementation

In this section we describe a specific implementation of the LATIS framework. This implementation is proceeded by a simplified version called LABBS (Learning Adaptive Bollinger Band System) that was created as a prototype and was presented at CIFEr 2012 in New York [14].

The various parameters which were left to be defined by the data modeller will be addressed. This includes, inter alia, what types of technical indicators to include in [P], how the initial population is created and the length of the time period for constructing [M]. We all also define some typical values for the parameters in table 6.2. The optimization algorithm will be dHPSO that is extended to multi-objective optimization. The choice for the time period for evaluating $I \in$ [P] is set to be size of the window used to fit the indicator. Determining whether or not an individual from [P] should be included in [M] is based on the return and Sharpe ratio performance metrics. For the other considerations, they will be discussed in the following subsections. This implementation does not include the covering step as the experiments are focused on evaluating the LATIS framework in relation to the AMH implication of cyclical effectiveness. Therefore, individuals are removed from [P] if they satisfy one of the required criteria but no new individuals are created. This methodology could lead to [P] becoming empty, however, in all experiments performed, the size of the population always remained positive and early stoppage was avoided.

The R and θ meta parameters are seeded with values from the training period. The R for a particular $I \in$ [P] is set to $1/10^{th}$ the size of the training window used to create I and θ is the return acheived during the training period. The final meta parameter, κ, was seeded with a value of 0.80, which was determined during experimentation. The initial values for the meta parameters do not have a large influence on the system as a whole, as long as the learning parameters for their update equations (i.e. α and β from table 6.2) are chosen appropriately.

6.4.1 Technical Indicators for LATIS

We have chosen two technical indicators to be included in [P], the first is the Bollinger band used in the cyclical effectiveness experiments and the second is a moving average indicator (MAI) as described in chapter 3.2.2 and shown in figure 3.2. Signals for a MAI are generated through the interactions of two or more moving averages (of different

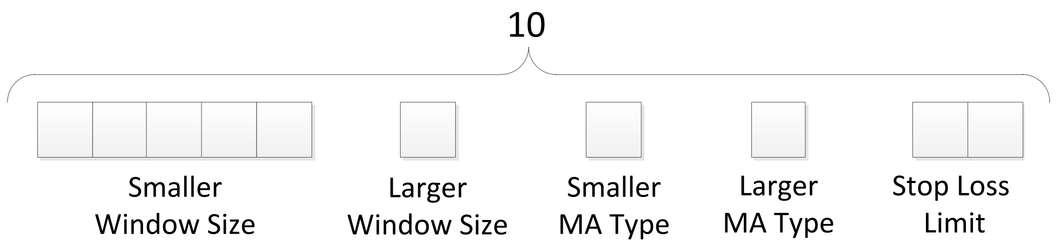

Figure 6.7: The mapping of a particle from the 14-dimensional solution space to the 5-dimensional parameter space for the MAIs.

lengths) of a time series. For this implementation the ABBs will be segmented into two separate functions for optimization, one for the upper band and one for the lower. As such, the PSO mapping is now from 10 particles to 4 parameters. Since the bands are independent of each other there is no benefit to optimizing them at the same time and by separating them the κ-regions can be in lower dimensional space. We have already shown in the related work how an MAI can be combined with a population based algorithm (chapter 3.3.1) but our implementation is different and is described as follows.

Each MAI requires two moving averages of different lengths, a type for each moving average (i.e. arithmetic or exponential) and stop-loss criteria for exiting trades. These requirements yield 5 parameters for optimizing where each particle in the PSO will have 10 particles. The mapping from the 10-d solution space to the 5-d parameter space is shown in figure 6.7. The larger window length is determined as % amount greater than the smaller window, this way the larger window is always guaranteed to be in fact larger and simplifies the mapping procedure.

6.4.2 Creating [P] - training procedure

Given a set of technical indicators to optimize a population [P] can be constructed by fitting the technical indicators to some training data whilst optimizing some economically desirable fitness function(s). In this implementation we train our technical indicators on a set of training data using a sliding window approach with increasing window sizes. This approach enables the technical indicators to be fitted to very specific and more general market conditions which are intended to improve the performance of the meta agent.

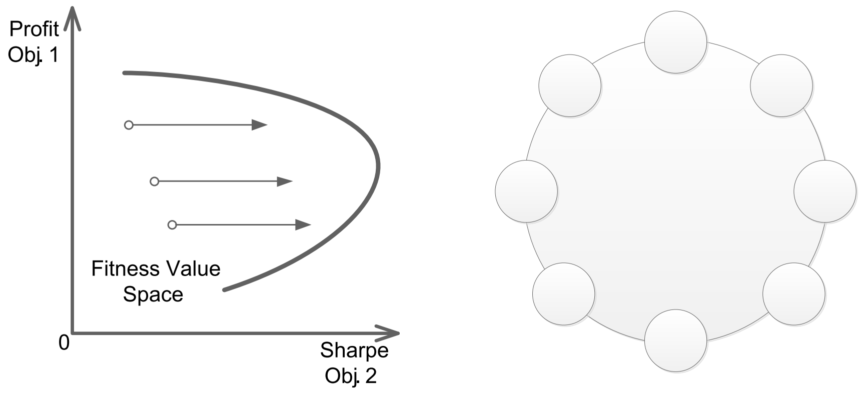

Figure 6.8: (left) The 2-dimensional fitness value space explored by the PSO algorithm , the solid line represents the pareto front. (right) The ring topology of a swarm where particles are influenced locally by their two closest neighbours.

The optimization will consider the two fitness functions which produced the best results from chapter 5.3.2, i.e., profit based and the Sharpe ratio. This creates a multi-objective optimization (MOO), depicted in figure 6.8, where we extend the canonical dHPSO to MOO using the dynamic neighbourhood approach of Hu and Eberhart [53]. This MOO approach is simple and straight forward but has been proven effective when the number of objectives is 3 or less. The algorithm works by optimizing objectives one at a time and executing the following steps:

1. Calculate the distances between each particle in the swarm in the fitness value space of the first objective (profit).

2. Secondly, locate the m closest particles for each particle in the swarm, these m particles form the local neighbourhood.

3. Find the local optimum among the neighbours in terms of the fitness-value of the second objective (Sharpe ratio). This local optimum is used as the $local_{best}$ position in the PSO equations (see chapter 2.1.1.1).

In this study the neighbourhood size was set to 3, so each particle was influenced by 2 neighbours, this creates a ring structure for the neighbourhood topology as depicted in figure 6.8. The other parameter values for the experiments are provided in table 6.2.

6.5 Empirical Results

In this section we report results from modelling simulated and real-world time series using the LATIS implementation described in section 6.4. The experiment design is based on Moody et al. [102] using the same equations for simulating the data and the real world time series is the monthly prices of the S&P 500 market index. The LATIS framework is intended to provide a method for online and adaptive learning when using technical analysis and the LATIS investment policy (the series of trades that LATIS executes) is meant to capitalize on the AMH implication of cyclical effectiveness. Given that the performance of LATIS is dependent on the quality of the models in [P], an informative metric as to LATIS's usefulness is the proportion of individuals in [P] that LATIS is able to outperform. Before the testing period commences we have N models, each of which was profitable in the time-period it was created for, and thus without knowledge of the future, each model would have a $1/N$ chance of being the top performer. If an $I \in$ [P] was chosen at random over a large number of trials then this random policy would on average choose an I in [P] that outperforms 50% of the population. Thus for the LATIS framework to be considered effective it must be able to outperform more that 50% of the I in [P]. Additionally, we also report the overall return of the LATIS algorithm (averaged over a number of runs) and the percentage of runs that LATIS outperformed an investment benchmark.

6.5.1 Trader Simulation

In order to evaluate the LATIS framework we begin with a Monte Carlo simulation, where 1000 independent realizations of equations 5.21-5.23 are generated and then modelled using LATIS. The values for κ and α are 3 and 0.9 respectively, which yields a random walk series with a significant amount of noise and is trending on short time-scales. At each time step LATIS can either be in a short, long or risk-free position. In figure 6.9 we provide four example plots of the simulated series used in the Monte Carlo simulation. We also present results from applying LATIS to one simulated series to show the evolution of the cumulative returns and the series of trades the LATIS was making. Due to the random component of PSO we present results that are averaged over 100 independent runs. The experiment setup parameters for LATIS and PSO are listed in table 6.2, where $|D_{training}|$ is the size of the training window used to construct an $I \in$ [P], $r_{training}$ is an indicator's return during the training period and f_{memory} is the fitness memory for dHPSO for determining when a particle is stagnant and its velocity update profile is changed. In table 6.3 we have the trading parameter settings for LATIS which includes the starting capital, the transaction rate for executing trades and the risk-free rate. As in [102] the simulated trader (LATIS) is able to take either a short, long or risk-free position but never two at the same time. To reiterate, a short position is when the trader sells an asset first and then buys it at a later time, thus profiting from a decline in the asset's price and a long position is buying an asset first and selling it at a later date.

Table 6.2: A list of the LATIS and PSO parameter settings for the simulated and real-world experiments.

LATIS Parameters

Parameter	Setting	Parameter	Setting
N	100	R_0	$0.10*\lvert D_{training}\rvert$
κ_0	0.80	θ	$r_{training}/\lvert D_{training}\rvert$
β	0.10	α	0.10
window sizes	{250,500}	training length	1000

PSO Parameters

Parameter	Setting	Parameter	Setting
particles	30	f_{memory}	5
C1	2	C2	2
intertia	0.99	neighbourhood	3

Table 6.3: A list of the trading parameter settings for the simulated and real-world experiments.

Trading Parameters

Parameter	Description	Setting
$Capital_0$	initial Capital	£1000
Transaction fee	the cost for executing trades	$\frac{1}{4}\%$
RF_{return}	risk-free return	2%
SL_{max}	maximum stop loss	50%

6.5.1.1 Simulated Data Results

This section reports the results from the simulated data experiments. Reported in table 6.4 is the average number of $I \in$ [P] that LATIS outperforms (metric 1), i.e., how many of its own agents does the meta learner outperform and the percentage of runs where LATIS outperformed the buy-hold-approach (metric 2). From the results we observe that on average the LATIS algorithm was able to outperform 91.47% of the $I \in$ [P] which demonstrates that LATIS was able to effectively combine the forecasts from the population and also identify when a particular individual was going to be effective. From the second metric was observed that in 91.1% of the simulations the LATIS algorithm was able to outperform the market benchmark, where the benchmark was the passive

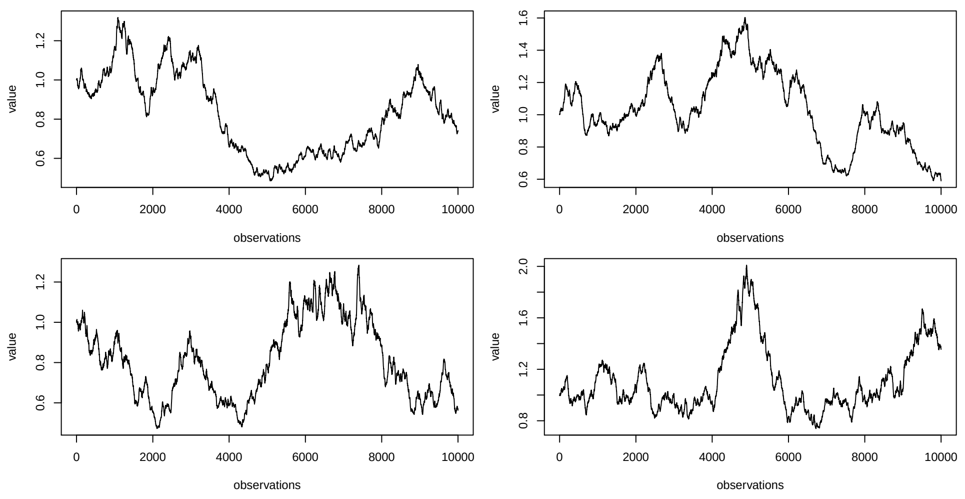

Figure 6.9: Four plots of simulated data generated by equations 5.21-5.23. The plot in the upper left hand corner is the time series used for the results reported in table 6.5.

buy-and-hold investment strategy. This result provides evidence that the extra work and costs associated with trading with LATIS were justifiable and that using the population of indicators modelling short-term trends leads to long-term excess returns.

Table 6.4: The results from a Monte Carlo simulation of 1000 replications. Reported are summary statistics for the average number of $I \in [P]$ that LATIS outperformed (metric 1) and the percentage of simulations where LATIS outperformed the market benchmark (metric 2).

Metric 1		Metric 2	
μ	σ	μ	σ
0.9147	0.1428	0.911	0.285

Now we focus on one realization of the Monte Carlo simulation, depicted in the upper left hand corner of figure 6.9, to analyze the variation in performance over a series of runs for one simulated time series. The reported results include the same metrics from the Monte Carlo simulation plus the average cumulative capital (metric 3). All results have been averaged over 100 independent runs and are reported in table 6.5. In figure 6.10 the average cumulative capital and average trading position are plotted for the online training phase of the algorithm. Considering the trade positions, the reported result for a given trade executed by LATIS is either {-1,0,1} reflecting the current trade for LATIS as a short position, risk-free rate or a long position respectively.

The results in table 6.5 complement the results from the Monte Carlo simulation (table 6.4) where on average LATIS outperformed over 90% of the $I \in$ [P] and in each run was able to outperform the market benchmark. The middle plot in figure 6.10 provides insight into the types of trades that LATIS was executing, where negative values indicate

Table 6.5: The average results from a 100 runs on a simulated data series. Reported are summary statistics for the average number of I ∈ [P] that LATIS outperformed (metric 1), the percentage of simulations where LATIS outperformed the market benchmark (metric 2) and the average cumulative capital achieved by the algorithm (metric 3).

Metric 1		Metric 2		Metric 3	
μ	σ	μ	σ	μ	σ
0.9098	0.040	1	n/a	5506.877	2522.960

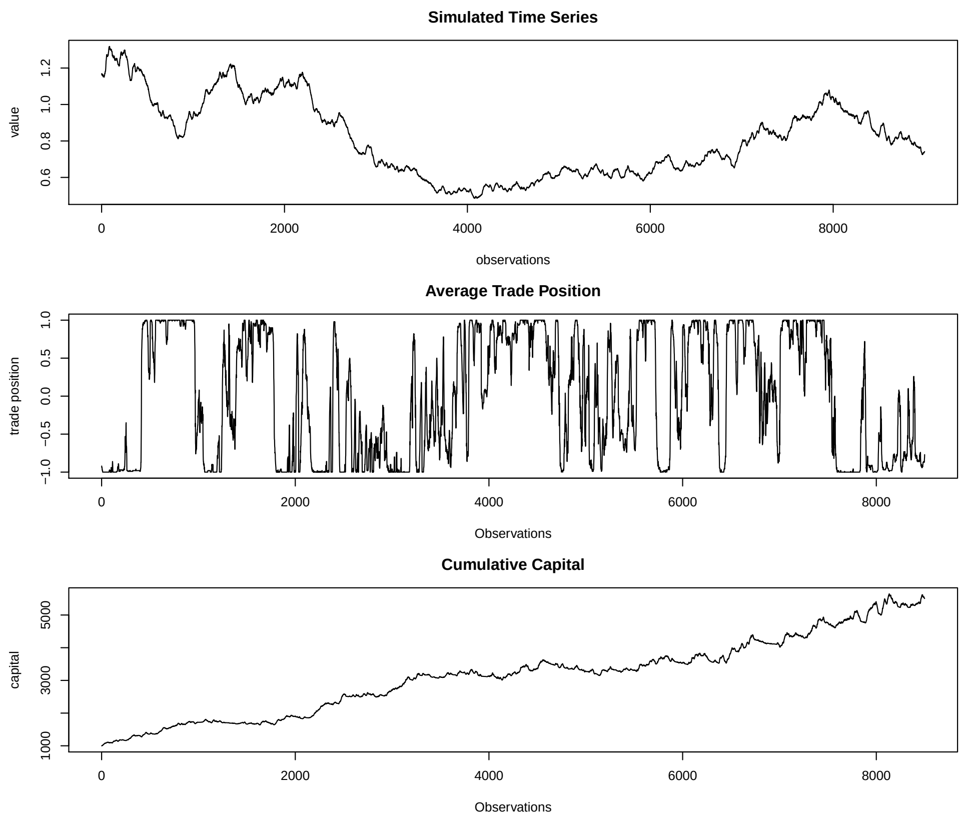

Figure 6.10: The plots of the simulated time series (top), the average trade position (middle) and the cumulative capital for LATIS over the online learning phase (bottom). All results are averaged over 100 runs.

short positions and positive values indicate long positions. During periods were the time-series was significantly trending upwards we can see that LATIS was regularly entering long positions and when the market was negatively trending, LATIS was entering into short positions.

6.5.2 Real World Results

The real world experiments use the monthly price data for the S&P 500 market index spanning 62 years and 10 months from January 1950 to October 2012. The first 25 years are used for training, followed by approximately 21 years (250 months) of updating the population ([P]) with out-of-sample data and finally 17 years for testing and online learning. The 20 year time period for updating the population is to allow all $I \in$ [P] to only have access to data outside of the training phase before LATIS begins to trade. For the real-world experiment the transaction cost is increased to 0.5% or half a percent to reflect higher costs associated with lower frequency trading and to account for other factors associated with real world trading which negatively impact trading profits. For the real world experiments the LATIS trading system can take long, short and risk-free positions. The parameters for the experiment setup and the data description are displayed in table 6.6 and a plot of the time series is provided in the top panel of figure 6.11. We report the same performance metrics as before with the simulated data experiments, where the reported results are averaged over 100 independent runs and displayed in table 6.7.

Table 6.6: A list of the trading parameter settings for the real-world experiments that have been changed. Any parameters not listed are the same as the simulated data experiments.

Experiment Parameters

Parameter	Description	Setting
transaction fee	the cost for executing trades	$\frac{1}{2}$%
window sizes	size of the training windows	{125,250}

Data Description

data name	S&P 500 market index
time period	01/1950 - 10/2012
series length	755
data type	monthly observations

*Table 6.7: The average results from a 100 runs on the monthly prices of the S&P 500. Reported are summary statistics for the average number of $I \in [P]$ that LATIS outperformed (metric 1), the percentage of simulations where LATIS outperformed the market benchmark (metric 2) and the average cumulative capital achieved by the algorithm (metric 3). The * denotes a statistically significant result at the 0.1% significance level.*

Metric 1		Metric 2		Metric 3	
μ	σ	μ	σ	μ	σ
0.7956*	0.167	0.58	n/a	2715.911*	1594.880

The results in table 6.7 show that on average LATIS was able to outperform 80% of the $I \in$ [P]. This represents a statistically significant increase based on a two-sided t-test (at the 1% significance level) from the 50% average assumed a priori. Also, in 58% of the runs LATIS was also able to outperform the buy-and-hold approach. The average cumulative capital of £2715.911 represents a percentage return of 171.5% over the 17 years which is also higher than the buy-and-hold approach (£2235.32). The increase in

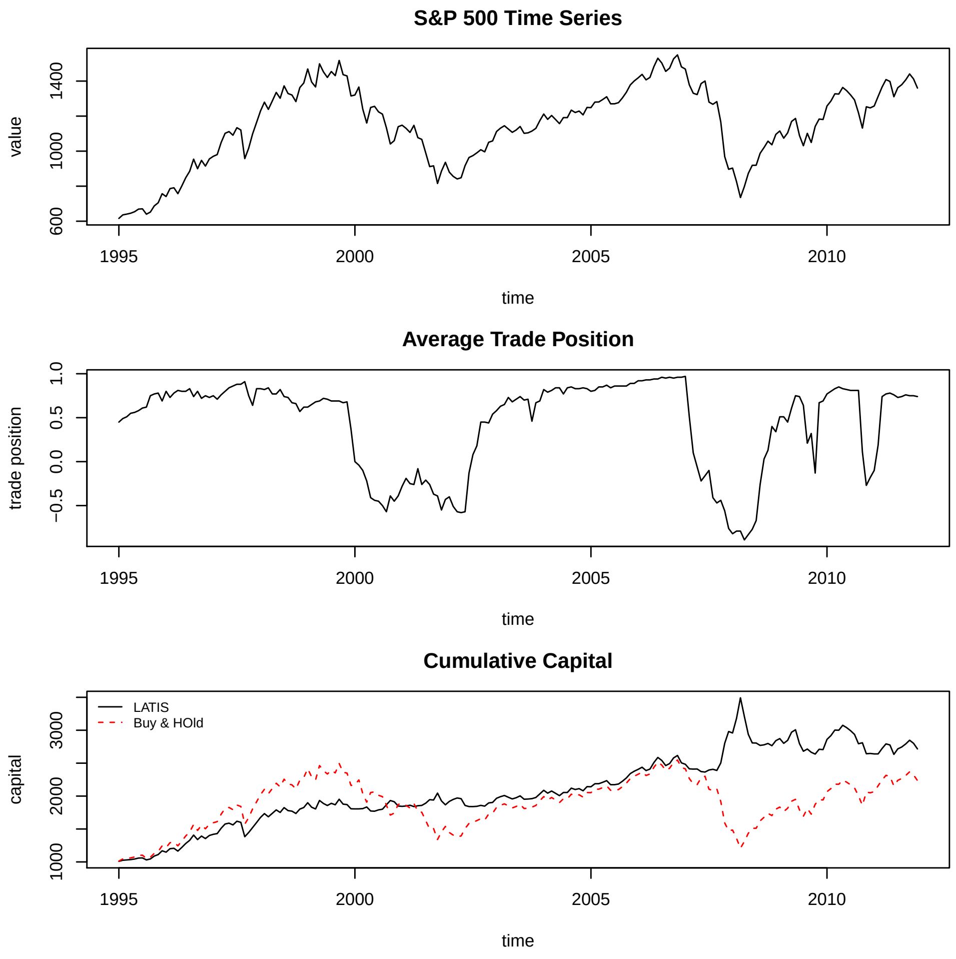

Figure 6.11: The plots of the S&P 500 time series (top), the average trade position (middle) and the cumulative capital for LATIS over the online learning phase (bottom). All results are averaged over 100 runs.

