Algorithmic Trading Based on Hidden Markov Models

— Hidden Markov Models as a Forecasting Tool When Trying to Beat the Market

UNIVERSITY OF GOTHENBURG
SCHOOL OF BUSINESS, ECONOMICS AND LAW

Bachelor's Thesis in Industrial and Financial Management
School of Business, Economics and Law
University of Gothenburg, Sweden
Spring term, 2016

Supervisor:
Ted Lindblom

Authors:
Josephine Cuellar Andersson 19910801
Linus Fransson 19930127

Abstract

Introduction – All actors in the financial market strive towards earning risk-adjusted excess return. The recent decades new technology development have revolutionised financial markets and today's actors are using advanced computer technology to develop trading algorithms in the pursue of earning excess returns. The trading algorithms are often based on statistical and mathematical models and the Hidden Markov Model (HMM) is one such model that has proven to be successful due to its ability to predict future price movements of financial assets.

Purpose – The purpose of the study is to evaluate the use of Hidden Markov Models as a tool in algorithmic trading in the Swedish OMX Stockholm 30 index.

Theoretical Framework – The HMM is a statistical model used to model stochastic processes and has historically been used in many areas where finance is one of them. The HMM is an extension to the Markov Model with the difference that the system includes hidden states that are studied via correlated observable states.

Method – In the study, two different trading algorithm based on the HMM were developed, a static and a dynamic. Both algorithms were backtested on two historical intraday data sets from OMXS30 in order to evaluate if the algorithms could make good predictions of future price movements and generate risk-adjusted excess return. A robustness test was also conducted to see how stable the performance of the algorithms were over time and over different market trends.

Results – The results shows that the static model has a hit-ratio larger than 50 % for the first test period but not for the second. The dynamic model has a hit ratio above 50 % for both test periods. However, neither the results from the static nor the dynamic model is statistically significant. The results also show that the two algorithms performance were inconsistent over time and that the static model has better risk adjusted excess return than index for the first period but not the second one while the dynamic model outperformed index for the second test period but not for the first one. Furthermore, the robustness test indicates that both model's hit ratio and performance were inconsistent over time.

Discussion – The static and the dynamic HMM trading algorithms can earn risk-adjusted excess return during limited time frames, but the results could not be statistical proven. However, HMM as a tool for predicting stock markets should not be ruled out as both of the models tested give indications of being useful even though they seems unstable. An important aspect to consider is that HMMs depend on patterns in historical data that can be found again in future data. Thus, if the data set used in this study reflects an efficient market, the HMM becomes obsolete.

Conclusions – It could not be concluded that the type of HMMs used in this study could perform better than random guesses of future price movements of OMXS30. Furthermore, it could not be concluded that the two trading algorithms developed in the study could generate risk-adjusted returns over time.

Keywords: Hidden Markov Model, prediction, forecast, finance, algorithmic trading, OMXS30.

Contents

Abbreviations

Frequently occurring abbreviations in this report.

AT Algorithmic Trading
CBT Computor-Based Trading
HFT High-Frequency Trading
HMM Hidden Markov Model
OMXS30 OMX Stockholm 30

Nomenclature

Symbol	Description
T	Number of observations in the data set
N	Number of different hidden states
M	Number of different observations
$Y_{1:T} = \{Y_1,Y_1,...,Y_T\}$	Sequence of observations
$X_{1:T} = \{X_1,X_2,...,X_T\}$	Sequence of hidden states
$S_{1:N} = \{S_1,S_2,...,S_N\}$	Possible values of a hidden state
$V_{1:M} = \{V_1,V_2,...,V_M\}$	Possible values of an observation
$A = (a_{ij})$	Transition probability matrix of size [NxN]
$a_{ij} = P(X_t = S_j \mid X_{t-1} = S_i)$	The probability of going from a hidden state S_i to a hidden state S_j
$B = (b_{ij})$	Emission probability matrix of size [NxM]
$b_{ij} = P(S_t = V_j \mid X_t = S_i)$	The probability of an observation V_j given a hidden state S_i
$\pi = (\pi_i)$	Initial probability of a state S_i of size [1xN]
L	Learning length when training the HMM
Δ	Parameter when defining the observable states
l	Prediction length in the dynamic training algorithm

Word List

Explanation of words in the report.

Deterministic Mathematical term describing a system where there is no randomness describing the time development of the system.

Discrete Discrete is the opposite of continuous and are hence separate, distinct and individual.

Hidden state A non-observable state used in the HMM.

In-sample testing Testing the HMM on the same data for which the HMM has been optimised.

Machine learning A subfield of computer science where algorithms are developed based on mathematical models. The parameters of the mathematical models are trained on a set of data. When the model has been trained and the right parameters have been found, it is possible to make predictions about future data.

Markov chain A stochastic model describing a sequence of possible events such as the probability of each event depends only on the state presumed in the previous event.

Markov property Also called memorylessness in the sense that the next state simply depends on the current one, and no previous states.

Out-of-sample testing Testing the HMM on different data than for which the HMM has been optimised.

Recursive Self-repeating. In computational science, a recursive method calls upon itself implying that the solution to a problem depends on solutions to smaller parts of the same problem.

Stochastic Unpredictable or random.

Chapter 1 Introduction

The first chapter aims to give an introduction and background to the research area and the studied problem. Next, the purpose and research questions of the study are presented along with the delimitations.

1.1 Background

The development of new technology has revolutionized the functions of financial markets and the way financial assets are traded (Hendershott and Riordan, 2009). Over the years, the stock exchange systems have changed in order to take advantage of new technology and to satisfy the constantly changing needs of the marketplace (Hasbrouck et al., 1993). Trading financial assets have gone from needing face-to-face interaction at a physical location to an automated process conducted by computers (Hendershott, 2003).

Beginning in the 1970s, the trading process started to become computerized as the first fully integrated computerized trading system, called the NASDAQ, was implemented in the U.S (Furse et al., 2011). In the 1980s, the computerization of the financial market continued as the use of Computer-Based Trading (CBT) systems were exploited. Functions that previously were performed by humans, such as monitor financial data (e.g. stock prices) and issue buy and sell orders, were now carried out by computers (Furse et al., 2011). The simplest and most primitive CBT system, called program trading, could automatically issue buy and sell orders when the stock price rose above or below a pre-determined trigger price (Furse et al., 2011). Such trading systems were actually blamed for the so called Black-Monday crash in October 1987 (Bookstaber, 2007). Ever since, the use and effects of CBT systems have been a target for extensive research and discussion.

Since the 1980s, the development of new IT and computer technology together with the substantially decrease in cost of computer-power, have made the use of CBT to grow substantially (Furse et al., 2011). Today, the use of CBT systems is widespread, from large hedge-funds and financial institutions to private investors. Computers are thus key players in financial markets (Lin, 2014).

The technology development during the past decade, have made the CBT system more complex and intelligent (Furse et al., 2011). A CBT system could be based on a variety of different trading strategies where algorithmic trading (AT) and its sub-set High-Frequency-Trading (HFT) are two such types (Furse et al., 2011). Algorithmic trading is defined by Hendershott and Riordan (2009) as 'The use of computer algorithms to automatically make certain trading decisions, submit orders, and manage those orders after submission' (p.2). Hence, algorithmic trading is in its simplest form a system where a financial asset is bought or sold based on predefined instructions and parameters

(Finansinspektionen, 2012). Algorithmic trading is widely used in the financial markets across the world and in many areas, the algorithmic trading stands for more than the majority of the market volumes. For instance, in August 2011, 55 percent of the closings at the Stockholm Stock exchange was executed by computer based algorithms (Isacsson, 2011).

1.2 Problem Discussion

All actors in the financial market strive towards one thing; to earn risk-adjusted excess rates of return. However, Fama (1970) argues that it is impossible to beat the market, as financial markets are efficient and the price of an asset or security always fully reflects the available information. In other words, it is not possible to beat the market by using any information that the market has knowledge about. Thus, it is meaningless to use fundamental analysis (i.e. analysis of financial reports and performance of companies in order to find undervalued stocks) or technical analysis (i.e. analysis of past prices in order to predict future prices) as it would not generate greater returns than a randomly selected stock portfolio with equal risk (Malkiel, 2003).

However, the effective-market hypothesis developed by Fama (1970) has been object for harsh questioning and several researchers have shown that financial markets are not always efficient. An example is the January effect discovered by Thaler (1987), which states that the stock prices of small companies generally increase during the month of January. Other examples are the bandwagon effect (i.e. people adopt behaviour from others) described by Shiller (2000). Other findings made by Lo et al. (2000) show that historical patterns such as double bottoms and head and shoulders formations can have some predictive effect of future prices.

In line with the likelihood that financial markets are not efficient at all times, actors in the market try to exploit such inefficiencies in order to gain risk-adjusted excess rates of return. According to Malkiel (2003), one such way is trying to predict trends and future prices of financial assets on the market. Many financial economists and mathematicians assume that future prices can at least be somewhat predicted and argue that trends and future stock prices are dependent on psychological and behavioural elements and that they can be predicted on the basis of past financial data series (Malkiel, 2003). Traditionally, statistical and advanced mathematical models have been used in order to predict such volatile financial time series (Granger and Newbold, 1986). Today, statistical and mathematical models are still used and due to the previously mentioned technology developments, they are far more complex than ever before.

Several old studies assume that the relationship between available information (e.g. historical time series) and future trends are linear (Enke and Thawornwong, 2005). For example, both Balvers et al. (1990) and Ferson (1989) uses linear-regression models in attempts to predict future time series. However, it is today widely accepted among researchers that financial times series are non-linear and thus difficult to predict (Enke and Thawornwong, 2005).

Realistic models of financial time series can be developed and such models are often based on

the Hidden Markov Model (HMM) (Mamon and Elliott, 2007). The HMM is a statistical tool with the ability to make good predictions of non-linear trends and account for high volatility changes (Kavitha et al., 2013). Several researchers have applied HMMs in order to analyse and predict economical trends and future prices of financial assets. Hassan and Nath (2005) used an HMM to forecast stock prices for interrelated markets. Idvall and Jonsson (2008) applied an HMM in order to forecast movements in a currency cross. Kavitha et al. (2013) used an HMM to forecast future trends on the stock market and Nguyen and Nguyen (2015) applied HMM for stock selection on S&P500. Furthermore, HMMs have also been used with great success on its own or in combination with other mathematical tools to predict of equity indices, such as the S&P500 index (Zhang, 2004). However, research using HMMs in order to predict future prices and trends of Swedish equity indices are lacking. There are several different equity indicies on the financial market in Sweden, and the most widespread is the OMX Stockholm 30 (OMXS30) that consists of the 30 largest companies due to market value at the Nasdaq OMX Stockholm.

1.3 Purpose and Research Questions

The purpose of the study is to evaluate the use of Hidden Markov Models as a tool in algorithmic trading in the Swedish OMX Stockholm 30 index. The purpose of the study can further be broken down into two research questions:

- Can an algorithm based on a Hidden Markov Model make good predictions of future price movements of the OMX Stockholm 30 index?
- Can an algorithm based on a Hidden Markov Model earn risk-adjusted excess rates of return in relation to the OMX Stockholm 30 index?

1.4 Delimitation

Since the study is under time constraint, the lack of time will limit the testing of the developed algorithm to historical data as the time available for tests on real-time data would be too short to get significance in the results.

Moreover, several different types of HMMs can be applied in a trading algorithm, from extremely complex to fairly simple ones. Since the study was under time constraint, it was not possible to study many different forms of HMMs, and the study was therefore delimited to concern only a time-discrete HMM with a finite number of states.

Chapter 2 Theoretical Framework

In this chapter, the theoretical groundwork of the report is laid. The chapter starts with some basic knowledge about algorithmic trading which is followed by pitfalls of backtesting and how Sharpe Ratio can be used to evaluate the performance of a trading algorithm. After that, a description of Markov Models in general is given which leads to the final part, the fundamentals of Hidden Markov Models.