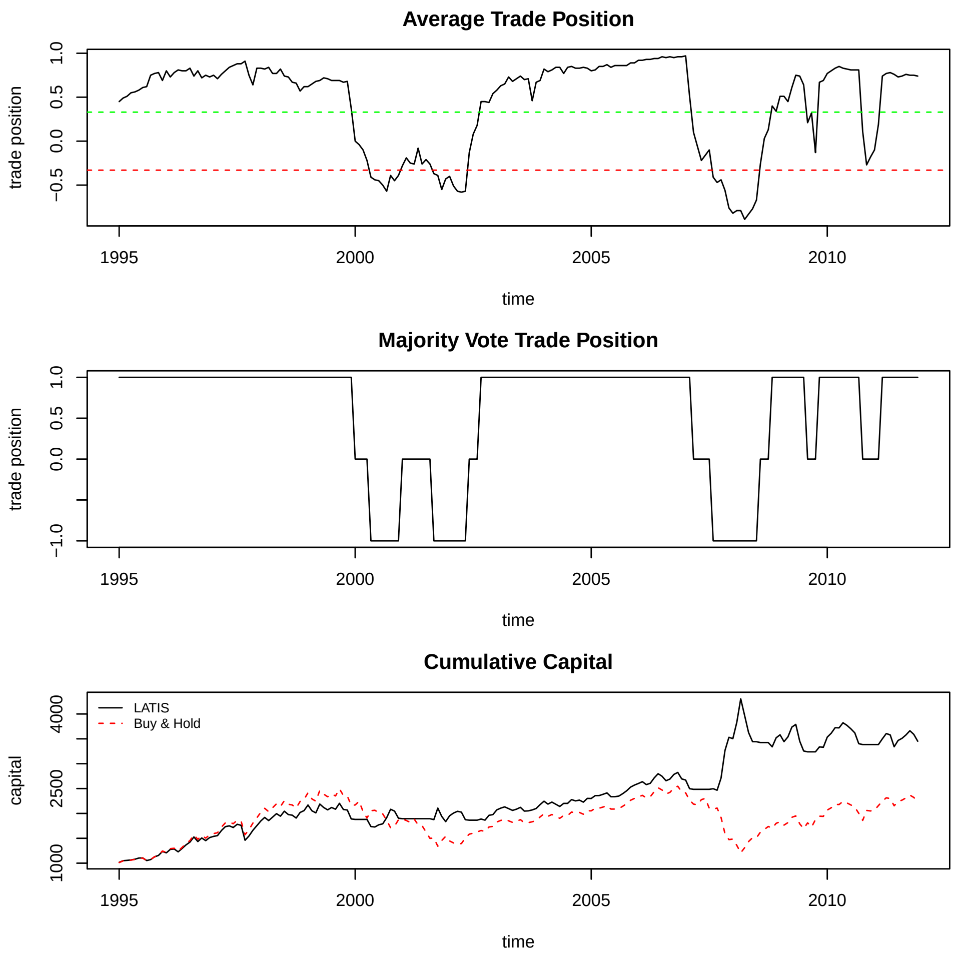

Figure 6.12: The plots of the average trade position (top), the majority vote trade position (middle) and the cumulative capital for LATIS over the online learning phase (bottom). The dashed horizontal lines on the top panel represent the trade decision boundaries for the majority vote. Values about 0.33 are considered a long position, values below -0.33 are a short position and everything in between is a position in a risk-free investment.

the cumulative capital is also significant at the 0.1% significance level based on a two-sided t-test. Figure 6.11 plots the cumulative capital during the online learning phase for the average of the 100 runs. We can see that in the beginning of the period when the market was significantly trending upwards LATIS was underperforming the buy-and-hold approach, however after the dot com market crash (2001), LATIS is able to take short positions to capitalize on the market contraction.

In figure 6.12 we plot the results from following a trading strategy of taking a majority vote from the 100 runs, where each run represents a trading strategy. This trading system is not as active as those previously discussed but does achieve a cumulative capital higher than the average of the 100 runs and the buy-and-hold strategy.

6.6 Chapter Summary

In this chapter we have proposed, implemented and tested a novel algorithm for optimizing a population of technical indicators and combining their heterogeneous set of signals into one coherent trading strategy. We have also extended the dHPSO algorithm to multi-objective optimization to accommodate different but equally important investment objectives. The overall framework of the proposed system is inspired from a Learning Classifier System but has been significantly changed to accommodate a population that does not contain classifiers. The implemented algorithm utilized Bollinger bands and moving average indicators but practically any technical indicator or function that can be fitted using population based optimization would be appropriate. The results from the simulated and real-world data demonstrate the usefulness of the LATIS framework and its ability to detect and exploit trends in a price series. The framework, having been influenced by the cyclical effectiveness results (chapter 5.3), demonstrates that technical indicators can provide actionable information after the time they are identified to be effective.

Chapter 7

Innovations in Discretization

The final contribution chapter covers the area of time series discretization where a real-valued time series is mapped to a symbolic representation. Having discussed the numerous benefits of discretization in sections 2.1.11 and 3.3.2 we now move to the development of a financial time series specific algorithm. The proposed algorithm extends a current state of the art approach to handle the generally non-stationary and non-Gaussian properties of financial time series. However, before the financial specific algorithm is introduced we begin in the following section with a discussion of an invalid core assumption of the SAX algorithm.

7.1 Invalid SAX assumption

The SAX discretization algorithm was the first symbolic approach that mapped a real-valued time series to a symbolic representation that was guaranteed to lower-bound Euclidean distance. At this time we highlight that the interest of this section concerns the SAX assumption of data being highly Gaussian and the use of the standard normal curve to choose partitions to discretize the data. Though not necessarily but generally and certainly in its canonical form the SAX approach chooses partitions on the standard normal curve that would produce an equal probability for each symbol in a finite alphabet to occur. An equal probability of occurrence for each symbol is considered desirable [2] [96] and supports choosing the partitions to segment the area under the curve into equal regions. This procedure is generally a valid approach as a time series is normalized to have a mean of zero and a standard deviation of one before the SAX algorithm is applied. However there exists a caveat to this assumption of equi-probability, which we will explain in more detail in the following sections, due to the intermediate step of Piecewise Aggregate Approximation (PAA). The PAA step is used for dimensionality reduction and, in brief, it converts a time series into a sequence of means. What we will show in this section is that when PAA is applied the distribution of the data is altered, resulting in a shrinking standard deviation that is proportional to the number of points used to create a segment of the PAA representation and the degree of auto-correlation within the series. Data that exhibits statistically significant auto-correlation is less affected by this shrinking distribution. As the standard deviation of the data contracts the mean obviously remains the same, since it was zero, however the distribution is no longer standard normal and therefore the partitions based on the standard normal curve are no longer valid for the assumption of equal probability.

7.1.1 Effects of PAA

As mentioned in the introduction, this section focuses on the effects of the dimensionality reduction step on the distribution of the data. We are asserting that when PAA is applied to a standard normalized data set that the resulting PAA representation will have a smaller standard deviation. This reduction can be trivial and therefore not affect the assumption of equal probability of each symbol in a finite alphabet. However, depending on the size of the PAA segments and the characteristics of the data this effect can have a significant impact. Trivially, we can highlight this effect by stating that the minimum and maximum of the series will be distorted closer to the mean of the distribution, in all but one special case. This special case occurs when the max and min are surrounded by equal valued points that outnumber or equal the number of points in a PAA segment. To further illustrate this effect we provide examples from simulated and real-world time series.

7.1.1.1 Simulated Time Series

This study utilizes three simulated time series which represent the two extreme cases of having highly correlated data (sinusoidal wave form) and completely uncorrelated (white noise) and a mixture of both (sinusoidal wave with added white noise). Figure 7.1 displays the autocorrelation functions for the three series. From these plots we observe that the random data has no significant autocorrelation at any lag, the sinusoidal wave has perfect autocorrelation at lag 1 (which tapers off slowly) and the sinusoidal wave with noise is in between. Additionally, table 7.1 displays the standard deviations of the simulated time series after the PAA step has been applied for various PAA parameter settings.

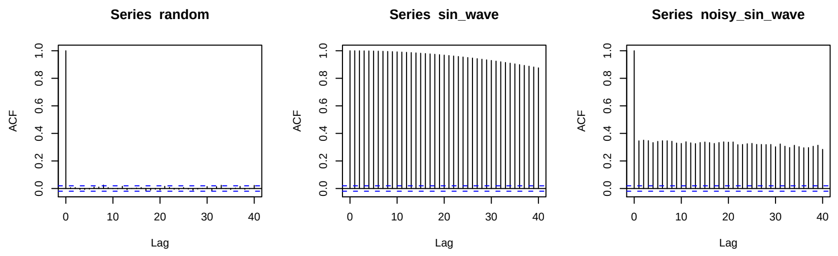

Figure 7.1: Autocorrelation Function (ACF) plots of the three simulated series.

Firstly we have the case where the time series is composed of random data drawn from a standard normal distribution that exhibits no statistically significant autocorrelation. Depicted in figure 7.2 is a plot of the series and box plots showing the distributions of the letters after the SAX algorithm has been applied. The box plots represent increasing values for the number of points used to construct the PAA segments. Clearly as the number of points increases within a PAA segment the distribution of letters that SAX is mapping to contracts closer to the mean, thus losing the desirable outcome of an equal distribution. The bar chart labelled "B" shows the distribution of letters when no dimensionality reduction is applied and is producing a uniform distribution. If we examine the results reported in table 7.1 we observe that the standard deviation of the distribution is shrinking as the PAA segments become larger.

Secondly we have the case of highly correlated data with the sinusoidal wave form. This data, depicted in figure 7.3, was generated using equation 7.1:

$$A * cos(2\pi\omega t + \phi) \tag{7.1}$$

where A is the amplitude, ω is the frequency of oscillation, and ϕ is a phase shift, where ϕ = B * π and B is a constant. For this simulation the values for A, ω, B were 2, 0.002, and 0.6 respectively. From figure 7.3 we can see that a sin wave is not Gaussian and therefore does not produce a uniform distribution of symbols. However, the data is highly correlated and the PAA step has no effect on the distribution of the data. This result is reflected in the corresponding results in table 7.1 where the standard deviation is only trivially affected.

The final example demonstrates that as the data exhibits lower degrees of autocorrelation the PAA step has a larger impact on the post PAA data distribution. In figure 7.4 we have the plots of the same sinusoidal wave as before but with added Gaussian white noise.

As we can see from the box plots in figure 7.4 the data is again not Gaussian but this time the distribution is being altered as the PAA step is applied with more and more data

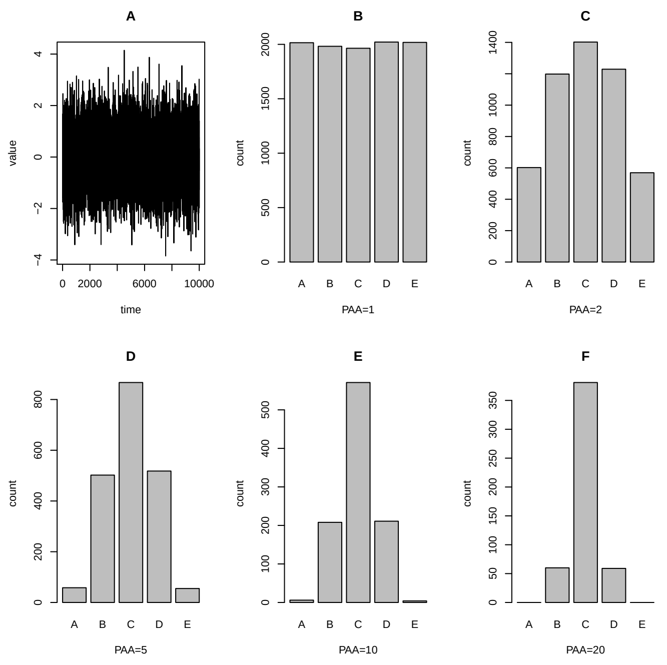

Figure 7.2: Plots related to a standard normal distribution. "A" is a time series plot of the data. "B"-"F" are box plots of the distributions of letters from a symbolic sequence derived from SAX for increasing number of points used to create the PAA segments. The desired distribution amongst the letters is uniform and is only achieved when the PAA step is skipped (Graph "B").

Table 7.1: The standard deviations for the data distributions ex-post the PAA step of SAX.

	1	2	5	10	20
random	0.9999	0.7139	0.4448	0.3167	0.2258
sin wave	0.9999	0.9999	0.9998	0.9993	0.9973
sin wave with noise	0.9999	0.8162	0.6909	0.6394	0.6098

points per segment. The effect is clearly observed in the standard deviations of the ex-post distributions (table 7.1) where the standard deviation shrinks from approximately 1 to 0.6098 for PAA segments equal to 20 data points.

7.1.1.2 Real World Time Series

To demonstrate this effect in real world time series we have chosen 12 data series from 8 sources available from the UCI machine learning repository [41] and downloaded from Dr. Eamonn Keogh iSAX webpage [124]. We have chosen 11 series which are negatively impacted by the PAA step and 1 which is not. The 12 data series are summarized in table 7.2 which reports the results from the Jarque-Bera test for normality and figure 7.5

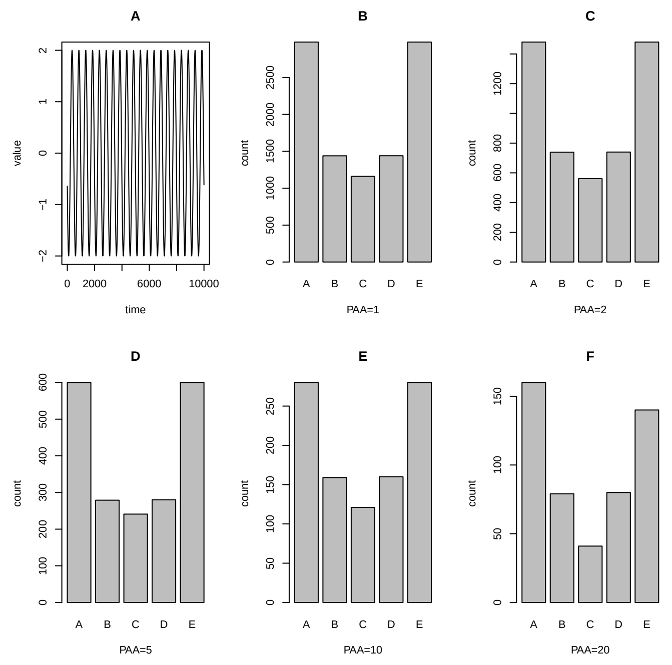

Figure 7.3: Plots related to a sinusoidal wave distribution. "A" is a time series plot of the sin wave data. 'B"-"F" are box plots of the distributions of letters from a symbolic sequence derived from SAX for increasing number of points used to create the PAA segments. The desired distribution amongst the letters is uniform and is never achieved as a sin wave is not Gaussian. However, the distribution is never significantly affected by the PAA step.

displays the ACF plots. Based on the Jarque-Bera test results only one of the time series is normal (robot_2). The ACF plots are arranged based on the impact the PAA step had on the ex-post distribution. Where the data series that was most affected is in the top left hand corner and the data set least affected is in the bottom right hand corner. From these plots we can see that the series with ACFs which are positive and taper off slowly (similar to the sinusoidal wave) were the least affected by the PAA step.

The real world time series are standardized and then converted to a PAA representation. Table 7.3 displays the ex-post standard deviations of the PAA representations of the standardized time series. As we can see for all time series the standard deviations of the PAA representations are shrinking as the size of the PAA segments increases. However, the spot FX rate exhibits only minor changes in its distribution, thus not impacting the assumption of equi-probability of each symbol. This is similar to the results obtained with the sinusoidal wave without added noise. It is also worth noting that the ACF of the FX rate is similar to the sinusoidal wave as well. The reduction in the standard deviation in the other 11 series is much more dramatic and would most likely have a negative impact on

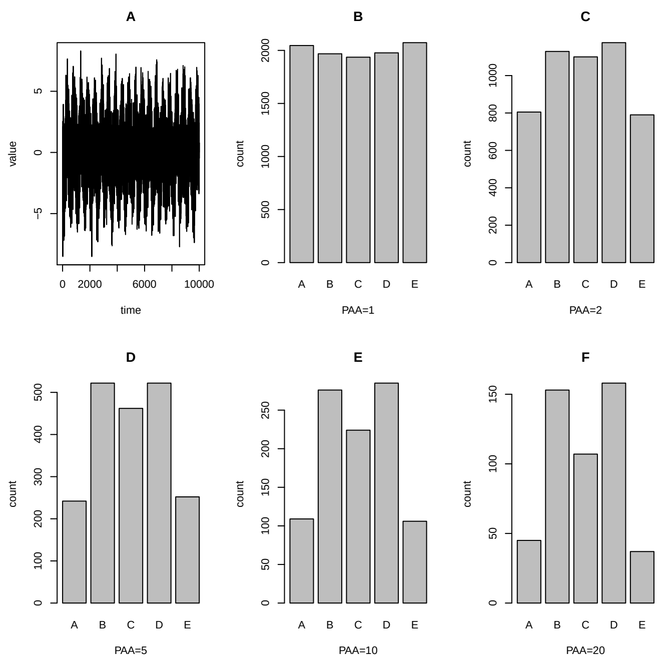

Figure 7.4: Plots related to a sinusoidal wave with added noise distribution. "A" is a time series plot of the sin wave data. 'B"-"F" are box plots of the distributions of letters from a symbolic sequence derived from SAX for increasing number of points used to create the PAA segments. The added noise lowers the level of autocorrelation within the sin wave and therefore as PAA is applied the distribution of the data changes.

Table 7.2: The real world time series used to demonstrate the negative effects of the PAA step on the ex-post distributions. JB test stands for the Jarque-Bera test for normality and reports the p-value obtained.

Data file (name)	Description	length	JB test
darwin.dat (darwin)	Monthly values - Darwin SLP series	1400	<0.001
flutter.dat (flutter_1)	Wing flutter data (input)	1024	<0.001
flutter.dat (flutter_2)	Wing flutter data (output)	1024	<0.001
robot_arm.dat (robot_1)	Data from a robot arm (input)	1024	<0.001
robot_arm.dat (robot_2)	Data from a robot arm (output)	1024	0.5042
sunspot.dat (sunspot)	Monthly data - 01/1749 to 07/1990	2899	<0.001
EEG_heart_rate.dat (heart)	Heart Rate after Epileptic seizure	7200	<0.001
water.dat (water_1)	Rainfall riverflow data (aprecip)	2191	<0.001
water.dat (water_2)	Rainfall riverflow data (discharg)	2191	<0.001
water.dat (water_3)	Rainfall riverflow data (Log Flow Rate)	2191	<0.001
spot_exrates.dat (fx_rate)	Spot prices for GBP in USD	2567	<0.001
balloon.dat (balloon)	Balloon collected radiation data	2001	<0.001

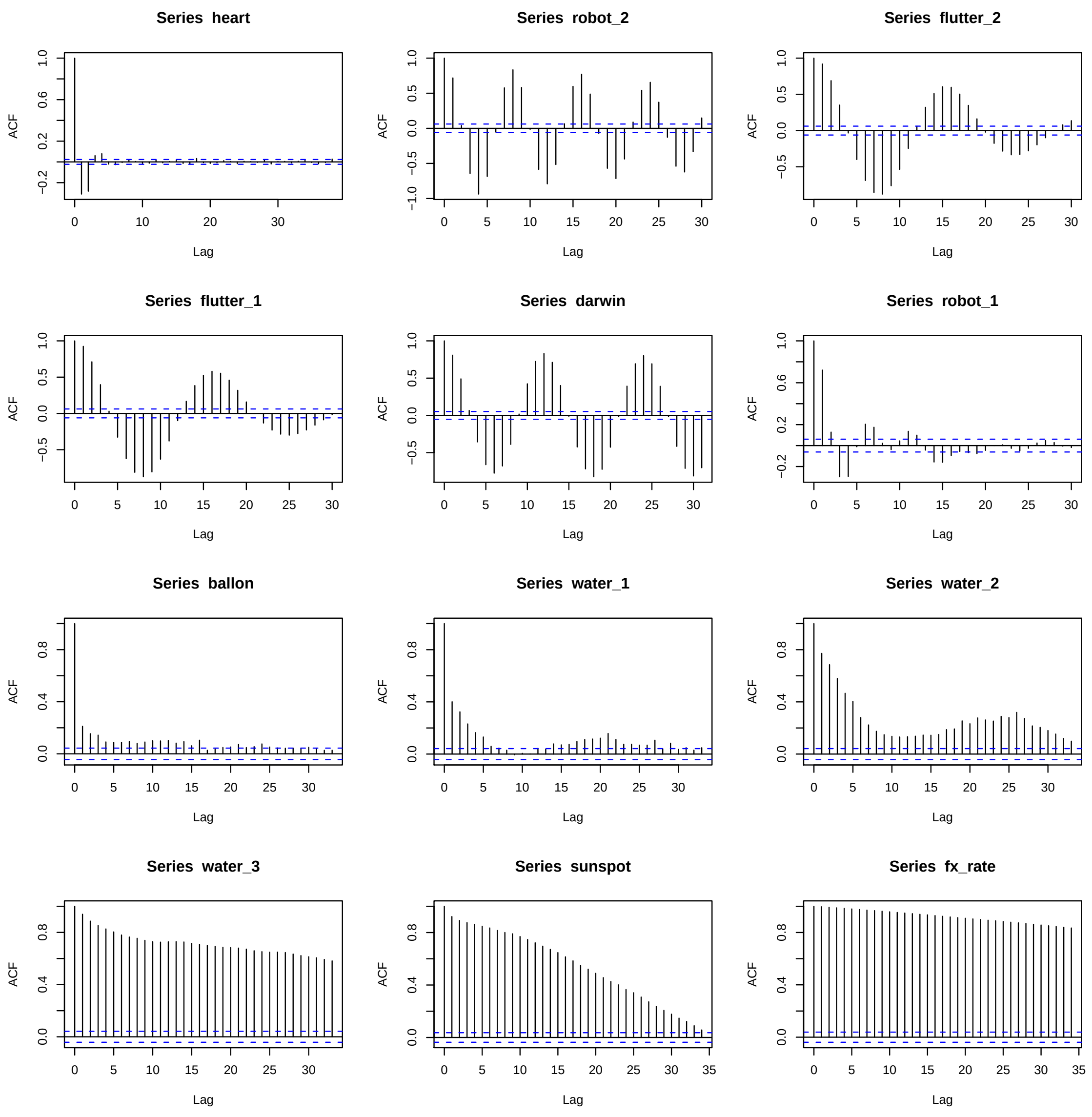

Figure 7.5: Plots of the Autocorrelation functions for the 12 real world time series considered in the study. The ACF plots are arranged based on the impact the PAA step had on the ex-post distribution. Where the data series that was most affected is in the top left hand corner and the data set least effected is in the bottom right hand corner.

Table 7.3: The standard deviations for the data distributions ex-post the PAA step of SAX.

	Size of PAA Segment			
Name	2	5	10	20
heart	0.5975	0.2246	0.1216	0.0659
robot_2	0.9268	0.5178	0.1966	0.1404
flutter_2	0.9794	0.8461	0.3982	0.1479
flutter_1	0.9810	0.8577	0.5665	0.1503
darwin	0.9496	0.7406	0.3295	0.2607
robot_1	0.9264	0.6449	0.4764	0.3690
balloon	0.7766	0.5944	0.4690	0.3934
water_1	0.8436	0.6683	0.5911	0.4337
water_2	0.9461	0.8745	0.7970	0.6157
water_3	0.9832	0.9570	0.9283	0.8914
sunspot	0.9799	0.9572	0.9341	0.8928
fx_rate	0.9994	0.9975	0.9951	0.9875

the symbolic distribution. This is the same property observed in the simulated sinusoidal wave with noise and the random data drawn from a standard normalized distribution. The ACFs of these 11 series vary in their behaviour but the most affected series are those which have no statistically significant positive autocorrelation at any lag or exhibit oscillating positive and negative autocorrelation.

7.1.2 Effect on Symbolic Distributions

With the property of shrinking distributions now documented in the post PAA representations, the question remains as to what effect this has on the distribution of symbols from a finite alphabet. To demonstrate this effect we assume that a discretization becomes more desirable as its distribution of symbols approaches uniformity. Additionally we introduce a potential fix for this effect, where the PAA data can simply be re-normalized before transformation into a symbolic sequence. Thus the effect can be measured using a chi-squared (χ^2) goodness-of-fit test between the SAX representation and the target uniform distribution. This test will be performed using the canonical form of SAX as well the aforementioned augmented version that renormalizes the PAA representation to have μ=0 and σ=1. As previously shown (table 7.2), only one series can be considered Gaussian (*robot_2*) and therefore is the only one which we would expect a uniform distribution from the symbolic sequence. For the other 11 series we would expect that the majority of them will reject the null of the χ^2 test but that the absolute deviations from the uniform distributions should be smaller for a normalized distribution with a σ=1. Therefore if the shrinking σ of the PAA representations is negatively impacting the mapping to a symbolic

*Table 7.4: The results from performing a χ^2 goodness-of-fit test on the SAX symbolic distributions with alphabet cardinality of 5. Reported are the absolute deviations from the uniform distribution. A * indicates the null was not rejected at the 5% significance level. SAX_n indicates the results when the PAA distribution was re-normalized prior to converting the PAA vector to symbols.*

	Size of PAA Segment							
Name	2		5		10		20	
	SAX	SAX_n	SAX	SAX_n	SAX	SAX_n	SAX	SAX_n
darwin	19.43	24.00	25.71	20.00	80.00	11.43*	94.29	17.14*
flutter_1	72.11	72.11	83.53	73.73	101.18	85.49	132.55	73.73
flutter_2	78.75	77.58	78.63	77.65	105.10	83.53	148.24	81.57
robot_1	6.17*	8.52*	33.92	14.31*	62.35	12.16*	80.00	21.96*
robot_2	13.75	10.63*	57.45	4.51*	120.78	9.41*	144.31	10.20*
sunspot	16.07	16.48	16.44	16.03	15.64	13.43*	17.50*	14.72*
heart	48.28	3.72*	110.69	4.86*	153.06	8.33*	158.89	8.33*
water_1	138.81	118.36	132.60	105.66	129.86	88.77	125.14	77.43
water_2	147.58	146.85	147.67	146.30	147.21	143.56	150.83	101.65
water_3	31.78	31.42	29.59	25.94	27.76	25.02	31.93	24.59*
balloon	72.60	70.40	73.00	52.00	81.00	34.00	90.00	42.00
fx_rate	24.97	25.13	25.81	26.20	24.53	25.31	26.88	26.88

sequence we should observe smaller absolute deviations and more acceptances of the null of the χ^2 test with the re-normalized PAA representations. The results are reported for the same PAA segmentations as displayed in table 7.3 and for alphabet cardinalities of 5 and 10. In tables 7.4 and 7.5 we report the absolute deviations from the uniform distribution for alphabet cardinalities of 5 and 10 respectively; any results which did not reject the null of the χ^2 test (at the 5% significance level) are marked with an asterisk (*). The null of a goodness-of-fit test is that the observed distribution is equal to the expected distribution. The χ^2 test statistic is calculated as follows:

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{7.2}$$

where X^2 is the Pearson's cumulative test statistic, O_i is the i^{th} observed frequency, E_i is the i^{th} expected or target frequency and n is the cardinality of the finite alphabet. In these experiments the target frequency would be calculated as 1/(cardinality of the alphabet), so for an alphabet cardinality of 5 the target frequency is 0.2.

To summarize tables 7.4 and 7.5, two statistics can be determined. Firstly, there are 96 cases reported across both tables and in 80 of those cases re-normalizing the data produced a distribution as close or closer to uniform. Secondly, the χ^2 test accepted the null that the two distributions (observed and expected) were equal in 29 of the 96 cases when the

*Table 7.5: The results from performing a χ^2 goodness-of-fit test on the SAX symbolic distributions with alphabet cardinality of 10. Reported are the absolute deviations from the uniform distribution. A * indicates the null was not rejected at the 5% significance level. SAX_n indicates the results when the PAA distribution was re-normalized prior to converting the PAA vector to symbols.*

	Size of PAA Segment							
Name	2		5		10		20	
	SAX	SAX_n	SAX	SAX_n	SAX	SAX_n	SAX	SAX_n
darwin	30.86	31.14	32.14	22.86	97.14	21.43*	111.43	34.29*
flutter_1	72.11	73.52	83.53	73.73	101.18	88.63	132.55	73.73
flutter_2	78.75	77.58	78.63	77.65	105.10	83.53	148.24	81.57
robot_1	7.97*	9.84*	36.86	16.67*	74.90	16.47*	92.55	41.18*
robot_2	13.75*	14.84	58.24	12.16*	122.35	21.96*	144.31	21.96*
sunspot	41.48	42.31	40.24	42.63	39.79	46.71	43.06	45.00
heart	49.17	5.78	114.86	4.86*	153.06	9.44*	158.89	11.67*
water_1	150.23	138.36	138.45	125.66	129.86	109.77	125.14	77.61
water_2	151.69	151.69	148.95	148.95	147.21	145.30	150.83	125.14
water_3	44.75	44.66	42.47	42.92	41.46	36.99	41.28	34.13
balloon	81.80	74.60	78.00	54.50	94.00	45.00	106.00	46.00
fx_rate	38.16	38.32	40.08	39.69	41.88	41.88	39.06	39.06

PAA distribution was re-normalized but only accepted the null in 4 of the 96 cases when it was not. Based on these results we can conclude that transforming the PAA distribution to standard normal before converting to a symbolic representation, facilities a more even distribution of the symbols. However, in the case of the FX rate, re-normalizing lead to poorer results from the χ^2 test in 4 of the 8 cases. In the 2 cases that a better result was achieved, it was only a marginal improvement. From these results we can conclude that, whether or not the effect of the PAA step can be determined a priori, a simple test of the PAA distribution is all that is required to decide if a re-normalization is necessary. Clearly in the case of the FX rate the re-normalization was not advantageous.

If we focus on robot_2, the only series that was Gaussian, we observe similar results to those obtained with the simulated data from the standard normal distribution. The σ steadily shrank and the absolute deviation from the uniform distribution grew as the size of the PAA segments became larger. When we examine the re-normalized PAA representation we see that in 7 of the 8 cases the symbolic sequence produced did conform to a uniform distribution and in the one case that the χ^2 test rejected the null, it was marginal.

7.1.3 Effect of Autocorrelation

We are asserting that autocorrelation is a non-trivial contributing factor to the effect of the PAA step on the standard deviation of the PAA representation distribution. Figure 7.5 displays the plots the ACFs of the time series in order of effect the PAA step had on the standard deviation for the PAA segment size of 20. This figure depicted a succession of ACF plots that became more and more positively autocorrelated as the effect on the standard deviation diminished. To further highlight this point in figure 7.6 we provide a plot of the sums of the autocorrelation coefficients up to 30 lags for the 12 series along with a plot of the standard deviations. In figure 7.6 the solid line represents the sum of the ACF coefficients and the dashed line represents the standard deviations. The two plots are clearly linearly correlated, where an increase in the ACF sum is related to a decrease in the effect on the post PAA distribution. This relationship is also expressed by the correlation coefficient between the two series which is 0.9527805. Additionally if we fit a linear regression model of the form:

$$Y = X\beta + \epsilon \tag{7.3}$$

where Y (the response variable) represents the σ and X represents the ACF coefficient sums as the predictor. The resulting adjusted R^2 value is 0.8989, thus indicating that approximately 90% of the variance in the post PAA standard deviations is explained by the autocorrelation.

7.2 Effect on Data Mining

The observation that a SAX discretization of a real-valued time series results in a shrinking standard deviation will undoubtedly effect data mining tasks. However, that effect is not necessarily negative nor is it homogenous within the variety of data mining tasks (i.e. classification and clustering), application areas and algorithms. In a general sense, we can quantify the effect in terms of information theory, where if the probability of each symbol occurring is no longer uniform we can base the information loss on a measure of entropy (H) within the system. For example, if we consider a time series, X_i, where $i = 1,..,n$, with a PAA segment size (m) = 10 and alphabet cardinality = 10 as well, then assuming a uniform distribution of letters we have the following number of bits required to transmit

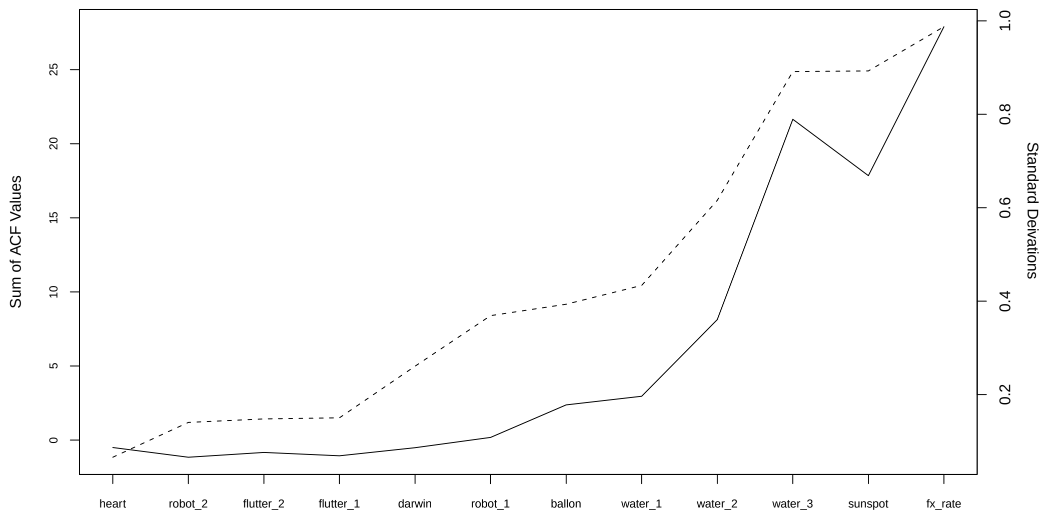

Figure 7.6: Plots of the standard deviations (dashed line) and ACF coefficient sums (solid line) for the 12 data series under analysis. The two plots reveal a strong linear relationship between the variables which is also expressed by the correlation coefficient between them of 0.9527805.

the signal, (i.e., information content):

$$\begin{aligned} H(X) &= -\sum_{j=1}^{n/m} p(x_j)\log_2 p(x_j) \\ H(X) &= 3.3219 \end{aligned} \tag{7.4}$$

where the number of PAA segments in X is n/m or the compression rate, $\forall$ j: $p(x_j) = 0.10$ and the information content per symbol with a uniform distribution is 3.3219. However, if the post PAA distribution has shrunk and the assumption of uniformity is invalid, then we can easily quantify the theoretical loss of information by plugging the new probabilities into equation 7.4.

Aside from a theoretical information loss there are several methods for making this new information valuable to data mining tasks, but ultimately the preferred methodology will be specific to the needs of the data mining expert and the problem at hand. However, we present the following as a non-exhaustive list of potentially viable approaches to utilizing this information:

1. Re-normalize the data after the PAA step
2. Filter the data based on the ex-post PAA step standard deviation

The first procedure was used in a previous section (section 7.1.2) to show that if the desired outcome was to be as close to a uniform distribution as possible then re-normalizing the PAA segment data would facilitate this goal. However, this may not be the case; certainly when extreme values are present and pertinent to the data mining task, diluting these points through renormalizing is not advantageous. The second procedure which uses a filter based on the post PAA standard deviation allows for this characteristic of the data to be directly considered when performing a data mining task. For example, when performing a classification of a given time series using the nearest neighbour algorithm, only the training exemplars with sufficiently similar post PAA distributions will be considered. It is possible that using a filter may result in no training exemplars meeting the criteria and thus leaving a testing exemplar unclassified. However, this is not necessarily a negative consequence and assuming the training data has been chosen appropriately (i.e. it is representative of the problem domain) then this situation should be avoided.

7.3 Case Study

To demonstrate the "actionability" of this information we present a small case study which considers the classification accuracy of the nearest neighbour (NN) algorithm using the SAX distance measure (equation 2.20). The data set under study is the popular Synthetic Lightning EMP data set [59] where we use the split of 2,000 cases for training and 18,000 for testing. Each time series exemplar contains 2000 points and is a member of either the "slow leading edge" class (class 0) or the "fast leading edge" class (class 1). To test if the information pertaining to the shrinking standard deviation could be used for improving performance, we compare the classification accuracies of the proposed procedures to the canonical procedure of simply applying SAX to the data and then classifying using NN. The experiments are performed using a range of PAA segment sizes and alphabet cardinalities, where the PAA segment sizes considered are 5 and 10 and the alphabet cardinalities range from 5 to 20. This setup results in 32 unique scenarios for comparison.

7.3.1 Experiment Results

The results from performing the experiment setup described previously are displayed in tables 7.6 to 7.8 for the canonical approach, the procedure of re-normalizing, and the filter approach respectively. The final two rows in each table report the averages and standard deviations across alphabet cardinalities for each PAA segment size. For the filter approach the results reflect using a threshold of 5% (based on the original $\sigma^2 = 1$), i.e., only training exemplars with a post PAA distribution within ±0.05 of a given testing exemplar's post PAA distribution will be considered for classification. Using a filter of this size allowed for all exemplars in the testing set to have at least one training exemplar for classification. In figure 7.7 we display box plots of the post PAA distributions for all data in the training set for PAA segment sizes of 5 and 10. From the box plots we can see that a range of post PAA distributions are present, this implies that the degree of autocorrelation within the training set exemplars is not homogenous.

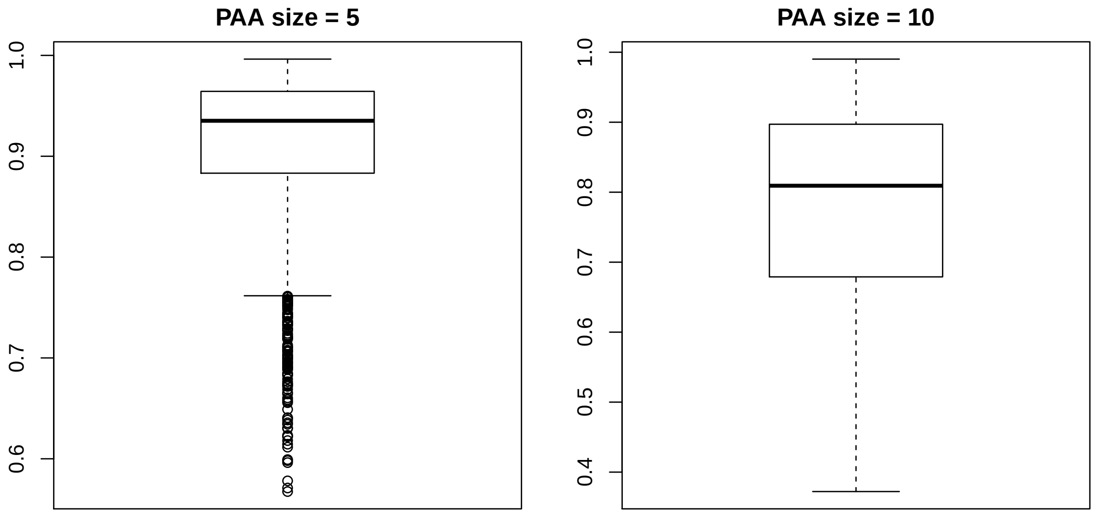

Figure 7.7: Box plots of the ex-post PAA standard deviations for the training data of the Synthetic Lightning EMP data set under different PAA segment sizes.

7.3.2 Results Analysis and Discussion

From tables 7.6-7.8 we can see that the average classification accuracy is higher for the filter approach for each PAA segment size and in 100% of the cases considered the filter approach yields a higher classification accuracy than the canonical SAX approach. However, the process of re-normalization after the PAA step leads to inferior results, suggesting that in this case a uniform distribution of letters was not desirable.

If we average across all results then the average accuracy for the filter approach is 86.0600% and for canonical SAX approach we have 85.0547% with standard deviations of 0.9094 and 0.8101 respectively. Performing a Student t-test on the average classification accuracies yields a t-score = 4.669456509, which based on a two-sided t-test is significant at the 0.1% significance level with a value = 0.000016295. The calculation

Table 7.6: Classification results for the NN algorithm using canonical SAX.

	PAA Segment Size			
$Alphabet_{size}$	5		10	
	Accuracy (%)	Correct	Accuracy (%)	Correct
5	82.59	14866	82.06	14771
6	84.61	15230	84.12	15142
7	85.01	15302	84.98	15296
8	84.9	15282	85.44	15379
9	85.06	15311	84.63	15233
10	85.18	15332	85.13	15323
11	85.47	15386	85.87	15457
12	85.1	15318	85.01	15302
13	85.47	15384	85.77	15438
14	85.41	15374	85.63	15414
15	85.39	15371	85.09	15317
16	85.38	15370	85.44	15380
17	85.47	15385	85.83	15369
18	85.37	15368	85.26	15347
19	85.33	15360	85.36	15364
20	84.99	15298	85.4	15372
mean	85.05	15308.56	85.06	15306.55
sd	0.7003	126.25	0.9200	162.35

was based on the following:

$$\begin{aligned} t_{score} &= (86.0600 - 85.05469)/(\sqrt{0.5 * (0.9094^2 + 0.8101^2)}) * \sqrt{(2/32)} \\ &= 1.0053/0.2153 \\ &= 4.6693 \end{aligned}$$

where over 32 independent cases for the experiment setup yields (32-2) degrees of freedom for the t-test. In addition to the student's t-test we have also performed a Wilcoxon rank sum test. The resulting test score was 232 which yields a p-value ≤ 0.001, thus rejecting the null that the two distributions are equal and therefore the filter approach was superior.

In addition to the increased accuracy, the filter approach also offers an advantage in terms of execution time, as distances are only calculated when post PAA distributions are similar. Ignoring implementation bias and the initial calculation of the ex-post standard deviations, the filter approach only adds one operation for each comparison in the testing phase, i.e. the similarity check. However, if the exemplars do not have similar post PAA standard deviations then the algorithm is saved *n* comparisons, *n* being the length of the

Table 7.7: Classification results for the NN algorithm using SAX but with a re-normalization of the data after the PAA step.