2.1 Algorithmic Trading

Algorithmic trading, also know as algo and black-box trading, is according to Nuti et al. (2011) the process of using computers and computer algorithms to automate one or several stages of the trading process such as analysis of trading opportunities and execution of buy and sell orders in the stock market. A trading algorithm can be divided into four different steps:

- Retrieve data regarding the asset
- Analyse data
- Make decision regarding trading the asset
- Execute the trade on the market

Since a trading algorithms consists of four different steps, the definition given by Nuti et al. (2011) is broad. Normally, when talking about trading algorithms, both practitioners as well as researchers refers to it as an algorithm where all of the four mentioned steps are fully automated, which is similar to the definition given by Hendershott and Riordan (2009) in section 1.1 *Background*. However, developing a trading algorithm and fully automating the different steps is not a trivial task (Idvall and Jonsson, 2008).

Moreover, trading algorithms can, if needed, process massive amounts of information and take rapid actions based on pre-determined instructions (Lin, 2014). The instructions can vary and be based on, for instance, quantity, price or patterns (Chan, 2013). Two of the most common instructions or strategies behind trading algorithms are according to Chan (2013) momentum and mean reversion strategies.

Momentum – Strategies based on identifying trends in the stock market. This can be done by computing the moving average or by performing other technical analysis. A trade based on assumptions about future prices can then be executed.

Mean Reversion – Strategies based on the hypothesis that a high or low price of an asset is temporary and soon will revert to its mean. Under this assumption, a price range can be defined such that the algorithm takes a long or short position if the price is lower or higher than the bounds of the defined range.

However, momentum and mean reversion are two fairly simple strategies, but trading algorithms can be far more complex (Lin, 2014). For instance, the algorithm can be based on machine learning, a sub-field of computer science, which concerns the usage of algorithms based on mathematical models that can learn from historical data in order to make predictions about future data (Marsland, 2015).

The use of algorithmic trading is as previously mentioned widespread and according to Lin (2014), almost all large financial institutions use algorithmic trading in some way. One of the benefits of trading algorithms, compared to manually trading by humans, is that trading algorithms are not affected by feelings and therefore makes rational decisions (News, 2016). Another benefit with trading algorithms are that they can be tested on historical data (Chan, 2009). Such testing is normally conducted after an algorithm has been developed and based on the results, the algorithm can either be further developed or implemented on the stock market, in real-time.

2.2 Backtesting

Backtesting is the process of testing a trading algorithm on historical data in order to see how it would have performed in the past during a specified time frame (Chan, 2013). The assumption of a backtest is thus that an algorithm performing well in the past has a better chance of performing well in the future (Ni and Zhang, 2005). Furthermore, during a backtest, the investor also has the opportunity to improve a trading algorithm. Thus, Ni and Zhang (2005) argues that backtesting can give valuable information and knowledge about an algorithm's potential performance in the market. However, it must be clear that since backtesting is performed on historical data, nothing can with absolute certainty be said about an algorithms future performance, but rather only give indications of the potential profitability in the future (Ni and Zhang, 2005). Moreover, conducting a backtest may seem like an easy thing to do, but according to Davey (2014), most people are doing backtests incorrectly and there are several potential pitfalls that need to be avoided.

Look ahead biases - Look ahead biases is according to Chan (2013) a common backtesting error that appears if an trading algorithm uses future information in order to make forecasts and decisions at current time. An example of such bias is if an algorithm, during backtesting, utilises a day's opening and closing prices, when making decision about taking a position on that very same day. Thus, the error is basically a programming mistake that only can appear when backtesting, as it in a real-time test would be impossible to use future information.

Data-snooping biases - According to Chan (2013), data-snooping (also referred to as data mining or overfittning) is another common bias in backtesting. The bias refers to the effect that occurs when an algorithm has too many free parameters that are fitted or tuned to the historical market patterns in order to make the historical performance look good. However, such random market patterns are not likely to repeat themselves in the future, and an algorithm with many free parameters fitted to such patterns will have low predictability of future trends and prices (Sullivan et al., 1999). The method to avoid data-snooping biases is according to Davey (2014) to test the algorithm on a so called out-of-sample data set (i.e. a data set that has not been used for training or optimisation of the algorithm). Preferably, the testing should be conducted on several out-of-sample data sets in order to get statistical adequate results (Chan, 2013).

Short-Sale constraints - Another bias that is important to be aware of is, according to Chan (2013), the short-sale constraint. The constraint concerns the fact that some financial assets are difficult to short which is important to be aware of when back-testing a trading algorithm with an option to short.

Survivorship bias - Another common pitfall when backtesting is according to Davey (2014) the surviovorship bias. The bias appear when the data used for backtesting do not include delisted stocks and this can, according to Chan (2013), cause highly misleading results. For example, if running an algorithm on real time data, it is not possible to know which of the stocks that will survive in the future. However, when backtesting, most of the data provided by databases only contain stocks that are alive today and a test on such data would therefore give much better results than a test on real time data. The bias can be seen as a form of the look-ahead bias as it takes future information into consideration. Worth noticing is that indices retain the performance of delisted stocks until the day that they are substituted of a new firm (SVD, 2015) and are thus per default survivorship bias free.

Dividends and split adjustments - According to (Chan, 2013), another factor to be aware of is that the data used for backtesting should be adjusted for dividends and splits as it otherwise can cause deceptive results. The normal approach to conduct such adjustment is to reinvest the dividends and to adjust the value of the stock to the number of shares outstanding.

2.3 Evaluation of Backtesting Results

After an algorithm has been backtested, it is important to evaluate the results. Several different measures are available to measure the performance of a trading algorithm, where the Sharpe Ratio developed by Sharpe (1966) is one of the best known. However, one of the key assumptions of the Sharpe ratio is that returns are normally distributed. Several researchers have questioned this ((Lo et al., 2000); (Bailey et al., 2014); (Sharpe, 1994)) and Lopez de Prado and Peijan (2004)

show that the daily returns are not always normal distributed, especially not in the case of hedge funds strategies including long/short of financial assets. However, Eling and Schuhmacher (2007) present strong evidence that the Sharpe Ratio is a good measure compared to other more complex measures under highly non-normal distributions.

When comparing the return of two different assets, it is better to compare their relative returns, where the return is adjusted for the risk taken, than comparing their absolute return (Sharpe, 1994). The Sharpe Ratio developed by Sharpe (1994) is a ratio measuring how well the asset's return compensates the investor for the risk involved in trading an asset. The Sharpe Ratio is defined as:

$$SR = \frac{E[R_a - R_b]}{\sqrt{var[R_a - R_b]}} = \frac{\mu_a - \mu_b}{\sigma_{ab}} \tag{2.1}$$

where R_a represents the asset return and R_b represents the return of a benchmarked asset, normally the risk-free rate of return or a index. Thus, $E[R_a - R_b]$ is the expected excess return and σ_{ab} is the standard deviation of the excess return. Furthermore, the Sharpe ratio is also closely related to the t-statistic used for hypothesis testing. The t-statistic can according Sharpe (1994) be calculated using (2.2).

$$t_{statistic} = SR \cdot \sqrt{T} \tag{2.2}$$

where T is the number of returns used in the calculation. However, it should be noted that this is only possible under the assumption that the returns are independently and identically distributed (Lo, 2002).

2.4 Markov Models

A Markov Model is a stochastic model used for modelling systems based on stochastic processes. Due to the uncertainty in the system, a probability distribution is taken on in order to describe the set of possible outcomes (Bhar et al., 2004). According to Axelson-Fisk (2010), a stochastic process is simply the evolution of a stochastic variable in time such that the jump between two different states is random. A first order Markov process, has the property that the next state can be determined solely from the present state without any knowledge of the previous ones. This property can be referred to as the memorylessness or simply the Markov property (Axelson-Fisk, 2010).

Markov Models are often used to find patterns appearing over a space of time. Axelson-Fisk (2010) argues that Markov models are popular due to their flexibility, implying that many processes can be approximated as Markov chains. The word chain may indicate a discrete state space. However, a Markov chain can also describe a continuous state space. In this study, solely time-discrete Markov models with a finite set of states will be consider.

Consider a Markov chain process with N possible states $S = \{S_1, S_2,...,S_N\}$. The time is discrete and can be described by $t = 1,2,...,T$. Let X_t denote the state that the system is in at time

point t. Using the definition of conditional probability (i.e. the probability that an event B will occur given that an event A has already occurred), the probability of finding a sequence of random variables $X = \{X_1, X_2,...,X_T\}$ for the state sequence $i_1,i_2,...,i_T \in S$ can be computed accordingly:

$$\begin{aligned} P(X_1 = i_1,...,X_T = i_T) =& P(X_T = i_T|X_{T-1} = i_{T-1},X_{T-2} = i_{T-2},...,X_1 = i_1) \\ & \cdot P(X_{T-1} = i_{T-1}|X_{T-2} = i_{T-2},...,X_1 = i_1) \\ & \dots \\ & \cdot P(X_2 = i_2|X_1 = i_1) \cdot P(X_1 = i_1) \end{aligned} \tag{2.3}$$

A first order Markov model satisfies the Markov property described above. According to Axelson-Fisk (2010), the definition of a Markov chain can be written as:

Definition of Markov chains Given a state sequence $i_1,i_2,...,i_t \in S$, the process $(X_1,X_2,...)$ is defined to be a Markov chain if it satisfies the Markov property such that

$$P(X_t = i_t|X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2},...,X_1 = i_1) = P(X_t = i_t|X_{t-1} = i_{t-1}) \tag{2.4}$$

The probability of a sequence X generated by a Markov chain can thus be written as

$$P(X_1 = i_1,...,X_T = i_T) = P(X_1 = i_1)\prod_{t=2}^{T} P(X_t = i_t|X_{t-1} = i_{t-1}) \tag{2.5}$$

To fully describe the model, two parameters called the initial probability distribution π and the transition matrix A need to be formated. Axelson-Fisk (2010) defines these parameters accordingly:

Definition of π and A The probability of the first state X_1 is determined by the initial probability distribution π, which is an N-dimensional vector with one probability for each state. π satisfied the following properties

$$\pi_i = P(X_1 = S_j), \quad j \in S \tag{2.6}$$

$$\sum_{i=1}^{N} \pi_i = 1$$

The process proceeds according to the transition matrix $A = (a_{ij})$. A is a matrix of size $[NxN]$ and consists information about the probability of going from one state to another such that

$$a_{ij} = P(X_t = S_j|X_{t-1} = S_i), \quad i,j \in \{1,2,...,N\} \tag{2.7}$$

The matrix describes a stochastic process meaning that the elements are non-negative and each row sums up to one

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{2.8}$$

Given a Markov model, the system can be developed in time. However, it should be noted that the process is not deterministic, but stochastic, and it is therefore possible to get different outcomes with each simulation.

2.5 Hidden Markov Models

In some applications, the Markov Model has limited power. It is therefore useful to extend the general Markov model into a Hidden Markov Model (HMM). HMM is an extension of Markov Models in the sense that an HMM also is a statistical model used when the investigated system is assumed to be a Markov process. What differentiates an HMM from a Markov model in general is that the system includes unobservable (or hidden) states. Rabiner and Juang (1986) describes an HMM as a doubly stochastic process where one of the underlying stochastic processes is hidden. The hidden process is a Markov chain going from one state to another, however, the states can not be observed directly but only via an observed process that is correlated to the hidden process (Bhar et al., 2004). According to Axelson-Fisk (2010), the observed process does not have to be a Markov chain.

HMM was first used in speech recognition but has throughout the years been applied to many other areas, where finance and forecasts in the stock market is one of them (Kavitha et al., 2013). The motivation for using HMM in finance is the challenge of good predictions due to non-linear trends and sudden changes in volatility in the stock market (Kavitha et al., 2013).

2.5.1 A Conceptual Description

Suppose the weather is to be determined in a location far away from the current position, such that the weather cannot be directly observed and thus is hidden. Moreover, assume there is simply two different possible states of the weather; it is either sunny or rainy. What is known (observable) is the temperature at the location far away. The temperature is then known as the observation whilst the weather is the hidden state.

The weather at time t can only be estimated based on how probable each possible state (sunny or rainy) is, given a previous state at time $t-1$. This is why a conditional probability has to be formed. The probabilities are more easily computed using:

- A transition probability matrix, representing the probability of going from one hidden state to another
- An emission probability matrix, representing the probability of an observable state given a hidden state

Each probability in the two matrices is time independent, meaning that the probabilities will not change as the system evolves in time. A more rigorous representation of the system will be presented in section 2.5.2 *The Mathematics*.