	PAA Segment Size			
Alphabet$_{size}$	5		10	
	Accuracy (%)	Correct	Accuracy (%)	Correct
5	81.89	14740	78.24	14083
6	83.3	14994	80.38	14468
7	84.2	15156	80.52	14494
8	84.16	15149	81.69	14704
9	84.22	15160	81.85	14733
10	85.54	15397	81.89	14740
11	84.54	15217	82.19	14795
12	84.46	15202	82.06	14770
13	84.8	15264	82.27	14808
14	84.79	15263	82.67	14881
15	85.12	15321	82.25	14805
16	84.58	15224	82.76	14896
17	84.62	15232	82.58	14865
18	84.87	15277	82.67	14881
19	84.86	15274	82.93	14927
20	84.84	15272	82.74	14893
mean	84.42	15196.36	81.86	14733.98
sd	0.8363	150.53	1.2151	218.71

SAX representation of the time series. The overall result is a savings of n-1 comparisons per testing exemplar and as the length of n increases, the larger the reduction in run time.[1] Since the ex-post PAA distributions of the training set can be analyzed prior to the testing phase (see figure 7.7) there is little risk of increasing the computation time of the algorithm using the filter. In other words, the filter approach is only used when there is variation in the post PAA distributions.

7.3.3 Conclusions

In this section we highlighted conditions under which one of the core assumptions of the SAX algorithm is invalid. When the PAA step is applied for dimensionality reduction, the standard deviation of the resulting distribution is almost certainly less than 1, and depending on the circumstances, can be much less than 1. Those circumstances concern two measurable variables, the number of data points within a given PAA segment and the degree of positive autocorrelation within the series. A time series with statistically

[1]The exact benefits from this speed-up are still to be studied.

Table 7.8: Classification results for the NN algorithm using SAX but with a filter based on the post PAA distribution.

	PAA Segment Size			
$Alphabet_{size}$	5		10	
	Accuracy (%)	Correct	Accuracy (%)	Correct
5	83.63	15054	82.87	14918
6	85.16	15329	84.41	15194
7	85.81	15446	85.17	15332
8	85.93	15468	85.84	15451
9	86.27	15529	85.86	15454
10	86.43	15557	86.27	15528
11	86.48	15567	86.68	15603
12	86.08	15494	86.33	15539
13	86.58	15586	86.91	15644
14	86.65	15597	86.63	15593
15	86.58	15585	86.37	15546
16	86.49	15568	86.39	15551
17	86.71	15607	86.36	15545
18	86.58	15584	86.6	15588
19	86.59	15586	86.4	15552
20	86.33	15540	86.53	15575
mean	86.14	15506.06	85.98	15475.81
sd	0.7815	140.541	1.0373	186.34

significant autocorrelation that tapers off slowly will be less affected by the PAA step and vice-versa. We have shown that the shrinking distribution negatively effects the symbolic representation of the time series with respect to the target uniform distribution. Finally, we have provided a small case study which demonstrates how knowledge of the PAA affect can improve classification accuracy when using SAX in conjunction with the nearest neighbour algorithm. The results from the case study were statistically significant at the 0.1% significance level (based on two separate tests) and on average produced 183.38 more correctly classified instances in the Synthetic Lightning EMP data set. Although the case study focused on a particular algorithm in a particular domain, the more general impact of the paper is that a previously believed property of an algorithm is invalid. How this new information affects each data mining task is left to the modelling expert but at least now this property (i.e. the shrinking standard deviation) can be taken into consideration. Adding the extra step of analyzing the post PAA distributions of the time-series data will allow a very popular algorithm to realize its full potential in the variety of domains where it has already proved very useful.

7.4 SAX for Financial Time Series

We discussed previous attempts to extend SAX in chapter 3.3.2 and highlighted that these studies failed to accommodate the problematic characteristics of financial time series. Certainly the researchers recognized that SAX had to be altered to improve the matching of similar shapes occurring at different time periods. However, the true cause was overlooked, which is the generally non-stationary nature of financial time series and the periodic departures from a Gaussian distribution. In this section we describe an algorithm called alSAX (Adaptive Local SAX), which extends the canonical version of SAX such that the algorithm automatically accommodates for non-stationarity and the departures from a normal distribution. The resulting algorithm preserves the important property of SAX that the distance measure lower bounds Euclidean distance and the dimensionality reduction of discretization.

7.4.1 Description of alSAX

One of the most widely used methods for modelling non-stationary data is to use a sliding window. The idea is that the window of data is relatively stationary (constant μ and σ) and therefore those properties can be exploited. This method is already used in implementations of SAX where a sliding window shifts by an observation at a time and performs the discretization; however, this is generally not how SAX has been utilized in the literature. Nevertheless this procedure, given an appropriate window size, facilitates a discretized representation that accommodates non-stationary data. A caveat to this approach is that any one data point is represented by various symbols and therefore the data compression and dimensionality reduction benefits are lost. Additionally, the negative impact of periodic departures from Gaussian is still present. These departures invalidate the assumption of equal probability of the each symbol occurring, which as discussed, is a desirable property of discretization algorithms.

The alSAX algorithm accommodates both of these properties by locally fitting a Gaussian distribution to the data with non-overlapping windows that vary in size based on a goodness-of-fit test for a normal distribution. Figure 7.8 illustrates the intuition behind the proposed method, where a locally fit Gaussian will be more likely to match primitive shapes in a non-stationary series. Each data point is also only represented by one symbol and therefore the dimensionality reduction and data compression properties are preserved. It is also worth noting that from a financial perspective this approach is also quite appealing since investors do not consider the entire price history of an asset when

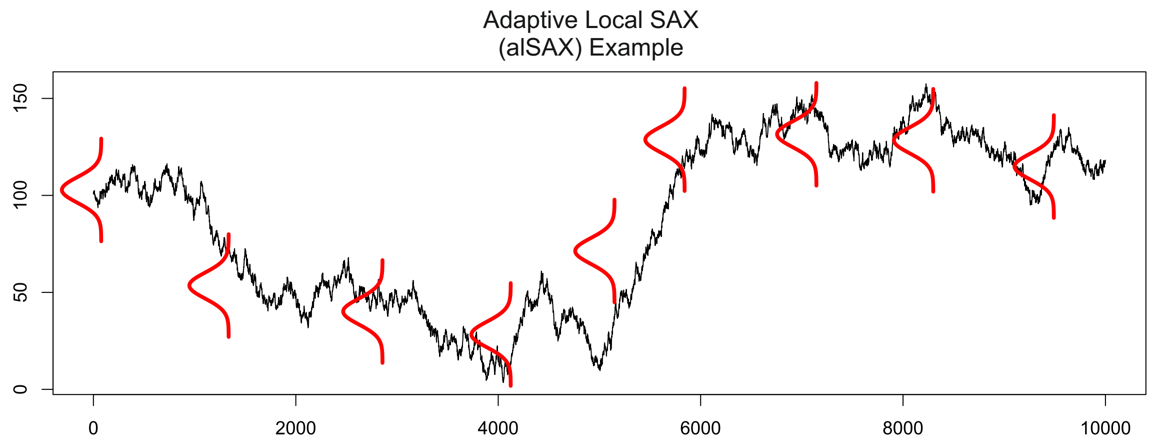

Figure 7.8: A depiction of locally fitted Standard Normal Distributions that allow for more primitive shapes to be matched by the SAX distance metric even though the data is non-stationary.

making decisions. For example, a latent piece of information concerning a financial asset is the 52-week high and low price, where current prices in relation to a local distribution are considered more relevant.

The decision as to the size of the sliding window is the only added parameter of alSAX. If we assume that having fewer SAX discretizations to perform is preferred then we will want to maximize the size of each window. Since there are two characteristics to consider (non-stationarity and non-Gaussian) we will want the largest window size that provides a data set which is stationary and normal. From the results in chapter 4.3.3 we can determine that choosing a window based on stationarity will not be useful since the standard unit-root tests are not sensitive enough for financial time series for sub period analysis. Another choice for the window size could be the largest window that maximizes the number of samples that accept the null of the Jarque-Bera (JB) test for normality. SAX assumes that the series is normal, which financial data is not, but using an appropriate sample size this assumption may be correct. We now perform a simple experiment that analyzes different window sizes for the data and tests for normality and returns the percentage of windows that accept the null of a normal distribution. The results reported in table 7.9 and figure 7.9 for the S&P 500 market index demonstrate that smaller windows are more likely to accept the null of a normal distribution. These results are informative but are not sufficient to make a decision. The window sizes need to be large enough for interesting patterns to be constructed and this has to be facilitated given that the PAA step will reduce the number of points within a window. For these reasons, simply choosing the smallest window size (window length of 25) would not be desirable. Thus a trade-off exists between the property of a normal distribution and the ability to construct meaningful patterns. For this reason we propose using an adaptive approach that maximizes the size of a window by iteratively

Table 7.9: The results from testing for a normal distribution within a sliding window using the Jarque-Bera test for increasing window sizes. The reported results represent the percentage of windows which accept the null of a normal distribution.

Window Size	P-value
25	0.9729
50	0.8667
100	0.5892
250	0.1199
500	0.0180
1000	0.0000

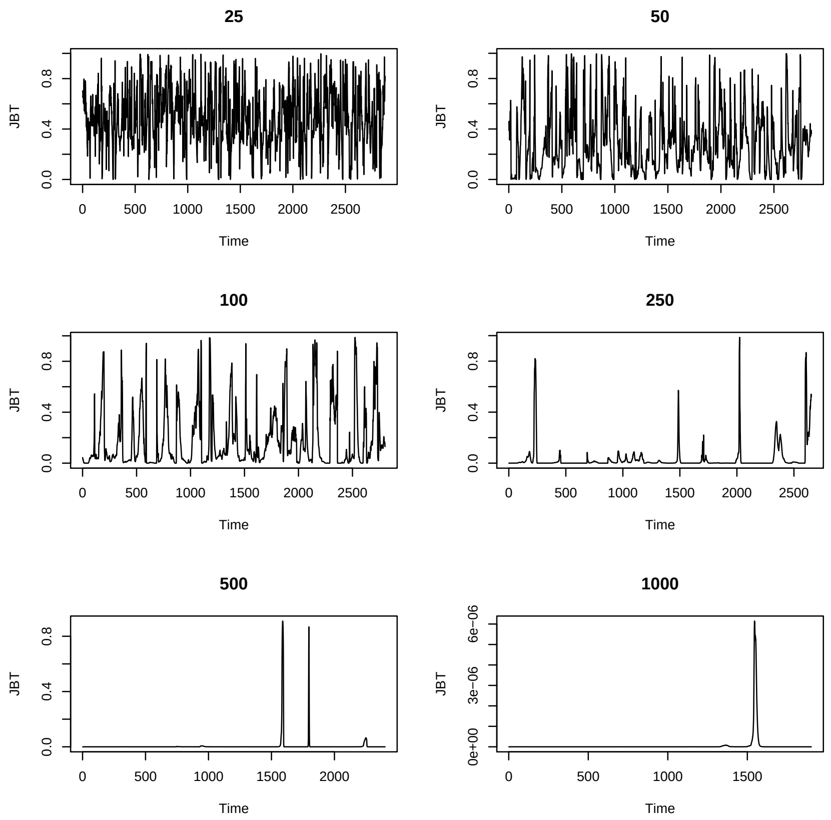

Figure 7.9: Plots of the p-values from performing the Jarque-Bera test for normality on data within a sliding window of increasing lengths.

growing it until the null of the Jarque-Bera test is rejected.

From the above discussion we can formalize the steps of alSAX in algorithm 8, where JB(d) is the Jarque-Bera test of data set d and d is allowed to increase in size as long as the p-values from the test ($JB_{p-value}$) continue to be larger than the α level of significance.

Algorithm 8 alSAX

```
Input alphabet size $\rightarrow A$, partion size $\rightarrow P$, time series $\rightarrow D$
Input minimum window size $\rightarrow minWindow$,
Input p-value for Jarque-Bera Test $\rightarrow \alpha$
for i in 1 to (length(D)-minWindow) do
    set window size $S$ = minWindow
    Create subsample $d \leftarrow$ D[i:(i+S)]
    JB($d$) $\rightarrow JB_{p-value}$
    while $JB_{p-value} > \alpha$ do
        $d \leftarrow d$+D[i:(length($d$)+1)]
        JB($d$) $\rightarrow JB_{p-value}$
    end while
    normalize $d$ to have $\mu = 0$ and $\sigma = 1$
    convert the time series to PAA using $P$
    substitute PAA segments for symbols from an alphabet of size $A$
    i $\leftarrow$ i+(length($d$)-(length($d$) % P))
end for
```

7.4.2 Experiment Setup

The motivation behind alSAX is to have a locally fit Gaussian and an adaptive window size to maximize the number of points within a sliding window. To test the effectiveness of alSAX a series of experiments are performed on different simulated and real-world data sets for different parameter values for the minimum window size, alphabet cardinality and the size of the partitions for the PAA segmentation. In the experiments we consider the canonical form of SAX, the proposed alSAX algorithm and a simplified version of alSAX that does not use an adaptive window, denoted loSAX or local SAX. Comparing alSAX with just a locally fit Gaussian of static size will allow us to gauge if the Jarque-Bera test offers any advantages. The results are compared using the chi-squared (χ^2) goodness-of-fit test (equation 7.2) under the assumption that a uniform distribution of the symbols is the desired outcome. The algorithm which produces an ex-post symbolic distribution closest to uniform will be deemed the most effective. The parameter settings for each experiment are listed in table 7.10 and details of the data sets are in table 7.11. The simulated data was generated using equations 5.21-5.23.

7.4.3 Experiment Results

The results from the aforementioned experiment setups (table 7.10) are displayed in tables 7.12 to 7.15. Reported is the absolute deviation from the target uniform distribution, any results which were not rejected by the χ^2 test are marked with an asterisk (*). If the

Table 7.10: The parameter settings for the SAX algorithms for the various experiments.

Exper.	|Alphabet|	PAA seg.	min. window
1	10	5	30
2	10	5	50
3	10	5	100
4	10	5	200
5	10	10	30
6	10	10	50
7	10	10	100
8	10	10	200
9	5	5	30
10	5	5	50
11	5	5	100
12	5	5	200

Table 7.11: Data description.

Dataset	Description	Length	JB test
SimData1	A independent realization of equations 5.21-5.23	10000	< 0.001
SimData2	A independent realization of equations 5.21-5.23	10000	< 0.001
S&P 500	Daily closing prices of the S & P 500 index	8074	< 0.001
IPC	Daily closing prices of the Mexican IPC index	5034	< 0.001
XOM	Daily closing prices for Exxon Mobil	8075	< 0.001
MMM	Daily closing prices for 3M	8075	< 0.001

χ^2 test accepted the null then the realized symbolic distribution from the discretization algorithm was statistically indistinguishable from uniform.

7.4.4 Results Analysis

To begin, if we consider the comparison between the local fitting approaches (alSAX and loSAX), there are 72 cases reported in tables 7.12 to 7.14. In approximately 85% (61/72) of the cases the alSAX algorithm produces a smaller absolute deviation from the uniform distribution. Additionally, the average absolute deviation across all data sets for alSAX is roughly half that of loSAX, where the absolute deviations are 12.628 and 25.170 respectively. If we consider the 11 cases where loSAX was superior the average difference in the absolute deviations between loSAX and alSAX was 1.577, however in the reverse the difference was several magnitudes larger at 15.274. Thus, the adaptive window was able to collect data that adhered to a normal distribution more effectively and therefore produced superior ex-post symbolic distributions. Focusing on the results in table 7.15 for the canonical SAX algorithm we observe that the ex-post symbolic distributions were

Table 7.12: Experiment Results for SimData1 and SimData2.

	SimData1		SimData2	
Exper.	loSax	alSAX	loSax	alSAX
1	36.4964965	13.65682841	33.93393393	10.07007007
2	27.53768844	16.04802401	27.63819095	14.65732866
3	15.65656566	18.34917459	18.48484848	14.60413515
4	19.59183673	14.92717228	19.3877551	17.07808564
5	74.53453453	22.1021021	76.93693694	13.13313313
6	36.38190955	24.52261307	38.3919598	16.31631632
7	25.25252525	23.70221328	22.62626263	23.93541877
8	24.28571429	15.67839196	20.40816327	19.55645161
9	22.36236236	3.541770885*	20.06006006	4.584584585*
10	9.648241206	7.943971986	11.15577889	7.043521761
11	9.090909091	7.243621811	4.141414141*	6.434694907
12	16.2244898	12.61677549	11.02040816	12.69521411

Table 7.13: Experiment Results for the S&P 500 and the IPC.

	S&P 500		IPC	
Exper.	loSax	alSAX	loSax	alSAX
1	33.92812887	8.996282528	37.56487026	11.45129225
2	26.70807453	10.35935564	25.60000000	11.92842942
3	16.62500000	10.03717472	13.60000000	13.91650099
4	12.87500000	12.60188088	14.00000000	14.22492401
5	67.06319703	14.47335812	72.57485030	15.58648111
6	39.62732919	15.39033457	38.80000000	15.30815109
7	29.75000000	15.81164808	24.40000000	22.39043825
8	19.25000000	13.73433584	20.80000000	20.20283976
9	25.79925651	5.402726146*	23.07385230	5.248508946*
10	12.29813665	3.915737299*	14.40000000	6.242544732*
11	4.500000000*	2.403965304*	6.00000000*	11.80914513
12	7.750000000	7.021943574	6.80000000*	7.051671733*

Table 7.14: Experiment Results for XOM and MMM.

	XOM		MMM	
Exper.	loSax	alSAX	loSax	alSAX
1	35.66294919	11.02293862	32.71375465	10.64516129
2	27.57763975	9.522628642	29.9378882	14.04466501
3	15.37500000	13.41235184	18.1250000	11.35068154
4	13.50000000	13.12500000	17.3750000	16.23824451
5	65.3283767	17.91563275	62.60223048	17.71712159
6	43.47826087	14.83870968	41.24223602	21.93548387
7	25.75000000	21.09862672	33.00000000	19.95043371
8	15.25000000	16.25000000	18.50000000	17.85268414
9	26.41883519	4.488530688*	25.30359356	3.945409429*
10	13.78881988	3.992560446*	15.77639752	4.069478908*
11	6.50000000	3.593262633*	5.62500000*	4.758364312*
12	5.50000000*	7.000000000	10.37500000	10.65830721

Table 7.15: Experiment Results for SAX without a sliding window. The column headings indicate the cardinality of the alphabet (let) and the number of points used to construct the PAA segments (pts).

Dataset	10let/5pts	10let/10pts	5let/5pts
SimData1	52.47623812	53.35335335	35.25762881
SimData2	27.07353677	26.74674675	27.07353677
S&P500	68.8228005	68.20322181	42.30483271
IPC	79.04572565	79.44333996	48.62823062
XOM	74.64684015	75.39033457	31.1771995
MMM	70.48327138	70.35935564	35.1425031

further from uniform than alSAX for all 6 data sets and experiment setups. The SAX algorithm did outperform loSAX on 3 occasions but this was only for the simulated data and for real-world results both locally fitting approaches produced ex-post symbolic distributions closer to uniform.

From these results we can conclude that taking into account the non-stationary nature of a financial time-series and its periodic departures from Gaussian improves the discretization process from real-values to a finite alphabet. The use of an adaptive window based on a goodness-of-fit test for a normal distribution also improved the ex-post symbolic distributions. Additionally, the locally fit Gaussians will facilitate closer matches of similar patterns emerging in a non-stationary time-series that occur in different regions of the normal PDF. Finally, the alSAX algorithm preserves the canonical SAX distance function and therefore distances defined on the alSAX symbolic mappings will lower bound Euclidean distance.

Table 7.16: Experiment results for SAX and alSAX for the subsequence matching task.

	Subsequence Length			
	4	6	8	10
SimData1	0.139 0.447	0.069 0.441	0.036 0.435	0.020 0.435
SimData2	0.140 0.432	0.063 0.434	0.035 0.428	0.016 0.414
SP500	0.159 0.459	0.075 0.452	0.038 0.453	0.020 0.443
IPC	0.156 0.518	0.084 0.530	0.044 0.510	0.024 0.526
XOM	0.145 0.489	0.069 0.484	0.033 0.486	0.018 0.464
MMM	0.138 0.462	0.063 0.454	0.028 0.471	0.013 0.464

7.4.5 Subsequence Analysis

In the last section we established that alSAX is superior to the canonical form of SAX for non-stationary data with respect to producing a symbolic mapping closest to uniform. Next we investigate how accurate alSAX is for the popular subsequence analysis task. The subsequence matching task can be formalized as follows: we have a target subsequence Q (called the query sequence) and a time series data set T which is mapped to a discrete representation P, where generally $|Q| << |P| << |T|$. The task is then to find all subsequences of P that are similar to Q, where all possible subsequences of P are queried. The reported result from such a task is the average number of matching results from sequence P, where fewer matches are generally preferred. The settings for SAX and alSAX were the same as experiment 9 from table 7.10. These parameter settings were chosen as they produced symbolic mappings statistically indistinguishable from uniform for alSAX.

In table 7.16 we report the results from performing a subsequence matching task on the 6 data sets in table 7.11. For each data set and subsequence length the results are averaged over 500 randomly chosen subsequences and only exact matches are considered sufficiently similar. The degree of similarity (or distance) between two subsequences is determined using the SAX distance measure (equation 2.20).

From the results reported in table 7.16 we observe that canonical SAX matched random subsequences at very high rate, where on average it found similar matches in 46.4% of the subsequences considered. This highly undesirable result is due to the non-stationarity of the data. In figure 7.10 we display SAX mappings for three of the financial time series under study. From these plots it is quite clear that the non-stationary nature of the data causes canonical SAX to overly smooth the data and much of the information is lost. These mappings result in extended durations of the same letter being produced one after another, which leads to the high subsequence match rates. Contrasting this result with

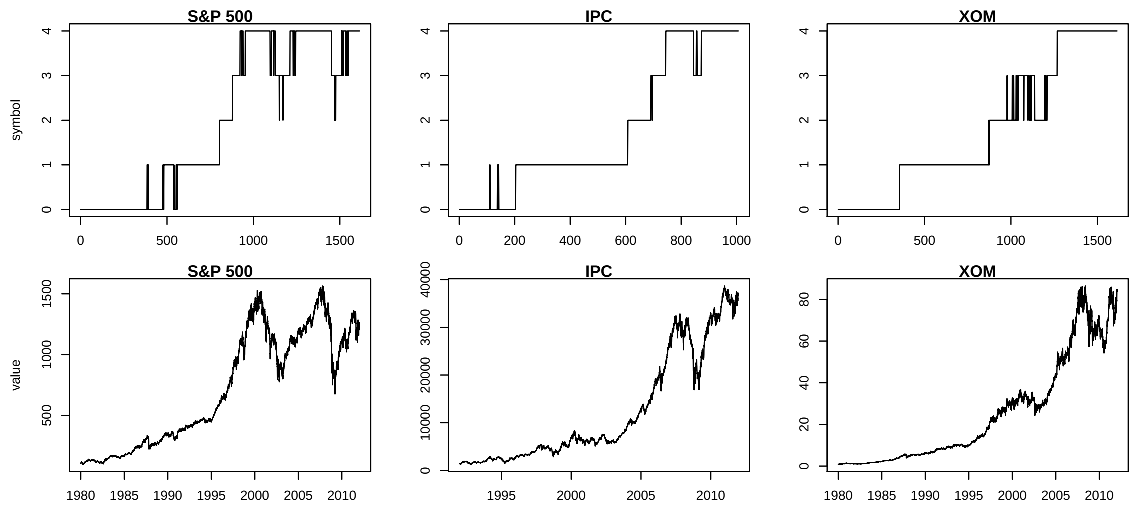

Figure 7.10: Plots of the SAX mapping for the S&P 500, IPC and XOM time series (top) and their original series (bottom).

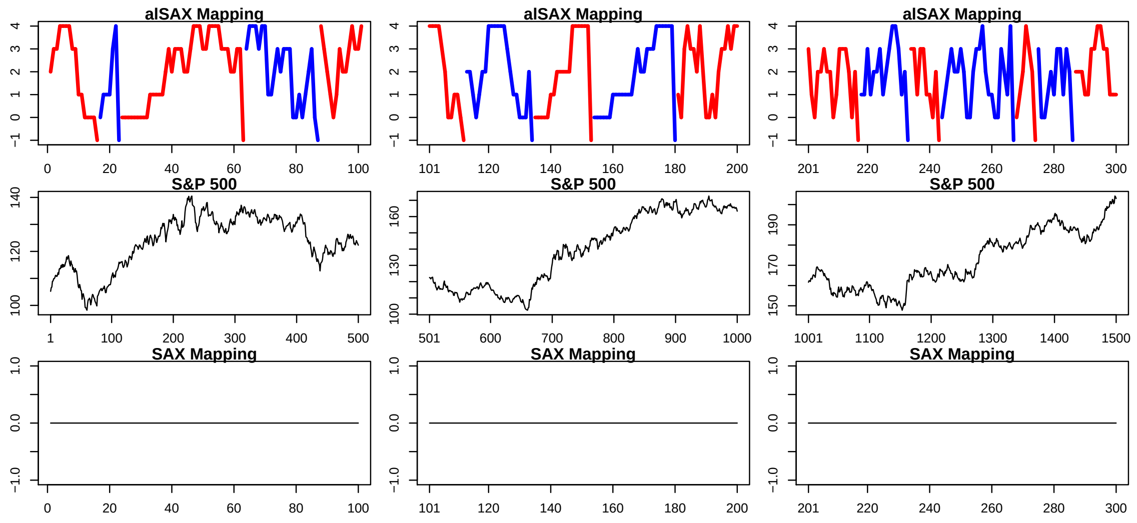

Figure 7.11: Plots the first 1500 points of S&P 500 index (approx. 6 years) and the alSAX and SAX representations of the same time period.

alSAX, we observe that on average only 6.8% of the subsequences considered matched the query sequence. To aid the analysis, and help explain why canonical SAX performed so poorly, figure 7.11 plots the first 1500 points of S&P 500 index (approx. 6 years) and the alSAX and SAX representations of the same time period. The SAX mapping labels the entire period as one symbol and all the dynamic behaviour during that time period is ironed out. This leads to a significant loss of information and without any added benefits for compression (alSAX only adds one extra place holder in the discretization per window). From the top three panels in figure 7.11 we observe that alSAX was able to capture significantly more information during these time periods and that despite the non-stationary nature of the time series was still able to extract interesting primitive shapes.

7.5 Chapter Summary

In this chapter we highlighted an invalid assumption of the SAX algorithm and demonstrated why the assumption is invalid. We also showed under what conditions this invalid assumption will negatively impact the symbolic distribution and a fix to counteract these problems. We have also proposed a novel algorithm which extends SAX to handle financial data. The proposed algorithm, alSAX, was demonstrated to produce superior ex-post symbolic distributions over other alternatives by making allowances for the generally non-stationary nature of financial time series and its periodic departures from a Gaussian distribution. We also tested alSAX in the popular time series task of subsequence analysis, where the results also demonstrated alSAX to be superior in realtion to canonical SAX.

We are also delighted to report that the invalid core assumption of SAX has been reviewed and confirmed by Dr. Emmon Keogh (the original co-author of SAX) and his comments regarding the work are:

“I very much like your paper. While there are several hundred papers on SAX, your observation is novel. Moreover you explanations are clear, and your experiments unimpeachable.”

Part III

Conclusions

Chapter 8

Conclusions and Future Work

This thesis has presented original research that concerns the overlap of the fields of computational intelligence and econometrics. The objective of the thesis can be generalized as an exercise in improving how computational intelligence is used for modelling and forecasting financial time series. From a holistic perspective, the thesis has considered the important problems encountered when modelling a complex system, i.e., how to characterize the system at a meta level, what are the specific and unique implications it holds which differentiate it from other systems of similar complexity, and how this knowledge can be utilized as a means for novel algorithm development and improvement.

This endeavour commenced with an in depth analysis of the complex system of interest, with a specific focus on characterizing the behaviour of the system in way that was meaningful for computational intelligence. To achieve this objective a thorough review of the related work was conducted, followed by the reproduction of previous results and finally the proposal, implementation and analysis of novel tests for dynamic behaviour in the financial markets.

Once a reasonable characterization was determined, the thesis explored how this characterization impacted machine learning. Experiments were designed, implemented and performed that concerned a spectrum of supervised learning and nature inspired population based optimization algorithms. The results demonstrated that the specific implications, drawn from the assumed characterization of the complex system, did in fact have a non-trivial impact on the robustness of models derived from computational intelligence techniques. Thus the result suggests that taking into account the behaviour of these dynamic characteristics will improve model robustness.

Equipped with an improved knowledge of the complex system and how its unique features affect computational intelligence, the thesis focused on the design, implementation and analysis of novel algorithms. The implication of cyclical effectiveness was utilized to create an online adaptive algorithm for financial forecasting. The unique framework and processes of the algorithm were directly influenced by the cyclical effectiveness results which demonstrated that models will be effective after the time period they were initially created for. Additionally, a financial time series discretization algorithm was proposed that was based on the non-stationarity of financial time series and the periodic departures it experiences from a Gaussian distribution. The proposed algorithm was able to achieve better results then it's predecessors in terms of mapping a real-valued time series to the target discrete uniform distribution and in subsequence matching, a common task for discretized time series.

8.1 Summary of Thesis

This section will provide a summary of the each of the contribution chapters in the thesis.

8.1.1 Chapter 4 - Validity

In this chapter we examined if the adaptive market hypothesis was a valid representation of the financial markets. This examination was facilitated through tests of measureable characteristics of financial time series. We examined if the behaviour of the financial markets was indicative of one market theory or another. In the first case the results from variable efficiency were confirmed as a means to validate an algorithm that was to be utilized in chapter 5. The results confirmed that the code was implemented correctly and that in the financial markets considered variable efficiency was present, in the informationally efficient sense of the phrase. Variable efficiency was represented by statistically significant non-linear correlation in the time series and its discovery demonstrated that periodic opportunities for improving trading models existed.

In the next section we considered the AMH implication of a time-varying risk to reward relationship in the financial markets. Specifically we analyzed the risk-to-reward relationship between individual financial assets and the market index. This analysis was realized through the lens of a novel dynamic model that was a generalization of a traditional linear investment model, the CAPM. The results demonstrated that the relationship between an asset and the market index (referred to as its β value) is heterogeneous amongst the variety of asset types considered in the study. In some cases the relationship was time-varying and therefore, was evidence in favour of the AMH, but for others the relationship was static throughout the analysis period as would be the case if the markets were efficient. However, the majority of the assets considered did exhibit variation in their β values and because the AMH is not invalidated by a static relationship, the evidence strongly supports the AMH alternative to market efficiency.

Finally, the generally held view that financial time-series is non-stationary was challenged. The aim of section 4.3 was to determine if this characteristic was also time-varying. An allowance for this alternative was provided by the rejection of the random walk hypothesis by Lo [95] and the results in section 3.1.1 which showed the dynamic behaviour of the variance ratio test for random walks. The experiment design considered an approach based on standard unit-root tests that were applied iteratively using a sliding window. The results from the approach revealed that the majority of financial markets are generally non-stationary but experience periodic departures to trend stationarity. This discovery demonstrates that the assumption of non-stationarity is not always valid and, dependent on the time interval under consideration, a time series can be either non-stationary or trend stationary. This has implications for modelling and forecasting financial time series for any model that a priori assumes a stationary process (i.e. a stationary model). To model non-stationary data with a stationary model the data needs to be pre-processed and the correct pre-processing methodology is dependent on the whether the series is non-stationary or trend stationary. This discovery thus demonstrates that by sampling the local time series for the existence of a unit-root will improve stationary model robustness.

8.1.2 Chapter 5 - Implications

In the implications chapter we explored what effects, if any, the validity of the AMH had on trading and investment models derived from computational intelligence techniques. This began with variable efficiency and its effect on the classification accuracy of supervised learning algorithms. The assumption being, that the detection of non-linear dependence in a time series, implies that higher accuracy should be expected. The experiments concerned six supervised learning algorithms from four machine learning paradigms and a simulated GARCH(2,2) process created to mimic real-world financial data. The results demonstrated that during time periods when non-linear dependence is detected in the time series, a statistically significant increase in classification accuracy can be expected. Additionally, the results suggest that trading models derived from machine learning algorithms should also be more profitable during times of non-linear dependence since profitability and accuracy are strongly positively correlated [81].

Secondly, we considered how variable stationarity affected modelling and forecasting financial time series with ANNs. The results from experiments with simulated and real-world time series demonstrated that superior forecasts could be achieved by taking into account the dynamic behaviour of stationarity. Additionally the experiments also refuted previous work concerned with modelling non-stationary data with ANNs.

Finally, the implication of the waxing and waning profitability of investment strategies was examined. First, we proposed a metric called cyclical effectiveness for evaluating this implication that was robust to the dynamic nature of the financial markets. This metric was then used to test the cyclical effectiveness of a hybrid trading model that was a blend of PSO and a popular technical indicator. The hybrid trading model was also proposed in this chapter and methods for training and implementation were provided. The results demonstrated that cyclical effectiveness was present in these models which implies that the models would be effective again after the time period they were initially developed for. Thus, this result demonstrated that the AMH implication of cyclical effectiveness was valid for computational intelligence models.

8.1.3 Chapter 6 - Innovations in Technical Analysis

Chapter 6 proposed, implemented and tested a novel machine learning algorithm that was a blend of population based optimization and reinforcement learning. The motivation for the design of the algorithm, called LATIS, was based on the cyclical effectiveness results which highlighted a meta learning opportunity for combining newly created and pre-existing models. The resulting framework for LATIS allowed for a blend of micro and macro modelling perspectives that combined a meta learner with a population of optimized technical indicators.

The benefits of the proposed algorithm were demonstrated on simulated and real-world data sets, where a thorough Monte Carlo simulation showed that the LATIS algorithm was able to combine the heterogeneous set of signals produced by a population of indicators into a coherent trading strategy that outperformed the vast majority of the indicators in the LATIS population. These results not only show the LATIS could determine when to trust the signals from any given indicator but that the implication of cyclical effectiveness could be used to derive a machine learning algorithm for modelling financial time series. The implementation used two technical indicators from the financial literature known as a Bollinger band and a moving average indicator, where the parameters of the indicators were fit to a subsample of data using the recently introduced dHPSO algorithm that was extended to handle a multi-objective optimization.

8.1.4 Chapter 7 - Innovations in Discretization

In the general area of time series analysis there is a strong interest in discrete representations of real-valued time series. As discussed in chapter 7 there are several benefits to discretization, such as, dimensionality reduction, the removal of noise and availability of machine learning algorithms that require discrete data. While our original motivation for using discretization was from an econometric perspective, the research has lead to a more important insight which we believe deserves to be shared with the research community. This refers to the opening section of chapter 7 where an invalid core assumption of the SAX algorithm was revealed. We demonstrated the negative effects of the PAA step of the SAX algorithm on the ex-post symbolic distribution on a variety of real-world data sets. The relationship between the effect and the autocorrelation function of a time series was also revealed using standard statistical tests. Finally, a case study was described that demonstrated how the effect of the PAA step could be taken into account for data mining. The results from the case study showed a statistically significant increase in classification accuracy for the nearest neighbour algorithm as well as a speed up in the execution time.

The second half of the chapter implemented and tested a novel algorithm for discretizing financial time series. The algorithm, called alSAX, is an extension of the SAX discretization algorithm, which takes into account the problematic characteristics of financial time series, i.e. being generally non-stationary with periodic departures from Gaussian. The alSAX algorithm was tested on simulated and real-world data sets and the ex-post symbolic distributions were gauged in relation to a gold standard. The results showed that the alSAX algorithm produced superior ex-post symbolic distributions is relation to the canonical version of SAX and a simplified version of alSAX.

8.2 Aims Revisited

Having summarized the contribution chapters of the thesis, we now revisit the aims or research objectives. In general the thesis was positioned as an exercise in evaluating and improving computational intelligence for financial modelling. More specifically, the aims or goals of the thesis can be encapsulated by the three research questions outlined in the introduction:

- *W*hat is a reasonable characterization of the complex system we are modelling?

This question was approached in the related work in chapter 3 and the first contribution chapter, chapter 4. The general conclusion from the related work is that a consensus is not held amongst financial practitioners or academics as to what governs the behaviour of the financial markets. There is however an increasing body of evidence that suggests the efficient market hypothesis is not valid. Specifically, this concerns the work presented on behavioural finance and the adaptive market hypothesis. The lack of informationally efficiency is also observed during the market bubbles and subsequent crashes that have occurred in developed and emerging markets around the world. Thus, the characterization initially assumed by the thesis was that the financial markets are dynamic and evolving and therefore a hypothesis which made allowances for this behaviour was adopted.

This position was further explored in the chapter 4. Evidence of market inefficiencies were confirmed from the econometrics work, which demonstrated that a times, markets were informationally efficient as implied by the EMH but at others non-linear dependencies existed and thus informational efficiency was lost. The argument for using the AMH as the basis for the market characterization was also strengthened by the results demonstrating a time varying risk-to-reward relationship in metals and the results from examining non-stationarity, which showed that this characteristic was also dynamic.

In light of the dynamic nature of the many market characteristics considered, the AMH presents a more plausible representation of market behaviour. The AMH also provided measureable characteristics that can be used to update investment and trading models as the market environment changes. Thus in summary, from a computational intelligence perspective, the AMH offers a reasonable characterization of the systems behaviour upon which further studies can be based.

- *W*hat implications does this hold for computational intelligence?

The AMH provided the basis for measurable characteristics which could be analyzed within a sound research framework. In chapter 5, three implications were isolated and analyzed for their affect, if any, on learning from time series data with supervised machine learning and nature inspired optimization algorithms. The analysis concerning variable efficiency showed that in periods where non-linear dependence was detected the algorithms experienced an increase in classification accuracy. Additionally, the analysis of variable stationarity, demonstrated that artificial neural networks benefit from appropriate pre-processing based on the existence or lack thereof a unit-root. Finally, trading models developed from particle swarm optimization exhibited cyclical effectiveness, as implied by the AMH, and therefore models would be effective again after the time period they were trained for.

From the results we can conclude that the implications of the AMH held non-trivial consequences for computational intelligence algorithms and that by making allowances for the dynamic nature of the financial markets, the forecasting performance of the computational intelligence derived trading models can be improved. Thus in summary, the implications of the AMH do effect algorithms from computational intelligence and local characteristics of a financial time series should be considered to improve performance.

- *W*hat improvements can be made to compensate for these implications?

The thesis presented improvements and innovations for computational intelligence algorithms in chapters 6 and 7. In chapter 6 an algorithm was developed for online and adaptive learning based upon the implication of cyclical effectiveness. The proposed algorithm described a meta learning approach that allowed the traditional interpretation of technical indicators to be preserved. The algorithm detailed a framework and learning procedure for training an initial population of individuals and how to utilize those models in an online learning context given that they exhibit cyclical effectiveness.

In chapter 7, a popular discretization algorithm was extended to handle the non-stationary nature and periodic departures from a Gaussian distribution exhibited by financial time series. The proposed algorithm was able to improve the performance in relation to a gold standard and demonstrated how the AMH implication of dynamic characteristics can be utilized in algorithm development. Indirectly, two other improvements we made in the implications chapter, where the results from variable stationarity and variable efficiency demonstrated that sampling the local characteristic of the time series could improve the forecasts of several supervised learning algorithms.

The work presented in the thesis was not intended to be exhaustive but to offer improvements to existing algorithms and to propose new algorithms derived from the implications of the AMH. Thus, in summary, there are lots of improvements which can be made and how the information regarding the AMH and modelling with computational intelligence is used is up to the machine learning expert but in the thesis we have presented a few which seemed important.

8.3 Summary of Contributions

Here we provide a brief summary and recap of the contributions in the thesis and under which category they fall. The contributions are listed in increasing order of importance.

8.3.1 Financial Analysis

The following is a list of main contributions when the thesis is considered from a purely financial point-of-view.

1. The thesis reproduced results on variable efficiency from the financial literature.
2. The thesis revealed the variable nature of stationarity in financial time series as opposed to the commonly held view that it is constant.
3. The thesis demonstrated that variable efficiency is a non-trivial consideration for trading models.
4. The thesis revealed the time-varying nature of alpha and beta from the Capital Asset Pricing Model (CAPM) as implied by the adaptive market hypothesis .

8.3.2 Computational Intelligence

The main focus of the thesis was in the overlap of econometrics and computational intelligence but some of the contributions were more general in nature and are classified as additions to computational intelligence in general.

1. Extending dynamic heterogeneous particle swarm optimization (dHPSO) to multi-objective optimization (MOO).
2. Refuting published results on training ANNs on non-stationary data.
3. Discovering an invalid assumption of the Symbolic Aggregate approXimation (SAX) algorithm and providing potential corrections.

8.3.3 Computational Intelligence for Financial Analysis

Finally the contributions in the overlap between econometrics and computational intelligence are the following.

1. Demonstrating that artificial immune systems are a viable modelling technique for financial time series.
2. Development of a new technical indicator based on Bollinger Bands and particle swarm optimization.
3. Demonstrating the effects of variable stationarity on price level estimation of the artificial neural network.
4. Demonstrating the effect of variable efficiency of supervised learning algorithms .
5. Demonstrating the existence of cyclical effectiveness in computational intelligence derived financial models.
6. Developing an online learning algorithm for optimizing financial technical analysis.
7. Developing a symbolic discretization algorithm for financial time series.

With respect to item 6 which refers to the LATIS algorithm developed in chapter 6 the proposed framework is general enough to be applicable to other models for the population, such as classifiers, and other environments besides financial. From a general overview perspective, LATIS provides a framework for online learning in a dynamic environment where local models are developed for different behaviour. When the inputs to the population need to be qualified by other more meta considerations, LATIS provides a way to qualify when a model is reliable and for how long. From this perspective, the framework can be considered as a more general contribution to online and adaptive learning but the specific implementation and testing is more appropriately labelled as a contribution to computational intelligence for financial analysis.

8.4 Future Work and Concluding Remarks

The thesis has introduced a novel approach to financial modelling using computational intelligence techniques, where the application area has been thoroughly analyzed and used as the basis for algorithm innovations and developments. From the results and conclusions discussed in the contributions chapters, we now discuss the following areas that can be pursued for further development.

Having revealed and confirmed the dynamic nature of market characteristics and shown the effect on supervised learning algorithms, this information can now be used to improve forecasting systems. These characteristics have a non-trivial impact and can potentially be used to describe the current market state. This information could be used in ensemble methods or in meta learning, where at any one time there may be competing models to describe future behaviour of the system. Specifically, the variable efficiency results could be extended to real-world data and utilized in a similar manor as the filter approaches discussed in section 3.2.2.

The LATIS algorithm proposed in chapter 6 focused on only two technical indicators and did not include the covering step of the algorithm. This approach was taken to focus the results and experiments on cyclical effectiveness. The work can now be extended to include this step of the algorithm as a means to acquire new information about the environment. This work could also be extended through the development of more complex training strategies for creating the initial population. This research direction would focus on maximizing the out of sample effectiveness of the $I \in [P]$. The algorithm could also be extended to include additional technical indicators and meta information concerning the market environment, such as the characteristics discussed in chapter 4.

Finally, the revealed invalid assumption of the SAX algorithm could be considered in a wide variety of application domains and data mining tasks and any previous work concerning SAX could be revisited to determine if the analysis can be improved. How this new information is utilized is a decision for the data mining expert but at least now this information is available and has been proven to have a significant effect. The benefits of using the SAX filter have not been fully studied and using this new SAX approach in a large data base could assist in determining the exact benefits and under which circumstances the filter will lead to more desirable results. The alSAX algorithm which extends SAX to financial data could be further analyzed in conjunction with other data mining/machine learning algorithms and potentially other data sets that have similar properties (i.e., non-stationary and periodic departures from Gaussian).