The HMM system can be graphically represented using a Trellis diagram, see Figure 2.1. $X = \{X_1,X_1,...,X_T\}$ is the sequence of hidden states (the weather) given by a first order Markov process and $Y = \{Y_1,Y_2,...,Y_T\}$ is the sequence of observations (the temperature). The horizontal arrows

represent a transition between two sequential hidden states, whilst the vertical arrows represent an observation. It should be pointed out that a state X_{t+1} does not depend on any previous observations nor any previous states except X_t.

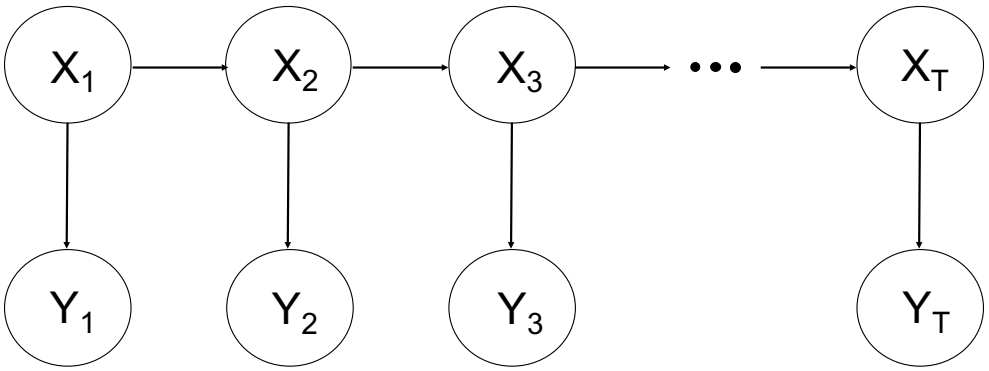

Figure 2.1: Trellis diagram representing X as the hidden sequence of states and Y as the sequence of observations.

2.5.2 The Mathematics

After the conceptual description in the previous section, a more rigorous mathematical representation of the HMM will be given. Let X denote the Markov process as before, however, it is now called the hidden process. The sequence of hidden states can take on the discrete set of states S which has the initial probability distribution π. A transition from one hidden state to another is represented by the transition probability matrix $A = (a_{ij}), \quad i,j \in S$. At each time step t, with a hidden state $X_t = i, \quad i \in S$, there is an emitted observation Y_t, taking on the possible values $V = \{V_1, V_2,...,V_M\}$. Each observation Y_t only depends on the current hidden state X_t (Rabiner, 1989), as indicated by Figure 2.1. The emission probability, namely the probability of a certain observation given a hidden state, is reflected by the emission probability matrix $B = (b_{ij})$, where each element is given by

$$b_{ij} = P(Y_t = V_i | X_t = S_j), \quad i \in \{1,2,...,M\},\ j \in \{1,2,...,N\} \tag{2.9}$$

To sum it up, the complete model can be described using the following elements:

Symbol	**Description**
T	Number of observations in the data set
N	Number of different hidden states
M	Number of different observations
$Y_{1:T} = \{Y_1,Y_1,...,Y_T\}$	Sequence of observations
$X_{1:T} = \{X_1,X_2,...,X_T\}$	Sequence of hidden states
$S_{1:N} = \{S_1,S_2,...,S_N\}$	Possible values of a hidden state
$V_{1:M} = \{V_1,V_2,...,V_M\}$	Possible values of an observation
$A = (a_{ij})$	Transition probability matrix of size [NxN]
$a_{ij} = P(X_t = S_j \mid X_{t-1} = S_i)$	The probability of going from a hidden state S_i to a hidden state S_j
$B = (b_{ij})$	Emission probability matrix of size [NxM]
$b_{ij} = P(S_t = V_j \mid X_t = S_i)$	The probability of an observation V_j given a hidden state S_i
$\pi = (\pi_i)$	Initial probability of a state S_i of size [1xN]

The following properties of A, B and π must be satisfied:

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{where } 1 \leq i \leq N \tag{2.10}$$

$$\sum_{j=1}^{M} b_{ij} = 1 \quad \text{where } 1 \leq i \leq M$$

$$\sum_{i=1}^{N} \pi_i = 1 \quad \text{where } \pi_i \geq 0$$

The HMM can then be represented by the two matrices A and B and the vector π. A more compact representation of the HMM can thus be written as $\lambda \equiv \{A,B,\pi\}$ (Rabiner and Juang, 1986). Just like Axelson-Fisk (2010) writes, the joint probability of a hidden sequence including its observable sequence is thus given by

$$P(X,Y|\lambda) = \pi_{X_1} b_{X_1}(Y_1) \prod_{t=2}^{T} a_{X_{t-1},X_t} b_{X_t}(Y_t|X_{t-1},...,X_1) \tag{2.11}$$

After modelling a system as an HMM, the parameters have to be initialised. The transition matrix along with the emission matrix can be estimated. A good model is found after training the model on historical data and finding the right parameters, adjusted to the specific system and its patterns.

2.5.3 The Baum-Welch Algorithm

When an HMM has been formed, the parameters have to be estimated such that they reflect the system in the best possible way. According to Rabiner (1989), optimising the parameters is the most critical part when developing the model and it is by adjusting the parameters that the best model is found. The probability of the observation sequence is then maximised. More specifically, the parameters can be found by deriving the maximum likelihood estimate given a sequence of observations. As implied, the problem is never solved exactly, however, using the Baum-Welch algorithm, a local maximum likelihood can be found (Rabiner and Juang, 1986).

2.5.4 The Viterbi Algorithm

The Viterbi algorithm, developed by Viterbi (1967), is a recursive method used to estimate the state sequence of a time-discrete, finite state Markov process (Forney, 1973). That is, the Viterbi algorithm can be used as a tool for decoding and finding the most probable state sequence X given an HMM with its parameters λ and an observed sequence Y (Bhar et al., 2004).

Finding the most probable state sequence can be useful in two scenarios. The first scenario denotes extrapolation whilst the second one denotes interpolation. More explicitly, the Viterbi algorithm can be used when an observation sequence is given along with a fully trained HMM and the most probable hidden sequence, cohered to the observations, is wanted. The extrapolation intend to make predictions about the future, which is the more relevant situation in this study.

2.5.5 Issues with Hidden Markov Models

Rabiner and Juang (1986) have identified some common issues with HMM. In this section, two issues relevant for this particular study is presented.

Small parameters – When using a finite set of training data, the parameters A and B can be set to zero if no occurrence is found. According to Rabiner and Juang (1986), it is then essential to understand if the small parameters depend on the data set being too small or if there is no pattern to be found. If one suspects the training set being too small, then efforts must be made to insure that no parameters becomes too small. However, if the effect of small parameters is a real effect, depending on no occurrence, then a zero probability parameter is reasonable.

Non-ergodic models – An ergodic model allows transitions between any hidden states. However, in some cases, the model becomes non-ergodic imposing a transition matrix with 100 % probability of going from one state to another (Rabiner and Juang, 1986), such as going from state 3 to state 4. This implies a deterministic jump since there is no probability of jumping to

any other state than to state 4. There is not much to do about these models more than being aware of the meaning of such transition matrix.

Chapter 3 Method

In order to achieve the purpose of the study, two major parts were conducted. Firstly, a trading algorithm was developed and secondly, the developed algorithm was backtested. As stated in the delimitation section above, the time frame of the thesis was too short to test the algorithm on real-time data with desirable statistical significance. Therefore, historical data were used instead.

3.1 Research Strategy

A research strategy should, according to Olsson and Sörensen (2011), be chosen based on the research question and purpose. In order to answer the research questions, collection and analysis of numerical data were needed. According to Bryman and Bell (2011) a quantitative research strategy is, therefore, a good choice. Furthermore, the study was of deductive approach as the research questions were deduced based on previous knowledge in the field and subject of empirical testing, which further indicates that a quantitative research strategy is preferable (Bryman and Bell, 2011). However, the study had some inductive elements as well. For example, the trading algorithm was not developed based on a predetermined structure. Instead, the algorithm was developed through an iterative process. However, the use of inductive elements in a research strategy with a deductive approach is not unusual (Bryman and Bell, 2011).

3.2 Data Collection

The data collection in the study was divided into two parts. The first was secondary data collection of literature and the other was the collection of secondary numerical data needed for the backtesting.

3.2.1 Literature Review

A literature review was conducted in order to gain specific knowledge in the chosen research field and lay the groundwork of the theoretical framework. Furthermore, the literature review was also used to verify that the purpose and research questions were relevant and important, which is proposed by Bryman and Bell (2011).

The literature review was initiated by examining literature covering the financial market, its development over time and the efficient market hypothesis. After that, the review continued within a more delimited area of algorithmic trading in general and algorithmic trading based on

HMMs in particular. Furthermore, all literature were reviewed with a critical approach, which according to Bryman and Bell (2011) is important to avoid a biased view on previously written literature.

3.2.2 Financial Data

The data in the study consisted of two different time series. The first one was end of the day data of OMXS30 from 30 September 1986 to 20 May 2016 downloaded from Nasdaq (2016) used for the optimisation of the algorithm and the robustness test later described. The second consisted of intraday data of OMXS30 used for the backtesting of the developed algorithm.

Intraday data in the backtesting was used since it was considered a better simulation of a real time scenario as the developed algorithm was designed to predict one day ahead predictions and use such prediction to take either a long or a short position in the market. Hence, if such predictions were made on daily open and close prices and a position was taken the same day, the backtesting would suffer from look-ahead bias. Thus, in order to avoid the bias and simulate real time conditions as good as possible, intraday data were used. However, the downside of using intraday data for backtesting was that such data are much more difficult to retrieve, especially for longer time period. This resulted in the time series for backtesting being only about 3 years long.

The original data set used for the backtesting consisted of minute data of OMXS30 from 2 January 2013 to 15 April 2016 downloaded from TheBonnotGang (2016). However, since the algorithm was designed to conduct predictions one day ahead, only one data point for each trading day was needed. The data point chosen was the minute opening and close data at 12.00 each trading day. The selected time of the day could basically been any time from 9.01 to 17.29, but in order to avoid the volatility in the beginning and ending of each day (Chan, 2009), a time in the middle of the day seemed reasonable. However, a few days a year, the OMXS30 closes at 13.00 due to half days and a time point before that was therefore suggested. Thus, 12.00 was considered to be a good time to use.

As previously mentioned, the algorithm was designed to make price predictions one day (8.5 trading hours) ahead. Thus, problems appeared when some of the trading days were only half days (closing at 13.00). In order to solve that and get the same length of all trading days, the price of the index was assumed to be constant from 13.00 until 17.30 on half trading days. However, since there are only a few half trading days every year, the assumption was considered to have minor effects on the results.

Moreover, since the data used was from an index, survivorship bias mentioned by Chan (2009) could be avoided which otherwise could have caused deceptive results. The data used were also adjusted so that dividends would not affect the test result as proposed by Chan (2009).

3.3 The Algorithm

Two versions of the algorithm based on an HMM were developed; one static and one dynamic.

3.3.1 MATLAB as Development Platform

The algorithm was developed and tested in MATLAB, which is one of the most common back-testing platforms used by quantitative analysts at financial institutions (Chan, 2009). One of the key advantages of MATLAB is that it contains several advanced mathematical and statistical functions and toolboxes which was used in the study. Along with the fact that MATLAB is very efficient performing matrix calculations, the program was regarded suitable for this study.

The developed algorithm used the Statistics and Machine Learning Toolbox in MATLAB. More specifically, the following functions were used:

- **hmmestimate(Y, X)** – Given a sequence of observations Y and the corresponding sequence of hidden states X, the function returns estimates of the transition and emission matrices.
- **hmmtrain(Y, A, B)** – Based on two initial estimations of the transition matrix A and emission matrix B, the function calculates the maximum likelihood estimates of A and B, based on the observation sequence Y, using the Baum-Welch algorithm.
- **hmmviterbi(Y, A, B)** – Given a well trained model, this function calculates the most probable state path for a hidden Markov model using the Viterbi algorithm.