Appendix A

Variance Ratios

In the original paper that introduced the Variance Ratios (VR) [95], the test statistics were created for assumptions of homoscedastic (RW1) and heteroscedastic (RW3) increments. Given that most economists would agree that the variance in financial markets is heteroscedastic a test statistic that signifies a departure from a random walk to due heteroscedasticity of the variance would not be of much interest. As the paper divulged the homoscedastic test statistic was more likely to reject the null of a random walk. As a result most studies that use the VRs for analysing weak-form efficiency focus on the RW3 test statistic. However in the thesis, results from both tests are reported in section 3.1.1.

The VRs exploit a property of random walk that the variance grows linearly in the observation interval, meaning the variance between X_t - X_{t+1} should be half the variance between X_t - X_{t+2}. Lets consider a time series $\{Y_t\}$ for t = $\{1 : nq\}$ observations converted to continuously compounded log-normalized returns $\{X_t\}$. Under the null of the random walk the following relationship holds asymptotically as the number of observations approaches infinity:

$$\bar{M}_r(q) \equiv \frac{\bar{\sigma}_c(q)}{\bar{\sigma}_a} - 1 \tag{A.1}$$

where q is the number of lags to consider and $\bar{\sigma}_c(q)$ and $\bar{\sigma}_a$ are unbiased estimators defined as:

$$\bar{\sigma}_a = \frac{1}{nq-1}\sum_{k=1}^{nq}(X_k - X_{k-1} - \hat{\mu})^2 \tag{A.2}$$

$$\bar{\sigma}_c(q) = \frac{1}{m}\sum_{k=q}^{nq}(X_k - Xk - q - q\hat{\mu})^2 \tag{A.3}$$

$$m = q(nq - q + 1)\left(1 - \frac{q}{nq}\right) \tag{A.4}$$

where $\hat{\mu}$ is the estimated mean of the sample. This ratio is not robust to heteroscedasticity but highlights the basis of the test statistic, where under the null hypothesis, equation A.1 evaluates to 0. This represnets a test of the RW1 hypothesis that assumes the increments are iid. The asymptotically standard normal test statistic is:

$$z(q) \equiv \frac{\sqrt{nq}\bar{M}_r(q)}{\sqrt{2(2q-1)(q-1)/3q)}} \tag{A.5}$$

To allow for more general forms of heteroscedasticity, Lo et al. [95] define the specific form of the test statistic as follows:

$$\bar{M}_r(q) \asymp \sum_{j=1}^{q-1}\frac{2(q-j)}{q}\hat{\rho}(j) \tag{A.6}$$

where $\hat{\rho}(j)$ is the autocorrelation coefficient estimator at lag j. Under the assumption of heteroscedasticity the estimators of asymptotic variance of $\hat{\rho}(j)$ and $\bar{M}_r(q)$ are respectively:

$$\hat{\delta}(j) = \frac{\sum_{k=j+1}^{n} q(X_k - X_{k-1} - \hat{\mu})^2}{[\sum_{k=1}^{nq}(X_k - X_{k-1} - \hat{\mu})^2]^2} \tag{A.7}$$

and

$$\hat{\theta}(q) \equiv \sum_{j=1}^{q-1}\left[\frac{2(q-j)}{q}\right]^2\hat{\delta}(j) \tag{A.8}$$

Using the equations above, the following test statistic can be computed to test for statistical significance:

$$z^*(q) \equiv \frac{\sqrt{nq}\bar{M}_r(q)}{\sqrt{\hat{\theta}}} \tag{A.9}$$

where if $z^*(q)$ is significant at $\pm$ 1.96 for 95% confidence interval.

Appendix B

AIS for Financial Forecasting

In chapter 5 the thesis explores the effect of non-linear dependence on the forecast accuracy of supervised learning algorithms. To maximize the generalization of the results the experiments include algorithms from each of the major paradigms within supervised learning. In that respect, all of the algorithms concerned, with the exception of the artificial immune system (AIS), had previously been studied and found to be effective when modeling financial time-series data. In the following paper the effectiveness of the AIS is determined, where its performance is gauged against an artificial neural network (ANN) and the *k*-Nearest Neighbors (kNN) algorithm. The results from the study revealed that the AIS was superior to the kNN algorithm and statistically equivalent to the ANN in terms of classification accuracy.

Modeling the Behavior of the Stock Market with an Artificial Immune System

Matthew Butler, Dimitar Kazakov

Abstract— **This study analyzes the effectiveness of an Artificial Immune System (AIS) to model and predict the movements of the stock market. To aid in this research the AIS models are compared with a k-Nearest Neighbors (kNN) algorithm, an artificial neural network (ANN) and a benchmark market portfolio to compare simulated trading results. The analysis shows that the AIS produced overall accuracy results of 67% over a 20 year test period and that the increased complexity of the model was warranted by the statistically significant superior results when compared to the simpler instance-based approach of kNN. The accuracy results were comparable to those obtained from training the ANN and the trading results outperformed the market benchmark, providing evidence that the stock market had a degree of predictability during the time period of 1989-2008. In general the practice of using the natural immune system to inspire a learning algorithm has been established as a viable alternative to modeling the stock market when implementing a supervised learning approach.**

I. Introduction

Evolutionary inspired algorithms are a popular learning technique applied to several financial modeling problems [1], [2], [3]. Their ability to work with highly non-linear and noisy data makes them a natural choice in solving the difficult prediction and optimization problems faced in the financial domain. Although this area has been popular the majority of the research in forecasting financial assets has been with genetic algorithms [4], genetic programming [5] and hybrids such as evolutionary neural networks. The group of algorithms inspired from the vertebrate immune system referred to as Artificial Immune Systems (AIS) has received very little attention in this area. These algorithms were initially used in unsupervised learning and as a result were not as appropriate for a lot of financial applications. This constraint has been lifted as several AIS algorithms have been developed for classification tasks [6], [7]. Much of the on-going research in financial forecasting could be generalized into the problems of patterns recognition and anomaly detection, both of which have been successfully attempted by AIS in other areas [8], [9]. Given the success of the somewhat related research endeavors with AIS, it seems appropriate to further investigate their abilities with market and other financial asset prediction. The AIS algorithms are instance-based learners which are not as popular in the research literature compared to other algorithms from supervised learning (SL), such as artificial neural networks (ANN) and support vector machines (SVM). Where an ANN will attempt to learn a global operator to generate predictions, the instance-based approaches look to identify situations which are similar to ones in the past. There is an increasing body of evidence which suggests that the returns in the stock market are not completely random. This body of work comes partly from the area of behavioral finance which attempts to prove that market efficiency, as described in the efficient market hypothesis [10], is not the underlying factor which governs market behavior. Research in behavioral finance is aimed toward explaining market behavior by combining the fields of psychology and economic theory. Behavioral finance has provided evidence to explain certain market phenomena such as the disposition effect [11] and the inefficient and slow integration of news into current market prices [12]. A definition of market predictability could be that if markets exhibit behavior that can be identified as similar to previously observed and such behavior results in a similar outcome than it is predictable. Under such a definition it seems logical that an algorithm which determines similarity between events, such as instance-based learners, would be appropriate for modeling such behavior, thus taking advantage of the specificity bias in instance-based learners.

This paper will apply a popular AIS algorithm for classification to the task of predicting monthly stock market movements. Specifically the algorithm will be training in a supervised learning context with tuples of information containing macro-economic data and its effect on market movements. The algorithm will be attempting to learn the highly non-linear relationship that exists between the performance of the market and measureable variables of the state of the economy. The algorithm used in this study is AIRS [6] which uses an instance-based representation and is inspired from clonal selection theory of acquired immunity. The algorithm will be compared three fold, first to another lazy-learning learning technique namely, k-Nearest Neighbor, secondly to an artificial neural network, and finally to a traditional investment model. The final benchmark is to add context to the results in terms of market performance which will assist in judging the real-world applicability of such a technique.

Manuscript received February 7, 2010.

M. Butler is with the Artificial Intelligence Group at the University of York, York, YO10 5DD (email mbutler@cs.york.ac.uk).

D. Kazakov is with the Artificial Intelligence Group at the University of York, York, YO10 5DD (email kazakov@cs.york.ac.uk).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

II. Overview of AIS

This section will outline the major principles behind the machine learning approach based on the immune system. The Artificial Immune System is a biologically inspired algorithm which draws its inspiration from the vertebrate natural immune system (NIS). The NIS can be generalized as a system which continually protects the body from harmful invaders, called antigens, and keeps the body in an equilibrium state. The AIS is not an exact model of how the NIS interacts with a living entity; rather, it draws on principles from the immune systems which are a natural fit for machine learning. The notion of "self" and "not self" is one of the more popular principles employed in AIS algorithms; it means that the immune system is able to recognize objects which are not harmful, such as red bloods cells, categorized as "self", commonly referred to as antibodies and antigens which are a threat to the wellbeing of the system, such as viruses. This principle is the main underlying theme to AIS for supervised learning, although the algorithms go into much more detail and employ other principles from the NIS to accomplish this goal; in all cases, the notion that the immune system can effectively distinguish between two distinct objects is the very basis of using it for an analogy in machine learning. In the NIS, if an antibody is activated by an antigen, then the two bind and the antigen is destroyed, eliminating the threat. How the immune system actually accomplishes this task forms the principles which influence the inner workings of AIS algorithms.

Negative selection (NS) was first used in AIS for recognizing "self" and "not self", with this type of learning the algorithm only uses examples of one-type of object such as in positive-only learning. NS comes from the Thymus which is an organ responsible for generating the T-Cells which circulate the body looking for invading antigens. The thymus continually creates T-cells which are first held in the thymus and tested to see if they are activated by any of the "self", which would mean that they recognize and react to "self", if they do than they are destroyed, otherwise released into the body as they will only be activated by "non-self" pathogens. A T-cell is activated if its degree of similarity is sufficiently close to an antigen and this degree of similarity is determined by an affinity measure. This type of learning can be very ineffective for supervised learning as the "non-self" space can be quite large and require a massive number of T-cells to accurately map it. Also by ignoring counter examples and only training on one type of data a large amount of useful information is ignored. To improve upon the principle of NS, the algorithm implemented in this study, which will be more thoroughly introduced in the next section, is referred to as clonal selection-based AIS where training is conducted with both negative and positive exemplars. The algorithm learns and builds a memory of negative and positive exemplars and later uses this experience to classify new antigens which enter the system or, representing, new, unlabelled examples to which the algorithm is exposed.

III. Related Work

In [13], where the k-Nearest Neighbor (kNN) algorithm has been used in financial time-series prediction, the authors state the motivation for using such a method is derived from the non-stationary nature of financial time-series. The higher level of complexity in that data creates problems for artificial neural networks to build a global operator to capture reoccurring patterns. The kNN algorithm was also explored in [14] for determining index predictability of the Warsaw Stock Exchange. The kNN algorithm was the top-performer, in terms or overall accuracy of predictions, in comparison to an ANN, GA-evolved logic programmers and Naive Bayes. Once again the author concluded that the superior performance could be related to the non-linear nature of financial data and that generated global methods to explain market prediction may contain too many problems. Related work with Artificial Immune Systems includes research [15] where an AIS was attempting to predict the performance of a bank over the coming year in the Taiwanese Banking Industry based on financial ratios. The algorithm was compared to other methods from similar research which included neural networks trained with a genetic algorithm or back-propagation, case-based reasoning, logistic regression analysis and quadratic discriminant analysis. The results were reported for overall accuracy and the AIS system, which was based on a resource limited AIS [16], generated the best results with a hit ratio of 97.30%. Along the same lines the authors in [17] used a hybrid-AIS algorithm to predict bankruptcies among Indian companies based on commonly reported financial ratios. The method employed several immune system analogies such as negative/positive selection and clonal selection. The results are compared with two statistical methods, the Altman Z-score and Emergent Market-score, and over the three test periods the immune inspired algorithms were the top performers. The authors found that while using r-continuous matching to determine similarity, the positive selection algorithm was the most effective. The work in [18] was directed at detecting abnormal fluctuations in stock prices in order to improve risk management of the stock market. The method was based on negative selection with r-continuous matching for determining affinity and a novel risk evaluation function was proposed to assist in determining if an unknown antigen was an anomaly or not.

IV. Algorithm Overviews

This section will introduce and provide an overview of the algorithms implemented in this study.

A. *Artificial Immune Systems*

The Artificial Immune System developed for this work was based on AIRS [6], where a set of memory cells are evolved during training by stimulating and mutating a population of Artificial Recognition Balls (*ARB*s). Although the implementation is based on AIRS it might be different from existing AIRS implementations in certain details. A high-level algorithm that is used to evolve a set of memory cells is as follows[1]:

1. *Data Normalization and Initialization*
    a. *All data is normalized between 0 and 1.*
    b. *L antigens are chosen from the data set to seed Artificial Recognition Ball (ARB) and Memory Cell (MC) pools, where L>0.*
2. *For each training vector in the dataset, do:*
    a. *MC identification and ARB generation.*
    b. *Competition for resources and development of a candidate memory cell.*
    c. *Memory cell introduction.*
3. *Step 2 is repeated for each of the training exemplars in the training data.*
4. *Classification of test data:*
    a. *Apply kNN algorithm for K>0 to determine local neighbourhood of mcs.*
    b. *Classify training vector by majority vote.*

1) Parameter and data structure description

This section will introduce some terminology and discuss the parameters which had the greatest effect on the algorithms performance in terms of accuracy for this implementation:

- Clonal rate: a variable in the equation which determines the number of mutated clones a given *ARB* or Memory Cell is allowed to produce.
- Mutation rate: The probability, set between 0 and 1, that any one input attribute within an *ARB* will be mutated, excluding the class attribute of that *ARB*.
- Stimulation Threshold: A parameter used to determine the stopping criterion for training on a particular antigen, set between 0 and 1.
- Stimulation Threshold Scalar: A value between 0 and 1 that when combined with the average affinity value (AAV) among the training antigens, given in equation [1], determines the cut-off point for *MC* replacement.

$$AAV = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} affinity(ag_i, ag_j)}{\frac{n(n-1)}{2}} \qquad [1]$$

where n is the number of training exemplars, ag_i, ag_j are the *i*th and *j*th training exemplars and the affinity (x, y) is the Euclidian distance between feature vectors, equation [4].

- Hyper mutation rate: Another variable in the equation (clonal rate x hyper mutation rate x stimulation value) which determines the number of mutated clones a given *ARB* or *MC* is allowed to produce.
- Total Resources: The total number of resources allowed to be shared amongst the *ARB* population. This parameter provides the selection pressure to ensure that the *ARB* pool only contains the most stimulated cells.
- K: The parameter which is used in classification to determine the number of neighbours in the local neighbourhood to consider for the majority vote.
- Stimulation Value: Determined using Euclidian Distance and is the value returned by the stimulation function, given by equation [2].

$$stim(x,y) = 1 - affinity(x,y) \qquad [2]$$

B. Artificial Neural Network

The artificial neural network (ANN) implemented is this study had a topology of two hidden layers with 6 nodes in the first hidden layer and 4 nodes in the second. This topology was arrived at after experimentation and taking into consideration overall run time and accuracy of the models. A sigmoid activation function was used at each node, given by equation [3].

$$\delta(a) = \frac{1}{1 + exp^{-a}} \qquad [3]$$

where a is the sum of each input into the node multiplied by its respective weight.

A. K-Nearest Neighbor

The k-Nearest Neighbours (kNN) algorithm is a simple lazy-learning instance based approach where a local neighborhood of *k*-points are identified based on a similarity function for each tuple in the testing data. Once a local neighborhood is established the testing tuple is classified based on a majority vote. The similarity function is based on the Euclidian distance between the testing tuple in question and each training tuple in the dataset training window, the similarity equation is given in [4]. The kNN algorithm provides a baseline for evaluating the more complex learning methods employed in AIS and ANN. In particular kNN will help assist in assessing the effectiveness of instance-based learning as an alternative to generating a global operator for financial forecasting.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad [4]$$

where d(x, y) is the Euclidian distance between vectors **x** and **y**, and x_i and y_i are the *i*th elements in each feature vector.

V. Data description

The algorithms are training to learn the behaviour of a stock market as a whole and as a result the input dataset reflects information pertaining to macro-economic indicators and past performances of other market indexes. These indicators include, inter alia, measures of inflation, corporate bond ratings and treasury-bill rates. The inputs are not the actual

[1] For a more detailed account of the algorithm please refer to [6].

values of each indicator but the rate of change from one month to the next. The market index which the algorithms are attempting to model is the Dow Jones Industrial Average (DJIA) which is a major market index which is comprised of the 30 largest blue-chip companies in the USA. The classification is binary {0, 1} where a 0 denotes a market contraction and 1 a market expansion. This technique has a shortcoming in that the algorithms are not concerned with magnitude and only direction which can lead to a more accurate model producing inferior investment returns. However classifying based on directional accuracy during training has been shown to produce models which are highly correlated with out-of-sample profitability [19], [20], which is the main objective of these investment models.

A. Data pre-processing

The data for the AIS and kNN algorithms was normalized between 0 and 1, this step is essential for the AIS implementation as the stimulation values and other parameters of the system are setup under the assumption that all data is in this range. The ANN was initially trained with this data format, however the results were very weak and the models were essentially a buy-and-hold approach where each month was predicted to increase, this was the result of a gradient close to 0, caused by the large number of inputs (28) and weights initialized between {0,1}. To combat this short-coming the data was normalized between {-1, 1} to increase the continuous range. The effect does not bias the data as it is still an unambiguous linear transformation.

VI. TRADING STRATEGIES

In addition to comparing the algorithms based on statistical measures, the outputs of the algorithms will be used to generate semi-active short-term trading models, where the models will be making investments locked in for one month at a time. The trading models will be compared to each other and an appropriate bench-mark, for these experiments a buy-and-hold trading strategy will be used. The benchmark portfolio will provide insight into the usefulness of any of the developed models as the extra effort required for the research should produce higher realized returns. Since the predictions are based on the market as a whole and the investments are as well, then the only way to outperform the benchmark is to accurately predict contractions. Under both trading systems the models are predicting the direction the market will move in the coming month.

A. Long positions with a risk-free alternative

The first method will only take long positions in the stock market where in the event of a market contraction being predicted the model will invest in a risk-free rate ($\mathcal{R}_f$). For the purposes of this study the risk-free rate is assumed to be an annualized rate of return of 2% when monthly payments are re-invested and compounded.

B. Long and short positions

The second and more risky strategy will take long and short positions in the market, such a strategy is seeking positive returns from the stock market in times when the market value is decreasing, the process of a taking a short-position is depicted in Fig. 1. A variety of market instruments are available to short the market, such as a put option, but effectively they all profit from market contractions.

Step 1: A short-seller borrows the shares from a lender for a fee and sells the shares on the market at the current price.
Step 2: At sometime in the future the short-seller purchases the shares from the market and returns the shares back to the lender.

Figure1 – An overview of the process of short-selling a security in the stock market or taking a short position.

VII. EXPERIMENT SETUP

The data set contains monthly data which spans a 30 year time period from 1978-2008. The experiments are performed with a sliding window approach, which helps eliminate the negative effects of the non-stationary nature of the time series. Each algorithm is trained for 10 years (120 data tuples) and tested for 1 year (12 data tuples), at which point the window shifts by one year. This approach allows for 20 separate, though not fully independent, training and testing periods, this equates to 240 test points for evaluating the competing performances of the algorithms. As stated, the experiments are conducted with an Artificial Immune System (AIS), k-Nearest Neighbour (kNN) and an Artificial Neural Network (ANN) with 2 hidden layers. With regards to the instance-based learners (kNN and AIS), the experiments are conducted under two different learning assumptions. In the first case the memory of the algorithms is wiped clean at the end of each training/testing window, under the second assumption the memory is allowed to accumulate from one window to the next.

VIII. EXPERIMENT RESULTS

The main focus of this research is the AIS and therefore the results from the AIS experiments will receive more attention. This section will initially report on the testing results of the AIS and subsequently of the other algorithms.

A. Results for AIS with accumulated memory

In table 1 we have the parameter settings for the AIS experiment which yielded the highest performance in terms of overall accuracy for the accumulated memory.

Table 1 – The parameter settings for the AIS experiments using accumulated memory which yielded the highest performance results.

Parameter	**Value**
MC Seed – # of antigens to seed the mc's	1
ARB Seed - # of antigens to seed ARB pool	1
Clonal Rate	10
Mutation Rate	0.15
Stimulation Threshold	0.90
Hyper-mutation Rate	2.0
Affinity Threshold Scalar	0.30
Total Resources	150.00
k – number of neighbors	3

In table 2 the AIS testing results are displayed for the algorithms performance using the above parameter settings. To assist the reader and for space considerations the testing results are grouped into sets of 4, yielding 5 periods of averaged results. For each period the reported results include accuracy, precision for each class, the number of memory cells at the end of training and the number of *mc* replacements. Precision is defined as the number of months correctly predicted to be a certain class c_i divided by the total number of months predicted to be C_i.

Table 2 - Various performance measures for the accumulating memory AIS during the testing periods. Prec() is the precision for class 1 and 0 respectively, *mc* stands for memory cells and *mcr* is memory cell replacements.

Period	**89-92**	**93-96**	**97-00**	**01-04**	**05-08**
Accuracy	0.688	0.708	0.625	0.625	0.667
Prec(1)	0.698	0.847	0.661	0.619	0.698
Prec(0)	0.667	0.492	0.396	0.717	0.583
# of *mc*	122	267.7	390.7	500.7	584.5
# of *mcr*	44.5	3.5	2.0	6.75	15.0

B. *Results for AIS without accumulated memory*

Table 3 displays the parameter settings for the optimal performance over the testing periods when the memory cells are deleted after each window.