3.3.2 Choice of Observable and Hidden States

When the HMM was developed, a decision regarding the number of hidden and observable states had to be taken. According to Rabiner and Juang (1986), this part can be very difficult and could involve trial and error before a good model size is found.

At 12.00 each trading day, a prediction about future price movement was to be conducted. Thus, the hidden states were chosen to reflect the price change from 12.01 the current day t to 12.00 the following day, at time $t + 1$. This is formalised in (3.1).

$$\text{Upcoming price movement} = \text{Opening price}(t + 1) - \text{Closing price}(t) \tag{3.1}$$

where 'Opening price' represents the index price at 12.00 and 'Closing price' indicate the index price at 12.01. Thus, a positive movement indicate a positive daily return in index.

The number of states was chosen to two, see Table 3.1, indicating either a drop or a rise in OMXS30 until the following day.

Table 3.1: The defined hidden states. There are only two different states, representing a drop or a rise in the index price until the upcoming day at time $t+1$.

Hidden state	**Meaning**	**Definition**
1	Drop	Upcoming price movement < 0
2	Rise	Upcoming price movement ≥ 0

Furthermore, the observable states were chosen to reflect the two variables 'Price movement' and 'Mean displacement'. The idea behind the choice was based on the strategies *Momentum* and *Mean reversion* described in section 2.1 *Algorithmic Trading* . 'Price movement' was meant to relate to the momentum strategy saying that the trend is likely to continue. 'Mean displacement' was instead related to the mean reversion strategy which assumes that the price of an asset is pending around its mean.

'Price movement' reflected the intraday change in price of the asset at time t, where t once again represented the current day. The movement in price at time t can thus be written as

$$\text{Price movement} = \text{Opening price}(t) - \text{Closing price}(t-1) \tag{3.2}$$

The variable had three different states; drop, constant or rise. Since the probability of a price movement equal to zero was considered very small, a movement smaller than a value $\Delta \geq 0$ was considered to be a movement equal to zero. The three states related to the variable 'Price movement' is shown in Table 3.2.

Table 3.2: The defined states for the variable 'Price movement'.

State	**Meaning**	**Definition**
1	Rise	Price movement $> \Delta$
2	Constant	$-\Delta \leq$ Price movement $\leq \Delta$
3	Drop	Price movement $< -\Delta$

The second variable 'Mean displacement' was dependent on the moving average of the 10 preceding days' closing price. The second variable was defined as the current day's displacement from the moving average according to (3.3).

$$\text{Mean displacement} = \text{Opening price}(t) - \text{Moving average}_{10}(t) \tag{3.3}$$

The different states for this variable was then defined to be higher, equal or lower than the moving average. Once again, the small value Δ was used to define a range where the displacement from the moving average was considered to be small and thus fell into the state 'equal'. The three states related to the variable 'Mean displacement' is shown in Table 3.3.

Table 3.3: The defined states for the variable 'Mean displacement'.

State	Meaning	Definition
1	Higher	Mean displacement $> \Delta$
2	Equal	$-\Delta \leq$ Mean displacement $\leq \Delta$
3	Lower	Mean displacement $< -\Delta$

The then observable states were defined as different combinations of the two variables 'Price movement' and 'Mean displacement'. Since there was two variables with three states each, there was nine possible states in total, see Table 3.4.

Table 3.4: The defined observable states. In total there are nine different states, which are combinations of the two variables with three possible states each. The total number of combinations thus becomes nine.

Observable state	Price movement	Mean displacement
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3
7	3	1
8	3	2
9	3	3

To return to the notation presented in section 2.5.2 *The Mathematics*, the number of observable states $M = 9$, with observation vector $V = \{1, 2, 3,...,9\}$ and the number of hidden states $N = 2$ with hidden states vector $S = \{1,2\}$.

3.3.3 Static Training Algorithm

Once the observable and hidden states were defined, the model was implemented in MATLAB. The algorithm is presented in pseudo code below.

- Load data
- Get observed sequence and hidden sequence given the data
- Get model
  - Estimate the transition and emission matrix with the MATLAB function hmmestimate()

- Train the model with the MATLAB function hmmtrain()

- For each upcoming day
  - Make a prediction using the model and the function hmmviterbi()
  - Trade the asset and calculate the return

The static training algorithm thus performs training on a set of data to find a proper model. The same model is then used to make prognosis about one day at a time, one day in advance, allowing the algorithm to perform a trade of the asset at 12.01.

3.3.4 Dynamic Training Algorithm

In the previous section, an algorithm based on static learning was developed. What defines the static training algorithm is that it finds one model λ which is then used throughout the time series to make predictions. In this section an algorithm based on dynamic learning will be developed. As implied by the name, a dynamic training algorithm depends on dynamic training data which moves along with time such that many different models λ, with different parameters A and B, are found. The dynamic training is illustrated by Figure 3.1.

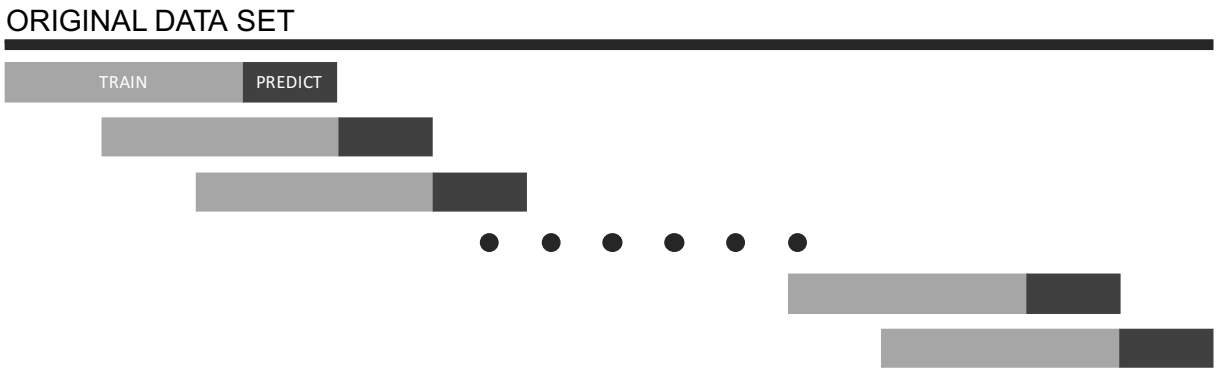

Figure 3.1: The figure represents a dynamic learning pool. One model is found for the first training, illustrated by the light gray box. Prediction (dark grey box) about the upcoming day is made and the training sequence is then updated with information from the most recent trading day. This will be repeated until the end of the data set is reached. Each training set will produce a new HMM.

Just as the static training algorithm, the dynamic training algorithm is dependent on the length of the training data L and Δ. Moreover, the dynamic training algorithm is dependent on the prediction length l, defining the number of days the same model should be used for predictions. In Figure 3.1, l represents the number of days given by a dark gray box.

In pseudo code, the algorithm based on dynamic training can be described accordingly

- Load data
- Get observed sequence and hidden sequence given the data

- While the time series has not reached the end
  - Estimate the transition and emission matrix with the MATLAB function hmmestimate()
  - Train the model with the MATLAB function hmmtrain()
  - With the found model, make a prognosis about l number of days using the function hmmviterbi()
  - For the l upcoming days, trade the asset and calculate the return

3.3.5 Determining the Parameters

There are two central parameters in the models. The first one is Δ, defining the resolution of the model's observable states. The second parameter is the length of the learning data L. To study the model's behaviour in relation to the two parameters, the algorithm was iterated for different values of Δ and L. The more appropriate values was then used for the backtesting. The parameters were chosen to maximise the ratio of correct movements along with the mean value of the developed capital based on the trading algorithm.

The parameter test was done on a completely different set of data than what was used for backtesting, to avoid any data-snooping. The data used for parameter testing was performed on the closing price of OMXS30 of each trading day ranging from 1 August 2011 to 28 December 2012.

Furthermore, the dynamic training algorithm is dependent on the prediction length l. This parameter was chosen to 1 trading day to obtain the most dynamic model possible.

3.3.6 Investment Strategy

In order to make money based on the developed price prediction tool, a trading strategy (i.e. when and how to take a position in the market) was needed. Since the price prediction tool was developed to make predictions of the future price movement, the investment strategy was designed to take maximal advantage of that. Therefore, the strategy was developed so that the algorithm would take a long position in index if a rise in index was predicted, and a short position if a drop in index was predicted. The prediction of the future movement was made at 12.00 and depending on the prediction, either a long or a short position was taken at 12.01. The position was then held until 12.00 the next trading day when it was liquidated, where-after a new prediction was made and a new position was taken.

3.4 Backtesting

After the trading algorithm was developed, a backtest was conducted. The main objective of the backtesting was to get insight about how the algorithm would perform in the future, on real-time data. However, as stated in the theory chapter, simulation of future performance based on historical data is difficult, and all of the pitfalls mentioned in the backtesting chapter were considered in order to minimise misleading results.

One of the most important errors to avoid was the data snooping bias. Especially as an algorithm based on HMMs has many parameters that need to be fitted. Hence, over-fitting of those parameters would have caused misleading results. A cross-validation approach was therefore used, as proposed by Chan (2013).

The algorithm was tested on two different data subsets in order to make sure that the algorithm performed well on more than one data set of OMXS30. The first set was data from 2 January 2013 to 27 June 2014 and the second was data from 22 October 2014 to 15 April 2016, both representing 370 trading days. The two data sets were then divided into three parts which can be seen in Table 3.5. Moreover, the testing of the algorithm was so called out-of-sample testing, see Figure 3.2, which means that the algorithm was not tested on the same data as it had been trained on. By doing so, the data snooping bias could also be avoided. Furthermore, since two different data sets were used, the statistical significance will according to Chan (2013) rise as the probability of random results on two test samples is lower than the result from one.

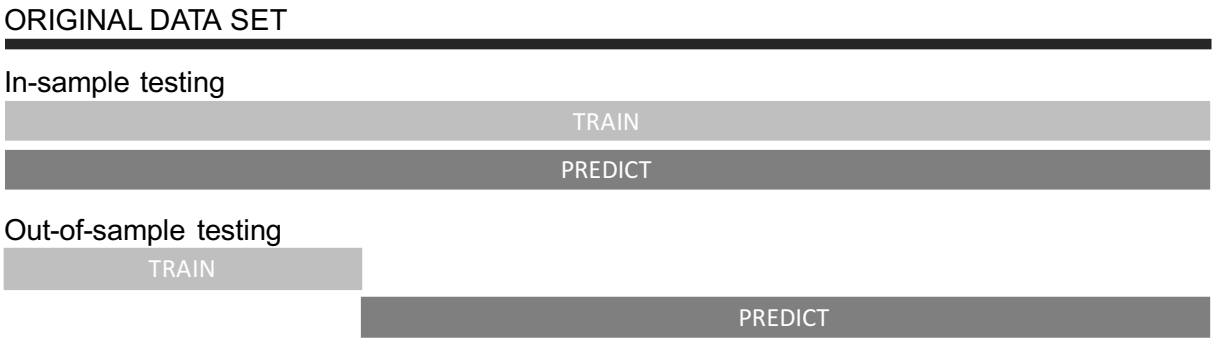

Figure 3.2: There are two different test methods for backtesting. In-sample testing implicates that the testing is performed on the same data as it is optimised on. Out-of-sample testing, on the other hand, is when the optimisation is done on one part of the data set and the actual testing is conducted on the other part of the data set.

Table 3.5: The data used for backtesting. Ten trading days (Tr days) in the beginning of the data sets are lost when computing the 10-valued moving average (Mov avg). On the eleventh trading day, the training can begin.