Table 3 - The parameter settings for the AIS experiments using non-accumulated memory which yielded the highest performance results.

Parameter	**Value**
MC Seed – # of antigens to seed the mcs	1
ARB Seed - # of antigens to seed ARB pool	1
Clonal Rate	10
Mutation Rate	0.15
Stimulation Threshold	0.93
Hyper-mutation Rate	2.0
Affinity Threshold Scalar	0.30
Total Resources	150.00
k – number of neighbors	7

Table 4 displays the algorithms performance with the above settings in table 3.

Table 4 - Various performance measure for the non-accumulating memory AIS during the testing periods. Prec() is the precision for class 1 and 0, *mc* stands for memory cells and *mcr* is memory cell replacements.

Period	**89-92**	**93-96**	**97-00**	**01-04**	**05-08**
Accuracy	0.604	0.750	0.687	0.604	0.708
Prec(1)	0.645	0.846	0.704	0.595	0.743
Prec(0)	0.555	0.600	0.662	0.542	0.639
# of *mc*	91.75	109.3	110	103	97.75
# of *mcr*	26.25	4.0	2.5	15.0	19.5

From the tables above we see that the differences in terms of parameter settings for the two memory conditions are the stimulation threshold and the *k* number of neighbors to consider. When fewer memory cells are available more of them are required to make a positive prediction. As well, the AIS models benefit from training memory cells to higher threshold when only the recent past is considered. The number of memory cell replacements over a given training window yields some interesting insights into how the AIS models are learning, in figure 2 we have a plot of the DJIA market index over the testing window along with the number of memory cell replacements that each model had for each year. In the early `90s when the market was fairly stable and trending, the number of *mc* replacements was quite high for both models. This is because the antigens entering the AIS system are very similar and therefore the evolved antibody is also quite similar to other existing *mcs*, this makes the possibility of replacement more likely. A complement to this observation is that when the market was more volatile the number of *mc* replacements declines for both models, displaying that the evolved antibodies are unlike any which have been seen before as the current market conditions are unique at that time as well. The two largest differences between the yearly accuracies occur in 1991 and 2002, with the AIS system allowed to retain its memory having the superior accuracy in 1991 and the reverse for 2002. In terms of precision when predicting positive market movements both models produced results below 50% only once over the 20 years and in both cases this was achieved when the models had their largest differences in overall accuracy (1991 and 2002). It is interesting to note that the extended memory cell AIS performs better when the market is less volatile and the movement is relatively flat, where as the shortened memory cell AIS is performing better when the market trend experiences a drastic change. This behavior is based on each models interpretation of memory and how quickly the models are able to adapt to market changes. The extended memory cell model exhibits a behavioral trait of conservatism, which is the slow updating of models in the face of new evidence, so in quick trend changes the quality of the predictions is eroded. This observation could also indicate that the market behavior was non-stationary and therefore the extended memory compromises the prediction quality. This behavior has been linked to stock returns and market inefficiencies in [21]. The shortened memory cell models have a somewhat myopic view of market behavior and can be more susceptible to small fluctuations, which is usually most predatory to investor earnings in times of sideways moving markets, as was experienced in 1991.

A. *Comparison results*

In table 5 the results from all of the algorithms are displayed. Reported for these models are the overall accuracy, precision for both classes and their respective standard deviations. For the kNN and AIS models the results are shown for both accumulating and non-accumulating memory.

Table 5 - Testing results for the AIS, ANN and kNN algorithms. Prec() is the precision for class 1 and 0 and sd is the standard deviation. AIS/kNN-1 and AIS/kNN -2 are the accumulating and non-accumating memroy models respectively.

	Accuracy	**sd**	**Prec(1)**	**sd**	**Prec(0)**	**sd**
ANN	0.692	0.112	0.748	0.146	0.691	0.286
kNN-1	0.458	0.168	0.551	0.125	0.286	0.065
kNN-2	0.542	0.168	0.594	0.136	0.333	0.118
AIS -1	0.663	0.12	0.715	0.149	0.571	0.294
AIS-2	0.671	0.134	0.706	0.168	0.600	0.269

From the table above we see that the ANN was able to produce slightly higher overall accuracy for the testing sets with an accuracy of 69.17% compared to the AIS models of 67.08% and 66.25% for the non-accumulating and accumulating models respectively.

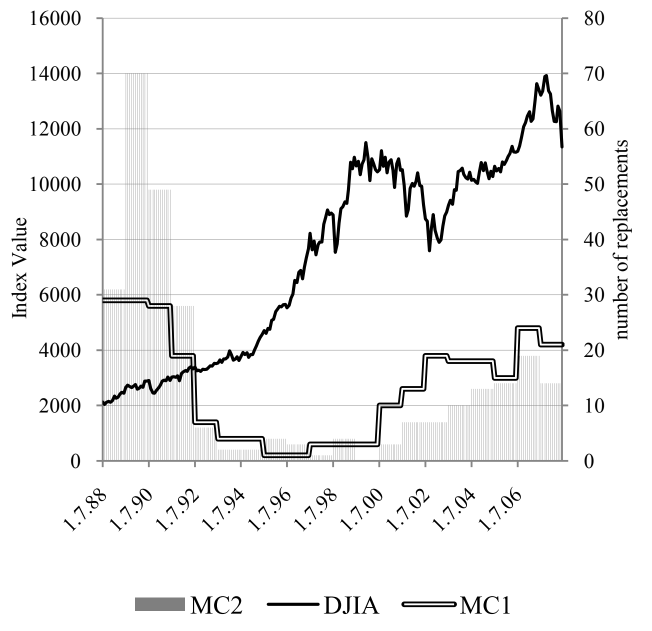

Figure 2 - A plot of the DJIA index value from 1988 - 2008 along with the number of memory cell replacements, MC1 and MC2, for the accumulating and non-accumulating AIS models respectively.

The kNN models were both significantly inferior across each measure and the additional data points gained from an accumulating memory lead to a reduction in model robustness. For both AIS models and the ANN the predictions were more reliable for market expansions where we have a higher overall precision and lower standard deviation. For the instance-based learning approaches there appears to be an inverse relationship between the value of *k* and the number of data points available in the global neighborhood, where the larger the training set or the number of *mcs* the smaller the value of k^2. Figure 3 provides a plot of the yearly accuracies for the ANN and the AIS models (kNN is not represented as the results were significantly inferior to the other approaches). From figure 3 we see that the AIS models are outperforming the ANN in the latter part of the testing period (2005 -2008).

IX. TRADING RESULTS

A. Long-positions with risk-free alternative

The trading simulations generated from each of the models under the strategy which only takes long positions in the stock market (as discussed in section 6.A) are displayed in table 6. The results with regards to annual returns and cumulative profits do not account for transaction costs, this is for simplicity reasons and that this research is mainly intended for institutional investors. Reported are cumulative return (assuming an initial investment of \$1000.00 and returns are 100% re-invested), average yearly return and the Sharpe Ratio [22] which is a commonly used metric for investment managers to gauge how efficient a trading strategy is with the extra risk it is exposed to, where the higher the value the better. The Sharpe Ratio, $\mathcal{S}$, is shown in equation [5]: the numerator is the risk adjusted expected return and the denominator is the amount of variance in those returns for the reporting period; the more risky an investment the greater the degree of variance in its value.

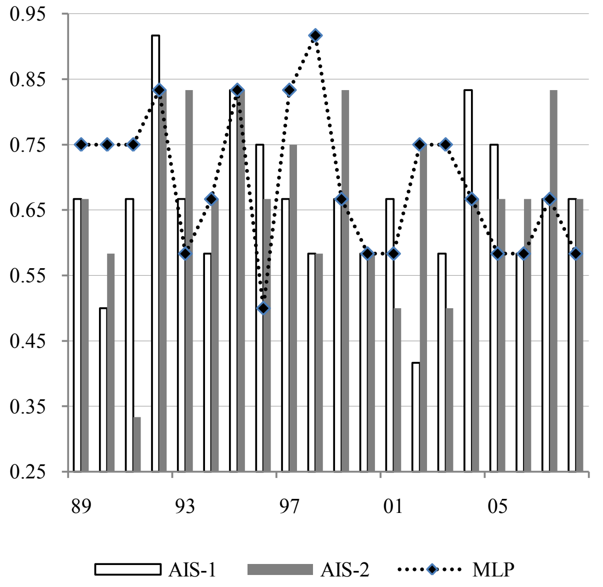

Figure 3 - Accuracy results for the AIS and ANN models during the testing period.

$$\mathcal{S} = \frac{\mathrm{E}[\mathcal{R} - \mathcal{R}_f]}{\sqrt{var[\mathcal{R} - \mathcal{R}_f]}} \qquad [5]$$

where $\mathcal{R}$ is the return on the asset and $\mathcal{R}_f$ is a risk-free rate.

Table 6 - Results from trading simulations without short positions allowed for each model over the testing period. The benchmark is the DJIA performance from 1988-2008.

Model	Cumulative Return (\$)	Average Yr. Return	Average Sharpe Ratio
ANN	21478.82	0.159	4.792
kNN-1	1652.58	0.027	<0
kNN-2	3379.53	0.061	<0
AIS-1	11241.11	0.117	3.634
AIS-2	13026.22	0.126	3.971
Benchmark	5299.51	0.082	2.042

We can see that a slightly higher average yearly return for the ANN leads to significantly more valuable trading profits, which is why the cumulative return is not quoted on its own as one abnormal gain can lead to drastic differences in cumulative profit. The Sharpe ratio for the ANN is also the highest which demonstrates that it was also the most

[2] The value of *K* for the kNN approaches were chosen empirically and were 1 and 7 for the accumulating and non-accumulating models respectively.

efficient with the additional risk its model was exposed to. The AIS models were comparable to the ANN, with the non-accumulating memory model producing the superior result. When compared to the benchmark the ANN and AIS models were able to outperform it with regards to cumulative return and their Sharpe ratios.

B. *Long and short positions*

In table 7 we have the trading results for models which take long and short positions in the market.

Table 7 - Results from trading simulations with short positions allowed for each model over the testing period. The benchmark is the DJIA performance from 1988-2008.

Model	Cumulative Return ($)	Average Yr. Return	Average Sharpe Ratio
ANN	56986.32	0.226	5.398
kNN-1	331.43	-0.035	-1.171
kNN-2	1754.58	0.034	0.337
AIS-1	15930.46	0.136	3.386
AIS-2	21438.52	0.154	4.306
Benchmark	5299.51	0.082	2.042

Once again the top performing model is that of the ANN; however there is a significant difference between the AIS models and the kNN approach, which is consistent with previous results. As well the AIS models and the ANN were able to outperform the benchmark in terms of cumulative return and Sharpe ratios. From the results we can infer that under instance-based learning approaches the models have not benefited from an accumulating memory with the kNN models performance negatively affected the most. Using a trading model with shorting allowed emphasizes the kNN models inability to predict market contractions (as seen in table 5) where the low precision on class 0 has lead to above average losses in those periods. The AIS models and the ANN were better at modeling the market behavior before contractions which yielded positive investment gains in 2008 at a time when the market was experiencing a large trend shift. Figure 4 shows a plot of the monthly returns for each model and the market for the first 6 months of 2008.

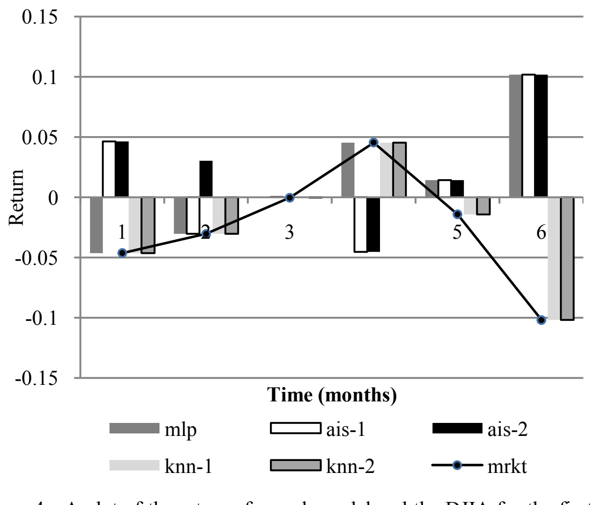

Figure 4 - A plot of the returns for each model and the DJIA for the first 6 months of 2008 when the market experienced a drastic trend change.

During this time period the market produces negative returns for 5 of the 6 months, conversely the AIS models with and without accumulating memory only had two and one negative month during the same time period respectively.

X. Results Analysis

From the results in section IX we have seen that small differences in accuracy can lead to significant differences in trading profits. However the trading profits may not generalize as the classification of market movements only considers direction and not magnitude, as a result two algorithms with equal accuracy could produce different trading results. To aid in determining the effectiveness of AIS for market prediction the results in-terms of accuracy will be considered as a statistical test can be performed to discover if the output of the models are significantly different. The test used in this study is a one-sided t-test for binary output distribution. Table 8 displays a matrix of p-values generated from the previously described statistical test.

Table 8 - p-values generated from a one-sided t-test for binary output distribution.

	ANN	AIS – 1	AIS – 2	kNN – 1	kNN – 2
ANN		0.167	0.244	< 0.001	< 0.001
AIS – 1	0.167		0.392	< 0.001	< 0.001
AIS – 2	0.244	0.392		< 0.001	< 0.001
kNN – 1	<0.001	< 0.001	< 0.001		< 0.001
kNN – 2	< 0.001	< 0.001	< 0.001	< 0.001	

Form the p-values reported above we can determine that the differences in accuracy between the ANN and AIS models was not statistically significant with p-values of 0.167 and 0.244 for the accumulating and non-accumulating models respectively. When comparing the instance-based learning approaches we see that the differences are statistically significant using a 99% confidence interval with p-values of less than .01 for each comparison.

Although the accuracies for the AIS and ANN are statistically significant when compared to the simpler kNN approach these results do not indicate how predictable the market was over this time period. A classifier[3] which always predicted the majority class (class 1) was constructed; its accuracy was 59.17%. The accuracies of the ANN and AIS models are statistically significant[4] in comparison to this majority-class classification, providing evidence that the algorithms have been able to learn a portion of the market behavior.

XI. Conclusion

The main focus of this work was to establish if an AIS is a suitable supervised learning technique to model the stock market. This analysis was considered from two angles, first with regards to kNN a much simpler instance-based learning

[3] The out-of-sample data had an approximate distribution of 60/40 for class 1 and 0 respectively.

[4] P-values were 0.000429, 0.010363 and 0.00467 for the ANN, and the accumulating and non-accumulating AIS models respectively.

algorithm to determine if the increased complexity of an AIS offered any advantages and secondly a comparison to other commonly held benchmarks, a ANN and a stock market portfolio, to obtain a more global view of its effectiveness. By all measures of performance introduced in this study the AIS was able to outperform the kNN algorithm. The added complexity of evolving a set of memory cells to model the search space rather than the actual previous instances produced superior results which were statistically significant and which generated substantially more profitable trading models. The ANN did outperform the AIS models whether they used accumulating or non-accumulating memory although the results were not statistically significant with regards to accuracy and the superior trading results cannot be guaranteed to generalize because of a short-coming of the classification technique. The data used in this study is not necessarily the most suitable for AIS algorithms and other datasets could be more favorable to instance-based approaches, future work will consider this question. The AIS models did outperform the other benchmark the market portfolio in terms of cumulative investment return and the Sharpe ratio. Although the returns did not include transaction costs, these would be minimal as the models only make trades on a monthly basis and only if required[5]. This trading simulation along with the comparison to the majority-class classifier provides evidence that the DJIA index monthly returns contained a degree of predictability from 1989-2008, which reflects work done by [23] where the authors had a similar conclusion from training with reinforcement learning on the S&P 500 (another US index) from 1970 - 1994. Given these results the AIS could be considered a viable option for modeling the stock market using an instance-based approach rather than developing a global operator such as with an artificial neural network or a support vector machine. In general the practice of using the natural immune system to inspire a learning algorithm has been established as a viable alternative to modeling the stock market when implementing a supervised learning approach.

REFERENCES

[1] Iba, H. and Sasaki, T. (1999). Using Genetic programming to Predict Financial Data. Evolutionary Computation, 1999. CE 99. Volume 1, page 251.

[2] Li, J. and E.P.K. Tsang (1990a). Improving Technical Analysis Prediction: An Application of Genetic Programing. Proceedings of The 12th International Florida AI Research Society Conference. Pages108-112.

[3] M. Butler and A. Daniyal (2009). Multi-objective Optimization with an Evolutionary Artificial Neural Network for Financial Forecasting. Proceeding of 11th conference on Genetic and evolutionary computation. Pages 1451-1458.

[4] Holland, John H (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

[5] Koza, John (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press. ISBN 0-262-11170-5

[6] Watkins, A., Timmis, J., Boggess, L.: Artificial immune recognition system (AIRS): An immune inspired supervised machine learning algorithm. Genetic Programming and Evolvable Machines 5 (2004) 291–317

[7] Leandro N. de Castro and Fernando J. Von Zuben. Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems. 2002; 6(3): 239-251.

[8] D. Dasgupta and S. Forrest. Novelty-detection in time series data using ideas from immunology, Proceedings of the International Conference on Intelligent Systems, Reno, Nevada (1996).

[9] A. Secker, A. Freitas and J. Timmis. AISEC: An Artificial Immune System for Email Classification. Proceedings of the Congress on Evolutionary Computation (CEC-2003), pp. 131-139. IEEE, 2003.

[10] Eugene Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. Journal of Finance, XXV,No.2 (March 1970), pages 226-241.

[11] Weber, M., and C. Camerer, 2000, The disposition effect in securities trading: An experimental analysis, Journal of Economic Behavior and Organization 33, 167-184.

[12] Frazzini, A. 2006. The Disposition Effect and Underreaction to News. Journal of Finance 61(4):2017–46.

[13] Maggini, M. Giles, C.L., and Horne, B (1997). Financial Time Series Forecasting Using K-Nearest Neighbours Classification. Proceedings of Nonlinear Financial Forecasting. Pages 169-181.

[14] S. Zemke, Nonlinear Index Prediction, in Proceedings of the International Workshop on Econophysics and Statistical Finance Palermo, Italy, September 1998, Physica A Vol. 269, no. 1, Elsevier Science, R. N. Mantegna, Ed., pp. 177–183.

[15] J. C. Hsieh, S. H. Chen, and P. C. Chang, Application of Artificial Immune System in Constructing a Financial Early Warning System: An Example of Taiwanese Banking Industry, in Innovative Computing, Information and Control,, 2006, pp. 183-187.

[16] J. Timmis and M. Neal, A resource limited artificial immune system for data analysis, Knowledge-Based Systems, 14(3-4), pp.121-130, 2001.

[17] Singh, R and Sengupta, R.N, Bankruptcy Protection Using Artificial Immuns Systems. Proceeding of the 6th International Conference on Artificial Immune Systems ICARIS 2007 LNCS 4628, pp 131-141.

[18] Wu Ze-jun, Chen Jia, Yang Huan, Lv Lin, Wang Xin-an, An Artificial Immune Model for Abnormal Fluctuation of Stock Price, ISCID, vol. 1, pp.274-277, 2008 International Symposium on Computational Intelligence and Design.

[19] G. Leitch and E. Tanner, Economic Forecast Evaluation: Profit Versus the Conventional Error Measures, American Economic Review, vol. 81, pp. 580-591, 1991.

[20] S. Hayward, Setting up Performance Surface of an Artifical Neural Network with Genetic Algorithm Optimization: in Search of an Accurate and Profitable Prediciton for Stock Trading, Congress on Evolutionary Computation, CEC 2004. Volume 1, pages 948- 954, 2004.

[21] Barberis, N., Shleifer, A., Vishny, R., 1998. A model of investor sentiment. Journal of Financial Economics 49, 307–343.

[22] W. F. Sharpe, Mutual Fund Performance, Journal of Business pp. 119-138, 1966.

[23] J. Moody, L.Wu,Y. Liao, and M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, *J. Forecasting*, vol. 17, pp. 441–470, 1998.

[5] If any model predicted the market to continue in the same direction for the succeeding month then no action is required and therefore no transaction costs are incurred.

Glossary

alSAX	adaptive local SAX
AMH	adaptive market hypothesis
AIS	artificial immune system
AI	artificial intelligence
ANN	artificial neural network
ADF	augmented Dickey Fuller test
ACF	autocorrelation function
AR	Autoregressive
BF	behavioral finance
CAPM	capital asset pricing model
CI	computational intelligence
DKF	diffuse Kalman filter
dHPSO	dynamic heterogeneous particle swarm optimization
EMH	efficient market hypothesis
EC	evolutionary computation
EM	expectation-maximization algorithm
GA	genetic algorithm
GP	genetic programming
HMM	hidden Markov model
J48	J48 Decision tree
kNN	k-nearest neighbors
LATIS	learning adaptive technical indicator system
ML	machine learning
MLP	multilayer perceptron
MOO	multi-objective optimization
PSO	particle swarm optimization
PP	Phillips-Perron test
PAA	piecewise aggregate approximation
RL	reinforcement learning
SL	supervised learning
SVM	support vector machine
SAX	symbolic aggregate approximation
VE	variable efficiency
VS	variable stationarity
ABB	adaptive Bollinger bands

Bibliography

[1] I.P. Androulakis, Wu J, Vitolo J, and C. Roth. Selecting maximally informative genes to enable temporal expression profiling analysis. In *Proceedings of Foundations of Systems Biology in Engineering*, 2005.