Data set	1	2	Tr days
Mov avg	2 Jan 2013-15 Jan 2013	22 Oct 2014-4 Nov 2014	10
Learn	16 Jan 2013-21 Aug 2013	5 Nov 2014-16 Jun 2015	150
Predict	22 Aug 2013-27 Jun 2014	17 Jun 2015-15 Apr 2016	210

Another pitfall relevant to this study was the short-sale constraint as the trading strategy involved shorting of OMXS30. However, since the asset was an index, which normally is very easy to short through different financial instruments, it was assumed that the backtesting results would be somewhat accurate and realistic. The only problem was that a position can not be taken directly in the OMXS30 index, but financial derivatives with OMXS30 as underlying asset can be used instead. Such instruments do not have as good liquidity as the data from the OMXS30 index and the difference in liquidity will therefore be taken into consideration when reviewing the results. However, the effects were assumed to be small.

3.5 Evaluation of Results

After the price prediction tool had been implemented and backtested on historical data, the results had to be evaluated to be able to draw any conclusions.

In order to evaluate the price prediction tool, the ratio (denoted *Ratio*) of right number of predictions was used according to (3.4).

$$\text{Ratio} = \frac{\text{Correct number of predictions}}{\text{Total number of predictions}} \tag{3.4}$$

Since the model HMM consisted of two different hidden states, the probability of choosing the right state was assumed to be 50 %. Thus, a ratio higher than 50 % was considered good. Similar ways to evaluate the performance of a price prediction tool have been used in several previous studies (Idvall and Jonsson, 2008; Kannan et al., 2010).

The performance of the trading algorithm was measured by comparing the rate of return of the trading algorithm with a buy and hold strategy of the OMXS30 for the same test period. Transaction costs were excluded. In order to evaluate any excess return of the algorithm, the relative return at the end of the period (denoted as *End Return*) was calculated, see (3.5).

$$\text{End Return} = \frac{\text{Return at last day of algorithm}}{\text{Return at last day of index}} \tag{3.5}$$

Comparing only the returns between OMXS30 and the algorithm's returns could give miss leading results as the risk of one strategy could have been higher than the other. Therefore, the Sharpe

Ratio was used, making it possible to compare the risk-adjusted returns of the trading strategy with the buy and hold strategy. In the Sharpe Ratio, the bench-marked asset R_b was set as the daily return of OMXS30 and the daily Sharpe ratio was calculated using equation (2.1).

Interesting to note is the fact that a buy and hold strategy of index is used as the benchmarking assets in this study. So the value of the benchmarking assets will therefore go down when index goes down an vice versa. However, since the algorithm has the possibility to short index, it is more likely that the algorithm will generate excess return over index in a bear-market than in a bull market. Imagine a one day ahead prediction, if the algorithm will take long position and index goes up, there will be no excess return over index that day. But if the algorithm takes a short position on a day when index drops, then the algorithm will have a positive daily return and the index a negative one, making the relative excess return over index large. However, it is important to remember that when trading in real-time it is not possible to know if the period is a bear or a bull market. But when evaluating a back-test compared to index, the effect is important to be aware of.

3.6 Statistical Significance of Results

When backtesting an algorithm on a finite sample size, problems regarding the possibility of randomness in the results occur (Chan, 2013). Therefore, the significance of the results was to be tested. The test method used was hypothesis testing which according to Easterby-Smith et al. (2015) is a good method to use when the objective is to draw conclusions of performance that go beyond the tested sample.

The initial step of the hypothesis testing was to state null hypothesis, one for each research question. In order to evaluate the statistical significance of the algorithm as a price prediction tool the following null hypothesis was stated:

Null hypothesis 1: The algorithm does not perform better than random predictions of future price movements.

The first null hypothesis is based on the efficient market hypothesis being true and that future prices can not be predicted based on publicly available information. Hence, in order to reject the null hypothesis, it had to be shown that the developed algorithm could perform non random price predictions.

For the second research question, where the significance of the algorithms possibility to earn risk-adjusted rate of returns were investigated, the following null hypothesis was stated:

Null hypothesis 2: The average return from the algorithm is not better than the average return of the index.

After the two null hypotheses were stated, the work of designing good statistical tests to try the two hypothesis were needed. A 95 % confidence level was stated such that the p-value needed

to be less than 0.05 in order to reject any of the hypotheses. For the first null hypothesis, the statistical test built on the assumption that the index could either go up or down the next day and that the probability of the direction of the movement was 50 %, which was inspired by the test conducted by Idvall and Jonsson (2008) to measure the same thing. Hence, the algorithm had two outcomes with equally large chance to happen, either a correct prediction of the price or a wrong one. This made the outcome binary and a Bernoulli trial were therefore conducted and a p-value for each data set was calculated using the MATLAB function *binocdf*. The p-value for the ratio should be interpreted as the probability that the algorithm performs as good or better hit ratio given that it is a random generator. Hence, the p-value had to be less than the significance level in order to reject the null-hypothesis. However, for a hit ratio less than 50 %, the p-value became irrelevant since it could not give any indications about the probability that the algorithm performed better than random predictions.

For the second hypothesis, Chan (2013) argues that there are different ways to hypothesis test the rate of return of an algorithm and the critical part is to determine the distribution of the daily returns. However, in this study, the distribution was assumed to be a Gaussian distribution and the t-statistics was calculated based on the Sharpe Ratio as mentioned in the theory chapter. After the t-statistics were found, the p-value could be calculated using the MATLAB function *tcdf*. Overall, the p-value should be interpreted as the probability to get at least as good or better average daily return given that the algorithm's return in relation to index is zero. Hence, given a Sharpe Ratio greater than zero, the p-value had to be less than the significance level for the null hypothesis to be rejected. However, for a negative Sharpe Ratio, the p-value become irrelevant since it could not give any indications about the probability that the algorithm performed better than the market.

3.7 Robustness of Algorithm

When developing a trading algorithm, it is essential that it is robust and can perform well in different types of markets (Chan, 2013). Therefore, in order to evaluate the robustness of the trading algorithm and investigate how it would perform over time and different markets, a robustness test was conducted. Furthermore, the robustness test was also used to further investigate if the results from the two main data sets were due to randomness or if the performance was consistent over time.

However, as previously mentioned, intraday data for long series back in time were not available and the data used were therefore end of day closing price of OMXS30 from 30 September 1986 to 21 April 2016. Hence, the performance from the backtest was somewhat unreliable since it was assumed that the entry price every day was the closing price of the same day. The results from one data set in the robustness test should therefore only be compared with another data set from the robustness test and not with the two main data sets used in the backtesting. The data for the robustness test was divided into 20 different data sets, each one including 370 unique trading

days.

3.8 Quality of the Study

3.8.1 Validity

Validity is, according to Bryman and Bell (2011), the degree to which a study measures what it is supposed to measure, which in this case refers to a price prediction tool and the possibility to develop a trading algorithm that earn risk-adjusted excess returns. Thus, in order to evaluate that, historical data was used and the test environment was as close to real time conditions as possible, but as previously stated, good performance of an algorithm on historical data does not necessarily say anything about the performance in the future. Instead this study relies on the assumption that good performance on historical data increases the possibility for good performance in the future in accordance with Chan (2013). However, this affects the test validity of the study negatively as it can not be completely certain that the tests used are measuring what they should measure. Moreover, this is characteristic for all studies that use backtesting to test investment strategies or trading algorithms.

Another thing to reflect upon is the external validity of the study, which according to Bryman and Bell (2011) concerns to what extent the conclusions of the study, can be generalised to a larger context (e.g. different time-frames, other assets and geographical markets etc.). Even though results that are statistical significant implicates high external validity, it does not really indicate that the results from a backtest can be generalised. The statistical testing gives some indications on how the algorithm would have performed on an infinite sample of the same data as it was tested on, but normally, market data tends to shift over time. However, since two cross sectional data sets were used, the validity of the model and possibility to generalise improves significantly in comparison to if only one data sets would have been used.

3.8.2 Reliability

According to Bryman and Bell (2011), reliability concerns the consistency of a measure, which in this study can be interpreted as consistency of results from the method used to test the two research questions. However, the reliability of the study could be considered good to the extent that the backtesting method used in the study will give the similar results if the the same test with the same data is conducted at different times or by different people. The replicability of the study which is closely related to the reliability according to Easterby-Smith et al. (2015) is also considered good as the working process and the different choices made have been documented in the methodology chapter. Furthermore, the data sources used in the study are cited and the data are easy to access, which enables replicability.

3.9 Method Discussion

Since the study had a limited time frame, the decision to only conduct tests on historical data was made. However, as previously stated, it is very difficult to make predictions about an algorithm's future performance based on results from backtests. Thus, in order to make better conclusions of the possibility to make accurate predictions and to earn risk-adjusted excess return, real-time tests would have been preferable.

Another thing that can have miss lead the results between the test conducted and the use of a trading algorithm buying and selling in real-time, is the fact that transaction cost were excluded. The reason for doing so was many, but the most apparent one is that the transaction costs such as brokerage et cetera varies a lot depending on for instance how big, measured in money, the trades are. If you trade with small amounts, the brokerage can be zero percent, while if you trade with a couple of millions that percentage is normally higher. Thus, in a real-time scenario, depending on the trading capital, the returns can differ from the results given in this study.

Another aspect to discuss is that the hit ratio was used to evaluate if the algorithm made good predictions of future prices. This can be questioned since there can be scenarios where an algorithm, that have a hit ratio below 50 %, generates high excess return. Consider a case where the hit ratio is under 50 %, but the algorithm manage to predict the large movements in the market while failing to capture the small ones. Such algorithm would generate high returns, even though the hit ratio is bad. Therefore, the validity of the hit ratio as a measure of good price predictions could be questioned. However, Kannan et al. (2010) argues that an algorithm with a good hit ratio is a useful tool to use in a trading algorithm. Together with the fact that other studies (Idvall and Jonsson, 2008; Kannan et al., 2010; Cedervall, 2012) have used the hit ratio from an algorithm as the measure of how good predictions it performs, the choice seemed reasonable.

Furthermore, the use of two test periods consisting of only 210 trading days could also be discussed as the length of the test period directly affects the results' p-value. Hence, in order to get significant results over 210 trading days, the performance of the algorithm had to be very good. Although a longer test period would have given the opportunity to get more significant results, a trade-off arose between the use of one longer test period or two shorter ones due to the lack of access to intraday data. However, in order to make any generalisations of the results it seemed more interesting to study the algorithm's performance based on two cross-sectional data sets representing a bear and a bull market. Moreover, several other studies have also had the same trade-off (Zhang, 2004; Lajos, 2011) and used several shorter test periods, with similar lengths to the one used in this study.

Moreover, in the study, only one trading strategy was used. In theory, infinitely amounts of different trading strategies could be used based on a good price prediction tool, such as the long only or short only strategies used by Lajos (2011). However, since study was under time constraint, only one trading strategy was used.

The hidden states in the HMM was chosen to reflect the price movement in index until 12.00 the upcoming day. For the model being a true HMM, the hidden process should satisfy the Markov property. The two defined states were never tried for satisfying the Markov property, which can be questioned. The study of the hidden sequence representing a Markov process was not within the scope of this report; hence, it was never performed. In defence, many studies before this one have chosen the hidden states in a similar way without testing the Markov property. For instance, Hassan and Nath (2005) define the hidden states to represent the next day's closing price and Zhang (2004) has defined an HMM with the five possible hidden states being *{big drop, small drop, no change, small rise, big rise}*.

When developing the model, there were issues, described in section 2.5.5 *Issues with Hidden Markov Models*. For instance, when investigating a proper learning length L, the transition matrix did not always satisfy the condition explained in (2.10), saying that each row much sum up to one. This was due to convergence issues in the Baum-Welch algorithm. Thus, it was important to understand the origin of the non-convergence issue, which in this case emerged from a too small L.

Furthermore, the transition matrices did not become fully ergodic in some of the models. Instead, there was a transition probability of 100 % going from state two to state two, while the probabilities of going from the first state to any of the two states were separated from zero. The different observable and hidden states were chosen as to minimise the cases of a non-ergodic transition matrix. However, the issue still arose in some of the models resulting in HMMs that preferred state two before state one.

These non-ergodic matrices were found by the Baum-Welch algorithm, which mathematically has been proven to find a local maximum for the parameters. Thus, one idea of preventing non-ergodic matrices is to manipulate them after the Baum-Welch algorithm. Indications during the backtests showed that this could improve the results. However, it was chosen not to manipulate the matrices in this study since there is a lack of scientific studies supporting this method.

Chapter 4 Results

This chapter presents the results generated by the algorithm described in the previous section 3.3 The Algorithm. *The first section contains results of forecasting OMXS30 with the static model. The next section presents results from the dynamic model. Lastly, the robustness test from the two developed models is given.*

4.1 Choice of Parameters

The two parameters Δ and length of learning data L were investigated concerning the ratio of correct predictions and the return. After analysing the results, the parameters were chosen accordingly:

- $\Delta = 2$ SEK
- $L = 150$ trading days

A more thorough discussion on how the parameters were chosen can be found in appendix A *Results when Choosing the Parameters.*

4.2 Static Training Algorithm

In this section, the results from backtesting the static training algorithm are presented. The backtesting was performed on two different data sets of OMXS30.

4.2.1 Data set 1

In the upper plot in Figure 4.1, the accumulation of an array containing correct (1) and wrong (-1) predictions is presented. The number of correct predictions is clearly higher than the number of wrong ones, giving the graph a positive trend. Furthermore, the time development of the capital is presented in the lower graph of Figure 4.1. The capital begins to grow and make a better return than the benchmarked index. However the excess return is not that substantial and diminishes with the development in time.

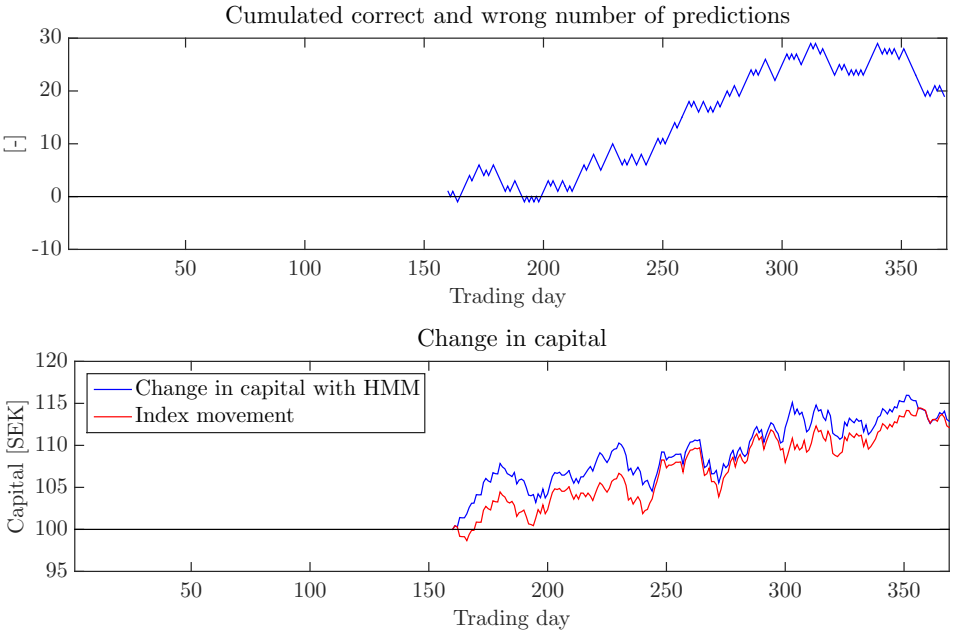

Figure 4.1: In the upper plot, the accumulation of correct predictions (1) and wrong predictions (-1) is presented. In the lower plot, the development of capital is presented. The data stretches from 2 January 2013 to 27 June 2014.

4.2.2 Data set 2

In the upper plot in Figure 4.2, the accumulation of correct (1) and wrong (-1) predictions is provided. The number of wrong predictions is higher than the number of correct ones. Furthermore, the time development of the capital is presented in the lower graph of the Figure 4.2. Again, the HMM-algorithm give rise to an excess return at the beginning of the period but drops heavily around trading day 300 due to multiple wrong predictions.

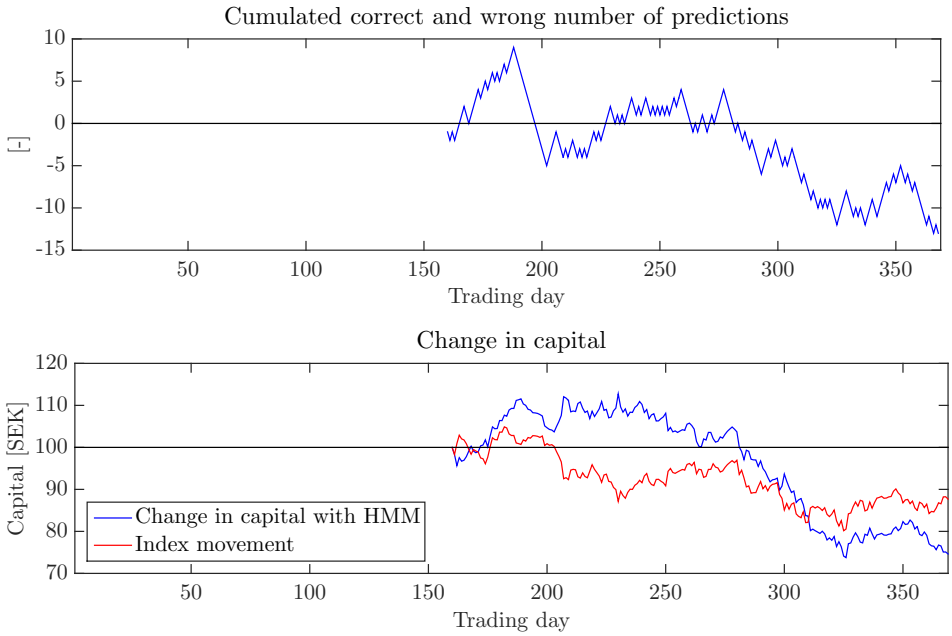

Figure 4.2: In the upper plot, the accumulation of correct predictions (1) and wrong predictions (-1) is presented. In the lower graph, the development of capital during the prediction time is provided. The data stretches from 22 October 2014 to 15 April 2016.

4.2.3 Summary

In Table 4.1, data from the backtests on data set one and two are presented. The column containing 'Ratio' gives the ratio of the correct number of predictions. There is a greater hit rate on the first data set than on the second one. The next column presents the p-value to evaluate the null hypothesis of the model performing random predictions. A small p-value implies that the model does not generate random predictions.

Furthermore, the relative end return is also presented. To evaluate for any risk-adjusted excess rates of return, the Sharpe Ratio was computed. The results are also presented in Table 4.1 and show that a small excess return is given for data set one while there is a negative risk-adjusted excess return for data set two. The next column contains the p-value generated from the Sharpe Ratio.

The final column, to the right, contains information about the overall market behaviour, namely how much index has moved during the prediction period. As previously mentioned, the first data set reflects a bull market while the second data set reflects a bear market.

Table 4.1: Summary of the static training algorithm.

Data set	**Ratio [%]**	**p-value**	**End Return [%]**	**Sharpe Ratio [%]**	**p-value**	**Index [%]**
1	54.07	0.1065	3.02	4.41	0.2616	12.0
2	48.33	0.6609	-12.91	-3.41	0.6891	-11.0

4.3 Dynamic Training Algorithm

Results from the dynamic training algorithm, tested on two different data sets, are presented in this section. The prediction length l was set to 1 trading day. The parameters, Δ and learning length L was chosen to be the same as in the static training algorithm, namely 2 SEK and 150 trading days, respectively.

4.3.1 Data set 1

The upper plot in Figure 4.3 shows the accumulation of correct (1) and wrong (-1) predictions for the first data set. There is a lot of incorrect predictions just before trading day 200 but after the drop, correct predictions are causing a positive trend, making the ratio of correct predictions be greater than 50 %. But even so, the algorithm does not outperform index in terms of development of capital. The evolution of capital is presented in the lower graph of Figure 4.3.

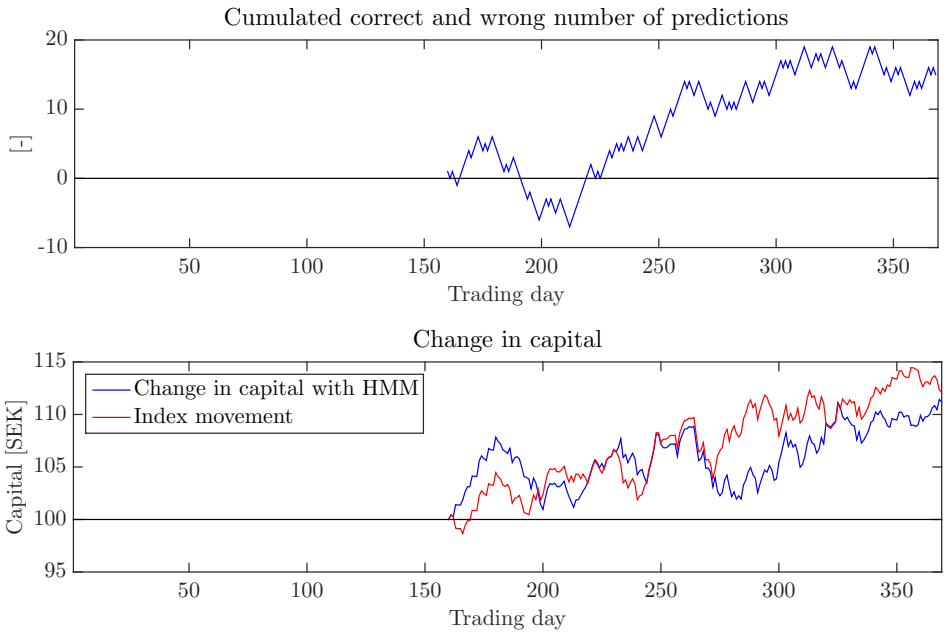

Figure 4.3: In the upper plot, the accumulation of correct predictions (1) and wrong predictions (-1) is presented. The lower plot shows the development of capital during the prediction period. The data stretches from 2 January 2013 to 27 June 2014.

4.3.2 Data set 2

Figure 4.4, shows the accumulation of correct (1) and wrong (-1) predictions along with the development of capital for the dynamic training algorithm applied on the second data set. There ratio of correct predictions in relation to the total number of predictions is greater than 50 %. The algorithm outperforms the index and a great excess return is obtained in the end of the period.

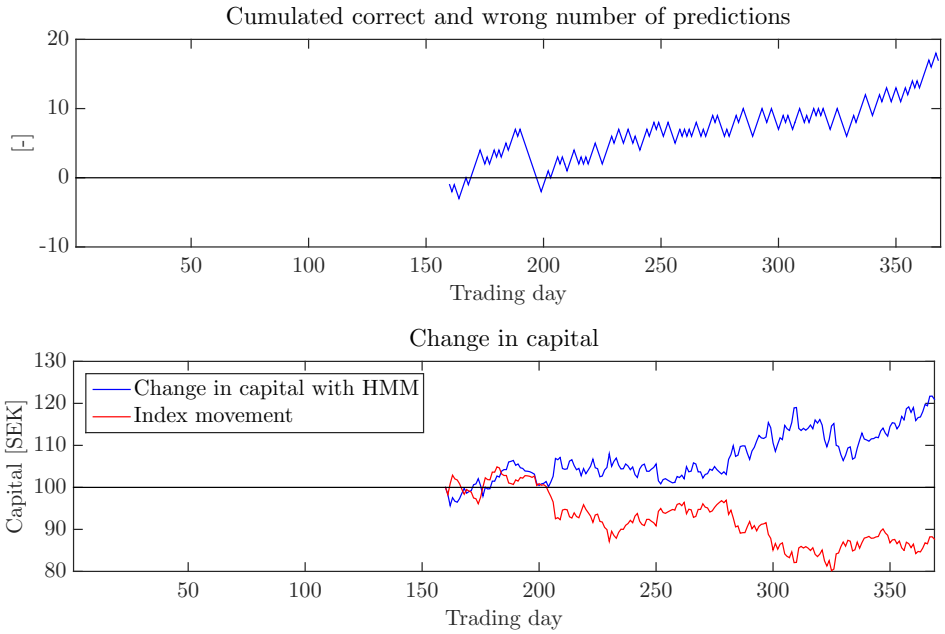

Figure 4.4: In the upper plot, the accumulation of correct predictions (1) and wrong predictions (-1) is presented. The development of capital during the prediction time is shown in the lower plot. The data stretches from 22 October 2014 to 15 April 2016.