[2] Alberto Apostolico, Mary Ellen Bock, and Stefano Lonardi. Monotony of surprise and large-scale quest for unusual words. In *Proceedings of the sixth annual international conference on Computational biology*, RECOMB '02, pages 22–31, New York, NY, USA, 2002. ACM.

[3] W.B. Arthur. *On learning and adaptation in the economy*. Institute for Economic Research, Queen's University, 1992.

[4] Nicholas Barberis, Andrei Shleifer, and Robert Vishny. A model of investor sentiment. *Journal of Financial Economics*, 49(3):307 – 343, 1998.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] Bruce Vanstone Bjoern Krollner and Gavin Finnie. Financial time series forecasting with machine learning techniques: A survey. In *European symposium on artificial neural networks: Computational and machine learning. Bruges, Belgium*, 2010.

[7] Fischer Black. Studies of stock price volatility changes. In *Proceedings of the 1976 Meetings of the American Statistical Association, Business and Economics Statistics Section*, pages 177–181, 1976.

[8] M.E. Blume. Betas and their regression tendencies. *The Journal of Finance*, 30(3):785–795, 2012.

[9] T. Bollerslev, R.F. Engle, and J.M. Wooldridge. A capital asset pricing model with time-varying covariances. *The Journal of Political Economy*, pages 116–131, 1988.

[10] R.D. Brooks, R.W. Faff, and M.D. McKenzie. Time-varying beta risk of australian industry portfolios: A comparison of modelling techniques. *Australian Journal of Management*, 23(1):1–22, 1998.

[11] P. Buhlmann. Extreme events from the return-volume process: a discretization approach for complexity reduction. *Applied financial economics*, 8(3):267–278, 1998.

[12] Matthew Butler and Dimitar Kazakov. Modeling the behavior of the stock market with an artificial immune system. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[13] Matthew Butler and Dimitar Kazakov. Particle swarm optimization of bollinger bands. In *ANTS Conference*, pages 504–511, 2010.

[14] Matthew Butler and Dimitar Kazakov. A learning adaptive bollinger band system. In *IEEE Computational Intelligence for Financial Engineering and Economics*, pages 1–8, 2012.

[15] Daniel O. Cajueiro and Benjamin M. Tabak. Evidence of long range dependence in Asian equity markets: the role of liquidity and market restrictions. *Physica A: Statistical Mechanics and its Applications*, 342(3-4):656 – 664, 2004.

[16] Daniel O. Cajueiro and Benjamin M. Tabak. Ranking efficiency for emerging markets. *Chaos, Solitons and Fractals*, 22(2):349 – 352, 2004.

[17] D.O. Cajueiro and B.M. Tabak. Ranking efficiency for emerging equity markets ii. *Chaos, Solitons & Fractals*, 23(2):671–675, 2005.

[18] Rachel Caspari. The evolution of grandparents. *Scientific American*, 305:44–49, August 2011.

[19] S.W.K. Chan and J. Franklin. A text-based decision support system for financial sequence prediction. *Decision Support Systems*, 52(1):189–198, 2011.

[20] A. Chauhan. Automated stock trading and portfolio optimization using xcs trader and technical analysis. Master's thesis, University of Edinburgh, 2008. MSc Thesis.

[21] A.P. Chen and Y.H. Chang. Using extended classifier system to forecast S&P futures based on contrary sentiment indicators. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2084–2090. IEEE, 2005.

[22] S.H. Chen. *Genetic algorithms and genetic programming in computational finance*. Springer, 2002.

[23] S.H. Chen, M. Kampouridis, and E. Tsang. Microstructure dynamics and agent-based financial markets. *Multi-Agent-Based Simulation XI*, pages 121–135, 2011.

[24] S.H. Chen, N. Navet, et al. Failure of genetic-programming induced trading strategies: Distinguishing between efficient markets and inefficient algorithms. *Computational Intelligence in Economics and Finance: Volume II*, 2:169–182, 2007.

[25] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM.

[26] Taufiq Choudhry and Hao Wu. Forecasting the weekly time-varying beta of UK firms: Garch models vs. kalman filter method. *The European Journal of Finance*, 15(4):437–444, 2009.

[27] Joshua D. Coval and Tyler Shumway. Do behavioral biases affect prices? *Journal of Finance*, 60(1):1–34, 2005.

[28] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[29] Francis X. Diebold and Lutz Kilian. Unit-root tests are useful for selecting forecasting models. *Journal of Business and Economic Statistics*, 18(3):265–273, 2000.

[30] I. Domowitz and C.S. Hakkio. Conditional variance and the risk premium in the foreign exchange market. *Journal of International Economics*, 19(1):47–66, 1985.

[31] C. Dose and S. Cincotti. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications*, 355(1):145–151, 2005.

[32] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[33] Markus Ebner and Thorsten Neumann. Time-varying betas of German stock returns. *Financial Markets and Portfolio Management*, 19(1):29–46, 2005.

[34] Andries Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 191–202. Springer Berlin / Heidelberg, 2010.

[35] Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2):383–417, May 1970.

[36] Eugene F. Fama, Lawrence Fisher, Michael C. Jensen, and Richard Roll. The adjustment of stock prices to new information. *International Economic Review*, 10(1):1–21, 1969.

[37] Eugene F. Fama and Kenneth R. French. Multifactor explanations of asset pricing anomalies. *The Journal of Finance*, 51(1):55–84, 1996.

[38] Fernando Fernandez Rodriguez, Christian Gonzalez Martel, and Simon J. Sosvilla Rivero. Optimisation of Technical Rules by Genetic Algorithms: Evidence from the Madrid Stock Market. *SSRN eLibrary*, 2001.

[39] Gonzlez-Martel Christian Fernndez-Rodrguez, Fernando and Simn Sosvilla-Rivero. Optimization of technical rules by genetic algorithms: evidence from the Madrid stock market. *Applied Financial Economics*, 15(11):773–775, 2005.

[40] M. Field, D. Stirling, F. Naghdy, and Z Pan. Motion segmentation for humanoid control planning. In *Proceedings of ARAA Australasian Conference on Robotics and Automation*, 2008.

[41] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[42] TC Fu, FL Chung, R. Luk, and CM Ng. Financial time series indexing based on low resolution clustering. In *4th IEEE Intl Conference on Data Mining (ICDM 2004) Workshop on Temporal Data Mining: Algorithms, Theory and Applications*, pages 5–14, 2004.

[43] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183 – 192, 1989.

[44] Massimo Gastaldi and Annamaria Nardecchia. The kalman filter approach for time-varying beta estimation. *Syst. Anal. Model. Simul.*, 43(8):1033–1042, August 2003.

[45] S. Grossman. On the efficiency of competitive stock markets where trades have diverse information. *The Journal of Finance*, 31(2):573–585, 2012.

[46] S.J. Grossman and J.E. Stiglitz. On the impossibility of informationally efficient markets. *The American Economic Review*, pages 393–408, 1980.

[47] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[48] R.A. Haugen and N.L. Baker. Case closed. *Handbook of Portfolio Construction*, pages 601–619, 2010.

[49] Melvin J. Hinich. Testing for dependence in the input to a linear time series model. *Journal of Nonparametric Statistics*, 6(2-3):205–221, 1996.

[50] M.J. Hinich and D.M. Patterson. Detecting epochs of transient dependence in white noise. Mimeo, University of Texas at Austin, 1995.

[51] J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[52] J. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In *Machine learning: An artificial intelligence approach*, volume 2, pages 593–623. Morgan Kaufmann, 1986.

[53] Xiaohui Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1677 –1681, 2002.

[54] Nguyen Quoc Viet Hung and Duong Tuan Anh. Combining SAX and piecewise linear approximation to improve similarity search on financial time series. In *Proceedings of the 2007 International Symposium on Information Technology Convergence*, ISITC '07, pages 58–62, Washington, DC, USA, 2007. IEEE Computer Society.

[55] Harold Hurst. Long Term Storage Capacity of Reservoirs. *Transactions of the American Society of Civil Engineers*, 116:770–799, 1951.

[56] H. Iba and T. Sasaki. Using genetic programming to predict financial data. In *Evolutionary Computation, Proceedings of the 1999 Congress on*, volume 1. IEEE, 1999.

[57] M. Ito and S. Sugiyama. Measuring the degree of time varying market inefficiency. *Economics Letters*, 103(1):62–64, 2009.

[58] C.M. Jarque and A.K. Bera. A test for normality of observations and regression residuals. *International Statistical Review/Revue Internationale de Statistique*, pages 163–172, 1987.

[59] Christopher Jeffery. Synthetic lightning emp data, 2005. http://nis-www.lanl.gov/ eads/datasets/emp.

[60] P.D. Jong. The diffuse Kalman filter. *The Annals of Statistics*, 19(2):1073–1083, 1991.

[61] M. A. Kaboudan. A measure of time series' predictability using genetic programming applied to stock returns. *Journal of Forecasting*, 18(5):345–357, 1999.

[62] MA Kaboudan. Genetic programming prediction of stock prices. *Computational Economics*, 16(3):207–236, 2000.

[63] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, pages 263–291, 1979.

[64] R.E. Kalman and R.S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(3):95–108, 1961.

[65] Michael Kampouridis, Shu-Heng Chen, and Edward Tsang. Market fraction hypothesis: A proposed test. *International Review of Financial Analysis*, 23(0):41 – 54, 2012.

[66] Michael Kampouridis, Shu-Heng Chen, and Edward P. K. Tsang. Testing the dinosaur hypothesis under empirical datasets. In *PPSN (2)*, pages 199–208, 2010.

[67] Michael Kampouridis, Shu-Heng Chen, and Edward P. K. Tsang. Market microstructure: Can dinosaurs return? A self-organizing map approach under an evolutionary framework. In *EvoApplications (2)*, pages 91–100, 2011.

[68] Michael Kampouridis and Edward P. K. Tsang. EDDIE for investment opportunities forecasting: Extending the search space of the gp. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[69] M. G. Kendall and A. Bradford Hill. The analysis of economic time-series-part i: Prices. *Journal of the Royal Statistical Society. Series A (General)*, 116(1):11–34, 1953.

[70] James Kennedy and Russell Eberhart. Particle swarm optimization. In *International Conference on Neural Networks, IEEE Service Center*, pages 1942–1948, 1995.

[71] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[72] Tae Yoon Kim, Kyong Joo Oh, Chiho Kim, and Jong Doo Do. Artificial neural networks for non-stationary time series. *Neurocomputing*, 61:439 – 447, 2004. Hybrid Neurocomputing: Selected Papers from the 2nd International Conference on Hybrid Intelligent Systems.

[73] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science, 1990.

[74] J.R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[75] Denis Kwiatkowski, Peter C. B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root : How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1-3):159 – 178, 1992.

[76] Ju-Sang Lee, Sangook Lee, Seokcheol Chang, and Byung-Ha Ahn. A comparison of GA and PSO for excess return evaluation in stock markets. In *IWINAC (2)*, pages 221–230, 2005.

[77] C. Lento. Combined signal approach: evidence from the Asian–Pacific equity markets. *Applied Economics Letters*, 16(7):749–753, 2009.

[78] C. Lento and N. Gradojevic. The profitability of technical trading rules: a combined signal approach. *Journal of Applied Business Research*, 23(1):13–27, 2007.

[79] C. Lento, N. Gradojevic, and C.S. Wright. Investment information content in Bollinger bands? *Applied Financial Economics Letters*, 3(4):263–267, 2007.

[80] J. Leung and T. Chong. An empirical comparison of moving average envelopes and Bollinger bands. *Applied Economics Letters*, 10(6):339–341, 2003.

[81] M.T. Leung, H. Daouk, and A.S. Chen. Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting*, 16(2):173–190, 2000.

[82] Robert A. Levy. Conceptual foundations of technical analysis. *Financial Analysts Journal*, 22(4):83–89, 1966.

[83] Jin Li and Edward P. K. Tsang. Improving technical analysis predictions: An application of genetic programming. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pages 108–112. AAAI Press, 1999.

[84] P.Y. Liao and J.S. Chen. Dynamic trading strategy learning model using learning classifier systems. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 783–789. IEEE, 2001.

[85] Kian-Ping Lim. Ranking market efficiency for stock markets: A nonlinear perspective. *Physica A: Statistical Mechanics and its Applications*, 376:445 – 454, 2007.

[86] Cao L.-Wang J. Zhang C Lin, L. The applications of genetic algorithms in stock market data mining optimisation. In N. F. F. Ebecken A. Zanasi and C. A. Brebbia, editors, *Data Mining V*. WIT Press, 2004.

[87] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. ACM Press, 2003.

[88] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, October 2007.

[89] J. Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The review of economics and statistics*, 47(1):13–37, 1965.

[90] S. Liu and T. Nagao. HXCS and its application to financial time series forecasting. *IEEJ Transactions on Electrical and Electronic Engineering*, 1(4):417–425, 2006.

[91] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation. In *In IEICE 17th Data Engineering Workshop, March 1-3 2006*, 2006.

[92] Andrew W. Lo. Long-term memory in stock market prices. *Econometrica*, 59(5):pp. 1279–1313, 1991.

[93] Andrew W. Lo. The adaptive markets hypothesis: market efficiency from an evolutionary perspective. *Journal of Portfolio Management*, 30:15–29, 2004.

[94] Andrew W. Lo. Reconciling efficient markets with behavioral finance: the adaptive markets hypothesis. *Journal of Investment Consulting*, 7:21–44, 2005.

[95] Andrew W. Lo and A. Craig MacKinlay. Stock market prices do not follow random walks: Evidence from a simple specification test. *The Review of Financial Studies*, 1(1):41–66, 1988.

[96] Stefano Lonardi. *Global detectors of unusual words: design, implementation, and applications to pattern discovery in biosequences*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2001. AAI3043754.

[97] Ritanjali Majhi, G. Panda, Babita Majhi, and G. Sahoo. Efficient prediction of stock market indices using adaptive bacterial foraging optimization (abfo) and bfo based techniques. *Expert Syst. Appl.*, 36(6):10097–10104, 2009.

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

[99] Sonia Mazzi and Matthew Butler. Dynamic risk analysis with incomplete information in an adaptive market. Currently under review at the Journal of Business and Economic Statistics, submitted September 10, 2012.

[100] A. McGovern, D. Rosendahl, A. Kruger, M. Beaton, R. Brown, and K. Droegemeier. Understanding the formation of tornadoes through data mining. In *Proceedings of 5th Conference on Artificial Intelligence and its Applications to Environmental Sciences at the American Meteorological Society*, 2007.

[101] Michael D. McKenzie, Robert D. Brooks, and Robert W. Faff. The use of domestic and world market indexes in the estimation of time-varying betas. *Journal of Multinational Financial Management*, 10(1):91 – 106, 2000.

[102] J. Moody, L. Wu, Y. Liao, and M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Applied Financial Economics Letters*, 17:441–470, 1998.

[103] Ian Morgan, Honghai Liu, George Turnbull, and David Brown. Time discretisation applied to anomaly detection in a marine engine. In *Proceedings of the 11th international conference, KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems: Part I*, KES'07/WIRN'07, pages 405–412, Berlin, Heidelberg, 2007. Springer-Verlag.

[104] A.T.Y. Musetti. Clustering methods for financial time series. Master's thesis, Swiss Federal Institute of Technology Zurich, 2012. MSc Thesis.

[105] Christopher Neely, Paul Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *The Journal of Financial and Quantitative Analysis*, 32(4):405–426, 1997.

[106] Charles R. Nelson and Charles R. Plosser. Trends and random walks in macroeconomic time series: Some evidence and implications. *Journal of Monetary Economics*, 10(2):139 – 162, 1982.

[107] Serena Ng and Pierre Perron. Lag length selection and the construction of unit root tests with good size and power. *Econometrica*, 69(6):1519–1554, 2001.

[108] Jangmin O, Jongwoo Lee, Jae Won Lee, and Byoung-Tak Zhang. Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Inf. Sci.*, 176(15):2121–2147, 2006.

[109] Stephanos Papadamou and George Stephanides. Improving technical trading systems by using a new MATLAB based genetic algorithm procedure. In *NOLASC'05: Proceedings of the 4th WSEAS International Conference on Non-linear Analysis, Non-linear Systems and Chaos*, pages 129–134, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS).

[110] Peter C. B. Phillips and Pierre Perron. Testing for a unit root in time series regression. *Biometrika*, 75(2):335–346, 1988.

[111] L. Pichl, T. Yamano, and T. Kaizoji. On the symbolic analysis of market indicators with the dynamic programming approach. *Advances in Neural Networks-ISNN 2006*, pages 432–441, 2006.

[112] J.Y. Potvin, P. Soriano, and M. Vallée. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004.

[113] R. Preen. An XCS approach to forecasting financial time series. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2625–2632. ACM, 2009.

[114] R. Preen. Identifying trade entry and exit timing using mathematical technical indicators in XCS. *Learning Classifier Systems*, pages 166–184, 2010.

[115] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 10.1007/BF00116251.

[116] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.

[117] M.G. Reyes. Size, time-varying beta, and conditional heteroscedasticity in UK stock returns. *Review of Financial Economics*, 8(1):1–10, 1999.

[118] C. Schittenkopf, P. Tiňo, and G. Dorffner. The benefit of information reduction for trading strategies. *Applied Economics*, 34(7):917–930, 2002.

[119] S. Schulenburg and P. Ross. Explorations in LCS models of stock trading. *Advances in Learning Classifier Systems*, pages 309–331, 2002.

[120] G.W. Schwert and P.J. Seguin. Heteroskedasticity in stock returns. *the Journal of Finance*, 45(4):1129–1155, 2012.

[121] G.C. Selden. *The Psychology of the Stock Market*. Ticker Publishing Company, 1912.

[122] W.F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk*. *The Journal of finance*, 19(3):425–442, 2012.

[123] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69 –73, 4-9 1998.

[124] Jin Shieh and Eamonn Keogh. isax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, pages 623–631, New York, NY, USA, 2008.

[125] A. Smith and W. Letwin. *The wealth of nations*. Dent, 1957.

[126] Christopher Stone and Larry Bull. Foreign exchange trading using a learning classifier system. In *Learning Classifier Systems in Data Mining*, volume 125 of *Studies in Computational Intelligence*, pages 169–189. Springer Berlin Heidelberg, 2008.

[127] Harish Subramanian, Subramanian Ramamoorthy, Peter Stone, and Benjamin Kuipers. Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO*, pages 1777–1784, 2006.

[128] Yoshiki Tanaka and Kuniaki Uehara. Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In *Machine Learning and Data Mining in Pattern Recognition*, volume 2734 of *Lecture Notes in Computer Science*, pages 297–322. Springer Berlin / Heidelberg, 2003.

[129] Alexandru Todea. Episodic dependencies and the profitability of moving average strategy on Romanian capital market. *Theoretical and Applied Economics*, 11((528)):186–193, November 2008.

[130] Alexandru Todea, Maria Ulici, and Simona Silaghi. Adaptive markets hypothesis - evidence from Asia-Pacific financial markets. *The Review of Finance and Banking*, 1(1):007–013, December 2009.

[131] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[132] W.C. Tsai and A.P. Chen. Using the XCS classifier system for portfolio allocation of MSCI index component stocks. *Expert Systems with Applications*, 38(1):151–154, 2011.

[133] Andrew Watkins, Jon Timmis, and Lois Boggess. Artificial immune recognition system (AIRS): An immune-inspired supervised learning algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, 2004.

[134] Martin Weber and Colin F. Camerer. The disposition effect in securities trading: an experimental analysis. *Journal of Economic Behavior & Organization*, 33(2):167–184, January 1998.

[135] G. Wilson and W. Banzhaf. Fast and effective predictability filters for stock price series using linear genetic programming. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.

[136] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 2(3):149–175, 1995.

[137] S.Y.B. Wong and S. Schulenburg. Portfolio allocation using xcs experts in technical analysis, market conditions and options market. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2965–2972. ACM, 2007.

[138] Holbrook Working. A random-difference series for use in the analysis of time series. *Journal of the American Statistical Association*, 29(185):11–24, 1934.

[139] P.-S. Wu and J.-S. Chiou. Multivariate test of Sharpe-Lintner CAPM with time-varying beta. *Applied Financial Economics Letters*, 3(5):335–341, 2007.

[140] T. Yamano, K. Sato, T. Kaizoji, J.M. Rost, and L. Pichl. Symbolic analysis of indicator time series by quantitative sequence alignment. *Computational Statistics & Data Analysis*, 53(2):486–495, 2008.

[141] J. Yao and J. Gao. Computer-intensive time-varying model approach to the systematic risk of australian industrial stock returns. *Australian Journal of Management*, 29(1):121–145, 2004.

[142] Paul D. Yoo, Maria H. Kim, and Tony Jan. Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In *CIMCA-IAWTIC'06*, pages 835–841, Washington, DC, USA, 2005. IEEE Computer Society.

[143] E. Zivot and D.W.K. Andrews. Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics*, 20(1):25–44, 2002.

[144] Michael Zoumboulakis and George Roussos. Escalation: complex event detection in wireless sensor networks. In *Proceedings of the 2nd European conference on Smart sensing and context*, EuroSSC'07, pages 270–285, Berlin, Heidelberg, 2007. Springer-Verlag.

[145] L. Zunino, M. Zanin, B.M. Tabak, D.G. Pérez, and O.A. Rosso. Complexity-entropy causality plane: A useful approach to quantify the stock market inefficiency. *Physica A: Statistical Mechanics and its Applications*, 389(9):1891–1901, 2010.

[146] Luciano Zunino, Massimiliano Zanin, Benjamin M. Tabak, Daro G. Prez, and Osvaldo A. Rosso. Forbidden patterns, permutation entropy and stock market inefficiency. *Physica A: Statistical Mechanics and its Applications*, 388(14):2854 – 2864, 2009.

Index