4.3.3 Summary

In Table 4.2, results from the backtests on data set one and two are presented. The ratio of correct number of predictions is greater than 50 % for both of the two data sets. The next column presents the p-value to evaluate null hypothesis one saying that the model performs random predictions. The backtest on data set two shows a relatively small p-value however the p-value for the first data set is 29 %.

Furthermore, the relative end return along with the relative average daily return is also presented. Data for the first data set show under-performance of the algorithm. However, as previously stated, the dynamic training algorithm result in high excess return. To evaluate for any risk-adjusted excess rates of return, the Sharpe Ratio was computed. The results are also shown in Table 4.2 and conclude that no risk-adjusted excess return is given for data set one while there is a positive risk-adjusted excess return for data set two. Finally, the last column contains the p-value generated from the Sharpe Ratio. Once again, the p-values are high.

Table 4.2: Ratio of correct number of predictions for each individual data set.

Data set	**Ratio [%]**	**p-value**	**End Return [%]**	**Sharpe Ratio [%]**	**p-value**	**Index [%]**
1	51.67	0.2901	-0.39	-0.31	0.5179	12.0
2	55.02	0.0639	52.02	8.17	0.1188	-11.0

4.4 Robustness

A robustness test was conducted in order to get an impression on how the algorithm would perform over time and on different markets. The two models were tested on 20 different data sets from OMXS30, between 1986 to 2016.

4.4.1 Static Training Model

Results of the robustness test on the static training algorithm is shown in Table 4.3. Studying the column with ratios, 11 of the 20 data sets show ratios less than 50 %. Examining the relative end returns and the Sharpe Ratios, there are big variations between the data sets.

Table 4.3: Results from robustness tests using the static training algorithm. The 18th data set shows zero excess return since the algorithm takes a long position each time making the algorithm's returns equal the index's returns.

Data set	**Ratio [%]**	**p-value**	**End Return [%]**	**Sharpe Ratio [%]**	**p-value**	**Index [%]**
1	52.15	0.2446	27.86	3.22	0.3208	-1.0
2	46.89	0.7967	-40.60	-23.58	0.9996	45.0
3	60.77	0.0007	124.18	13.43	0.0265	-2.0
4	54.07	0.1065	47.69	8.22	0.1175	-21.0
5	49.28	0.5550	-36.61	-12.82	0.9677	53.0
6	45.93	0.8658	-34.32	-12.47	0.9639	21.0
7	49.28	0.5550	-26.67	-12.42	0.9633	34.0
8	57.42	0.0133	5.53	1.02	0.4415	20.0
9	48.33	0.6609	-61.03	-18.39	0.9958	69.0
10	46.41	0.8336	13.37	1.99	0.3868	-37.0
11	55.50	0.0483	193.61	13.28	0.0278	-37.0
12	46.41	0.8336	-29.75	-8.85	0.8995	19.0
13	54.07	0.1065	-13.90	-11.02	0.9441	23.0
14	54.07	0.1065	-6.70	-5.66	0.7934	34.0
15	50.72	0.3911	34.87	3.14	0.3250	-32.0
16	54.07	0.1065	9.96	3.64	0.2991	7.0
17	54.07	0.1065	90.94	12.51	0.0357	-17.0
18	54.07	0.1065	0	0	NaN	15.0
19	44.02	0.9517	-25.64	-10.88	0.9417	4.0
20	48.33	0.6609	2.54	0.78	0.4553	-13.0
Mean	51.29	0.4025	13.77	-2.74	NaN	9.2

The 18th data set show zero relative end return but the last column contain NaN (Not a Number). This is due to the fact that the model predicts a hidden state two at each time point, making the algorithm take a long and hold position throughout the period. This results in no difference between the returns from the algorithm and index.

4.4.2 Dynamic Training Model

Table 4.4 show results of the robustness test performed on the dynamic training algorithm. 15 out of 20 data sets give ratios greater than 50 %. Examining the relative end returns and the Sharpe Ratios, there are big variations between the data sets.

Table 4.4: Results from robustness tests using the dynamic training algorithm.

Data set	**Ratio [%]**	**p-value**	**End Return [%]**	**Sharpe Ratio [%]**	**p-value**	**Index [%]**
1	56.94	0.0189	57.42	6.02	0.1921	-1.0
2	53.59	0.1342	-31.74	-19.66	0.9976	-45.0
3	55.98	0.0359	42.93	5.72	0.2040	-2.0
4	50.72	0.3911	31.74	6.33	0.1801	-21.0
5	51.20	0.3391	-26.60	-12.27	0.9616	53.0
6	49.76	0.5000	-12.26	-6.51	0.8266	21.0
7	44.02	0.9517	-34.51	-17.82	0.9948	34.0
8	48.80	0.6089	-29.49	-6.81	0.8375	20.0
9	54.07	0.1065	-22.97	-9.82	0.9219	69.0
10	53.11	0.1664	44.32	4.42	0.2612	-37.0
11	55.02	0.0639	144.41	10.08	0.0727	-37.0
12	49.28	0.5550	-18.98	-6.71	0.8341	19.0
13	57.42	0.0133	5.43	3.98	0.2826	23.0
14	58.37	0.0063	-2.85	-1.86	0.6061	34.0
15	45.93	0.8658	-45.16	-8.03	0.8771	-32.0
16	53.59	0.1342	7.98	4.39	0.2627	7.0
17	50.24	0.4450	-10.59	-2.46	0.6392	-17.0
18	51.20	0.3391	-16.99	-9.49	0.9148	15.0
19	52.15	0.2446	-0.71	-0.38	0.5217	4.0
20	50.72	0.3911	5.10	1.36	0.4222	-13.0
Mean	52.11	0.3155	4.32	-2.98	0.5905	9.0

Chapter 5 Discussion

The following chapter aims to discuss the results of the study.

5.1 Static Training Algorithm

For the static algorithm, the hit ratio results are inconsistent between the two data sets. The algorithm performs well on the first data set, with a hit ratio of 54.1 %, but worse on the second data set, with a hit ratio of only 48.3 %. Thus, the results indicate that the algorithm does not consistently make better predictions than a random generator.

The good hit ratio for the first data set results in a 3.0 % better end return than index and the daily Sharpe Ratio is as high as 4.4 %, indicating that the algorithm performs better risk-adjusted returns than index. For the second data set the number is significantly lower with an end return of 12.9 % worse than index and a Sharpe Ratio of -3.4. However, the algorithm does not perform badly throughout the period but only in the end of the prediction period, which can be seen in Figure 4.2. Thus, by comparing the results from the static algorithm for the two data sets, it can be concluded that the algorithm is not constantly out-performing the index, especially not in a bear market which had been desirable. Instead, the returns are rather volatile and differ between the periods. The robustness of the algorithm can thus be questioned.

5.2 Dynamic Training Algorithm

For the dynamic algorithm the results from the two data sets are more consistent than the ones for the static algorithm. The hit ratios are 51.7 % and 55.0 % for the first and second data sets, which is considered positive since both hit ratios are above 50 %.

The implementation of the dynamic algorithm in the trading algorithm is also proven to give more consistent returns than the static algorithm. For the first data set, the end return is -0.4 % worse than index, with a daily Sharpe Ratio of -0.3. For the second data set the return is substantially better than index and while index drops about 11 % during the period, the algorithm have an end rate of return of 52 % better than index. Hence, the trading algorithm out-performs index and the daily Sharpe Ratio of 8.2 indicates that the the risk-adjusted return are substantially better as well.

Overall, the results of the dynamic algorithm seem promising as it can out-perform index in a bear market. For the bull market on the other hand, the algorithm makes a minor drop compared to

index and according to Chan (2009), a trading algorithm should ideally produce stable results over time and during different markets. However, it is important to be aware of that the first data set is a bull market making it more difficult for the algorithm to beat index according to the reasoning in section 3.5 *Evaluation of Results*. Therefore, the results from the dynamic algorithm is considered good and indicates that the dynamic algorithm can be useful in order to gain risk-adjusted return over time.

5.3 The Null Hypotheses

Before making any conclusions about the results discussed earlier, it is important to notice that the results can be subject of randomness since the testing were done on two finite sample sizes. Hence, as stated in section 3.6 *Statistical Significance of Results*, two null hypothesis were formulated in order to evaluate the statistical significance of the results.

Starting off with the first hypothesis regarding the price prediction tool being a random generator, it can be concluded that neither the static nor the dynamic algorithm can be considered making better than random predictions at a 95 % confidence level. However, the dynamic algorithm is close for the second data set with a p-value of 0.0639, but it is not good enough to reject the null hypothesis. It is important to understand that the calculated p-value does not say anything about the model being merely a random generator of price movements. Instead it is only used to test the null hypothesis in order to prove that it is not a random generator. Thus, a high p-value does not imply that the model is a random generator, but rather that it can not be rejected as not being a random generator.

The second null hypothesis says that the average return from the algorithm is not better than the average return of the market. However, in order to reject the null hypothesis, a p-value of 0.05 or less had to be achieved, given a Sharpe Ratio greater than zero. As can be seen, none of the algorithms have a p-value lower than 0.05 and the second null hypothesis can not be rejected for any of the algorithms over the two data sets. Thus, it can not be statistical concluded that the two algorithms have performed better than index during the tested periods. However, once again, a p-value above 0.05 does not imply that the average return of the algorithm in relation to index is zero for an infinite sample of the data similar to the test periods, but rather that it can not be rejected as not true.

5.4 Robustness of the Algorithm

For a trading algorithm to be successful, it is, as previously mentioned, important that it is robust and performs well over time and in different markets. When analysing the results from the robustness test, there are few signs of robustness. For instance, in 11 out of 20 data sets, the static training algorithm performs less than 50 % correct predictions. This is a sign of great

instability in the algorithms and the p-values are too high to reject any of the two null hypothesis for the tested time periods.

The dynamic algorithm does seem to reflect somewhat better results than the static one. In 15 out of 20 data sets, the dynamic algorithm predicted the right price movement more that 50 % of the times. However, there is still a low statistical significance reflected by the high p-values. Hence, none of the two stated null hypotheses can be rejected for the dynamic algorithm.

Given that the dynamic algorithm performed better, in terms of ratio of correct predictions, in relation to the static algorithm, one could think that the dynamic algorithm would show better result in terms of return as well. However, looking at the sign of the returns as well as the mean value from the 20 backtests, the static algorithm seems to perform better. In 10 out of 20 data sets the relative end returns and Sharpe Ratios are greater than zero for the static algorithm. However, for the dynamic algorithm, the number of relative end returns and Sharpe Ratios greater than zero are solely eight.

To analyse if the algorithms would perform better on a specific market, such as a bull or bear market, the correlation between market and the end return along with the Sharpe Ratio was briefly analysed. It seems as if the static algorithm performed better in a bear market than in a bull market. In all eight of the data sets representing a bear market, the end return and Sharpe Ratio was positive. On the other hand, in the 12 data sets representing a bull market, 11 resulted in a return and Sharpe Ratio less or equal to zero. However, this can somewhat be explained by the reasoning in section 3.5 *Evaluation of Results*, saying that it is easier to make excess return compared to index in a bear market than a bull market.

Conducting a similar analysis for the dynamic algorithm, no convincing pattern can be found between a specific market and the performance of the model. Out of eight positive end returns in relation to index, six were performed in a bear market while two were performed in a bull market. Out of twelve negative end returns, nine were performed in a bull market while three were performed in a bear market.

5.5 Overview of the Algorithms' Performance

The results of the study indicate that both the static and the dynamic HMM trading algorithm can earn risk-adjusted excess return during limited time frames. However, it can not be statistically proven that the two developed algorithms can out-perform the market and yield excess risk-adjusted rates of returns.

The static algorithm reflected an inconsistency in the two original backtests. However, in the robustness test, there was a hint of correlation between a bear market and excess return. However, bad performance in bull markets in the robustness test show low probability of consistently making risk-adjusted excess return on real-time data.

The results from the dynamic algorithm tested on the two data sets give indications of usefulness in order to earn risk-adjusted excess return seen over a time period with different markets. However, results of the robustness test indicate that the algorithm may be too unstable to perform well over longer time frames.

5.6 Hidden Markov Models in Trading Algorithms

The results indicate that the developed algorithms are somewhat unstable and unreliable. However, HMMs in general should not be ruled out as good predictors of future price movements in the financial market as it is possible that a more complex HMM can perform good predictions. An example of a more complicated and complex type of HMM, often used in finance, is one where the observable states are assumed to be emitted according to some probability distribution.

Moreover, the states can be modeled as to emitting observations according to a mixture of Gaussian distributions. For instance, Hassan and Nath (2005) predicted next day's closing price using a three-dimensional Gaussian mixture model with good results. Another approach, taken by Nguyen and Nguyen (2015), is the use of an HMM to forecast four macroeconomic measures as guidance for stock selection, indicating that more factors than just historical data may be needed in order to successfully use HMM as a tool for generating good predictions of financial assets. Thus, the results from this report should not dismiss the usage of HMMs in finance. The results should instead be interpreted as to understanding that a more complex model might be needed in order to reflect behaviours in the financial market.

Furthermore, it is interesting to reflect on how a manipulation of non-ergodic models would have influenced the results. As previously stated, in section 3.9 *Method Discussion*, there were some indications that such an action could improve the HMM and thus the results. One such indication is the case with the 18th data set in the robustness test of the static algorithm. For each new trading day, the model predicts the second state due to a non-ergodic transition matrix.

Another aspect to take into consideration is that the HMM is designed to find patterns in the historical data, for which the model is trained. The HMM is then able to predict future data given that the patterns will repeat themselves in the future. However, if the efficient market hypothesis is true for the studied time period, no such patterns are present, making the use of an HMM obsolete.

Chapter 6 Conclusions

The purpose of the study was to evaluate the use of Hidden Markov Models as a tool in algorithmic trading in the Swedish market. To achieve that and answer the two research questions, two different algorithms based on a Hidden Markov Model were developed. One algorithm used static training while the other one used dynamic training to find a good model fit. The algorithms were then backtested on OMXS30 historical data, ranging from 22 Aug 2013 to 27 Jun 2014 and 17 Jun 2015 to 15 Apr 2016. Furthermore, a robustness test was conducted on OMXS30 historical data ranging from 30 Sep 1986 to 21 Apr 2016.

The first research question dealt with the algorithm based on the Hidden Markov Model being able to make good predictions of future prices of the OMXS30 index. From the results, it could be concluded that neither the static nor the dynamic models could perform significantly better predictions of the price movements of OMXS30 than a random generator would do.

The second research question aimed to investigate if an algorithm based on a Hidden Markov Model can earn risk-adjusted excess rates of return. In the study it could not be statistically proven that the two developed algorithms can out-perform the market and yield excess risk-adjusted rates of returns. Nevertheless, the use of an HMM should not be ruled out since the results indicates that both the dynamic and the static model can be useful in order to earn risk-adjusted excess return. However, further studies need to be conducted in order to evaluate that.

In the study, only two simple models of the HMM was used, which both gave indications of being useful in order to earn risk-adjusted excess return and it is therefore proposed to further evaluate the use of such algorithms. Furthermore, algorithms based on the HMM can be extremely complex. Hence, to further investigate the use of HMMs as a tool for building a profitable trading algorithm, it is proposed to study how more advanced and complex HMM algorithms would have performed on OMXS30. For instance, it would be interesting to extend the model into emitting observations according to a specific probability distribution. Furthermore, it could also be of interest to include macroeconomic variables to define the observable states. Interesting macroeconomic variables include inflation and exchange rate, which are two factors that are identified to influence the stock market prices of the Stockholm stock exchange according to Talla (2013).

Furthermore, it would also be interesting to study how a manipulation of non-ergodic models could effect the results of a trading algorithm based on HMM. Indications from this study show that such manipulation could improve the predictions of price movements.

Bibliography

Axelson-Fisk, M. (2010). *Comparative Gene Finding - Models, Algorithms and Implementation*, volume 11 of *Computational Biology*. Springer, London.

Bailey, D. H., Borwein, J. M., de Prado, M. L., and Zhu, Q. J. (2014). Pseudomathematics and financial charlatanism: The effects of backtest over fitting on out-of-sample performance. *Notices of the AMS*, 61(5):458–471.

Balvers, R. J., Cosimano, T. F., and McDonald, B. (1990). Predicting stock returns in an efficient market. *The Journal of Finance*, 45(4):1109–1128.

Bhar, R., Hamori, S., and (e-book collection), S. A. (2004). *Hidden Markov models: applications to financial economics*, volume 40. Kluwer Academic Publishers, Boston, Mass.

Bookstaber, R. M. (2007). *A demon of our own design: markets, hedge funds, and the perils of financial innovation*. J. Wiley, Hoboken, N.J, 1 edition.

Bryman, A. and Bell, E. (2011). *Business research methods*. Oxford University Press, Oxford, 3 edition.

Cedervall, F. (2012). Machine learning for technical stock analysis. Master's thesis, KTH Royal Institute of Technology.

Chan, E. P. (2009). *Quantitative trading: how to build your own algorithmic trading business*. John Wiley & Sons, Hoboken, New Jersey.

Chan, E. P. (2013). *Algorithmic trading: winning strategies and their rationale*. John Wiley & Sons, Inc, Hoboken, New Jersey, 1 edition.

Davey, K. J. (2014). *Building winning algorithmic trading systems: a trader's journey from data mining to Monte Carlo simulation to live trading*. Wiley, New Jersey.

Easterby-Smith, M., Thorpe, R., and Jackson, P. (2015). *Management and business research*. SAGE, London, 5th edition.

Eling, M. and Schuhmacher, F. (2007). Does the choice of performance measure influence the evaluation of hedge funds? *Journal of Banking & Finance*, 31(9):2632–2647.

Enke, D. and Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems With Applications*, 29(4):927–940.

Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2):383–417.

Ferson, W. E. (1989). Changes in expected security returns, risk, and the level of interest rates. *The Journal of Finance*, 44(5):1191–1217.

Finansinspektionen (2012). Kartläggning av högfrekvenshandel och algoritmhandel. `http://www.fi.se/upload/43_Utredningar/20_Rapporter/2012/hogfrekevens4.pdf`. Accessed: 2016-04-18.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Furse, C., Haldane, A., Goodhart, C., Cliff, D., Zigrand, J., Houstoun, K., Linton, O., and Bond, P. (2011). The future of computer trading in financial markets. Technical report, Final project report, Foresight, Government Office for Science, London (UK).

Granger, C. W. J. and Newbold, P. (1986). *Forecasting economic time series*. Academic Press.

Hasbrouck, J., Sofianos, G., and Sosebee, D. (1993). New york stock exchange systems and trading procedures.

Hassan, M. R. and Nath, B. (2005). Stock market forecasting using hidden markov model: a new approach. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 192–196. IEEE.

Hendershott, T. (2003). Electronic trading in financial markets. *IT Professional*, 5(4):10–14.

Hendershott, T. and Riordan, R. (2009). Algorithmic trading and information. *Manuscript, University of California, Berkeley*.

Idvall, P. and Jonsson, C. (2008). Algorithmic trading – hidden markov models on foreign exchange data. Master's thesis, Linköping university.

Isacsson, T. (2011). Datorer härskare over börshandeln. `http://www.svd.se/datorer-harskare-over-borshandeln`. Accessed: 2016-04-08.

Kannan, K. S., Sekar, P. S., Sathik, M. M., and Arumugam, P. (2010). Financial stock market forecast using data mining techniques. In *Proceedings of the International Multiconference of Engineers and computer scientists*, volume 1, page 4.

Kavitha, G., Udhayakumar, A., and Nagarajan, D. (2013). Stock market trend analysis using hidden markov models. *International Journal of Computer Science and Information Security*, 11(10):103.

Lajos, J. (2011). *Computer modeling using hidden Markov model approach applied to the financial markets*. PhD thesis, Oklahoma State University.

Lin, T. C. W. (2014). The new financial industry. *65 Alabama Law Review 567 (2014)*.

Lo, A. W. (2002). The statistics of sharpe ratios. *Financial Analysts Journal*, 58(4):36–52.

Lo, A. W., Mamaysky, H., and Wang, J. (2000). Foundations of technical analysis: Computa-

tional algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4):1705–1765.

Lopez de Prado, M. M. and Peijan, A. (2004). Measuring loss potential of hedge fund strategies. *The Journal of Alternative Investments*, 7(1):7–31.

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82.

Mamon, R. S. and Elliott, R. J. (2007). *Hidden Markov models in finance*, volume 104. Springer, New York.

Marsland, S. (2015). *Machine learning: an algorithmic perspective*. CRC press.

Nasdaq (2016). Omx stockholm 30 index. `http://www.nasdaqomxnordic.com/index/historiska_kurser?Instrument=SE0000337842`. Accessed: 2016-05-20.

News, A. B. (2016). Algorithm trading – what is algorithmic trading? `http://www.advisoryhq.com/articles/algorithmic-trading/`. Accessed: 2016-05-08.

Nguyen, N. and Nguyen, D. (2015). Hidden markov model for stock selection. *Risks*, 3(4):455–473.

Ni, J. and Zhang, C. (2005). *An Efficient Implementation of the Backtesting of Trading Strategies*, volume 3758, pages 126–131. Springer Berlin Heidelberg, Berlin, Heidelberg.

Nuti, G., Mirghaemi, M., Treleaven, P., and Yingsaeree, C. (2011). Algorithmic trading. *Computer*, 44(11):61–69.

Olsson, H. and Sörensen, S. (2011). *Forskningsprocessen: kvalitativa och kvantitativa perspektiv*. Liber, Stockholm, 3 edition.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications inspeech recognition. 77:257–286.

Rabiner, L. R. and Juang, B.-H. (1986). An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of business*, 39(1):119–138.

Sharpe, W. F. (1994). The sharpe ratio. *The journal of portfolio management*, 21(1):49–58.

Shiller, R. J. (2000). *Irrational exuberance*. Princeton Univ. Press, Princeton, N.J.

Sullivan, R., Timmermann, A., and White, H. (1999). Data-snooping, technical trading rule performance, and the bootstrap. *The journal of Finance*, 54(5):1647–1691.

SVD (2015). Fingerprint petar mtg från börsens finrum. `http://www.svd.se/fingerprint-petar-mtg-fran-borsens-finrum`. Accessed: 2016-05-20.

Talla, J. T. (2013). Impact of macroeconomic variables on the stock market prices of the stockholm

stock exchange (omxs30). Master's thesis, Jönköping University, JIBS, Economics, Finance and Statistics.

Thaler, R. H. (1987). Anomalies: The january effect. *Journal of Economic Perspectives*, 1(1):197–201.

TheBonnotGang (2016). 1 minute historical data download. `http://thebonnotgang.com/tbg/historical-data/`. Accessed: 2016-04-18.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Zhang, Y. (2004). *Prediction of financial time series with Hidden Markov Models*. PhD thesis, Simon Fraser University.

Chapter A Results when Choosing the Parameters

Given the algorithm based on static learning, the two parameters Δ and length of learning data L was investigated in relation to the ratio of correct predictions and the mean value of the capital. To avoid any data snooping, the tests were performed on a different set of data of OMXS30 than the actual backtesting.

A.1 Length of Learning Data

In Figure A.1, the results from iterating the algorithm for different lengths L of the training set is displayed.

In the upper plot, the ratio of correct movements are shown. In the middle plot, the mean value of the capital during each run is shown. To find an optimal value of L, we would like to maximise the ratio of correct predictions along with the mean value of the capital. There is no value of L that clearly optimises our algorithm based on the three criteria presented here. L was chosen to 150 trading days in the algorithm.

The fact that there is no clear optimum for the learning length given, the static model, is an indication that the robustness of the model does not depend heavily on this parameter. However, for some values, there is a lowering in performance. For instance, the two graphs show a reduction in the performance for a learning length of 70 trading days.

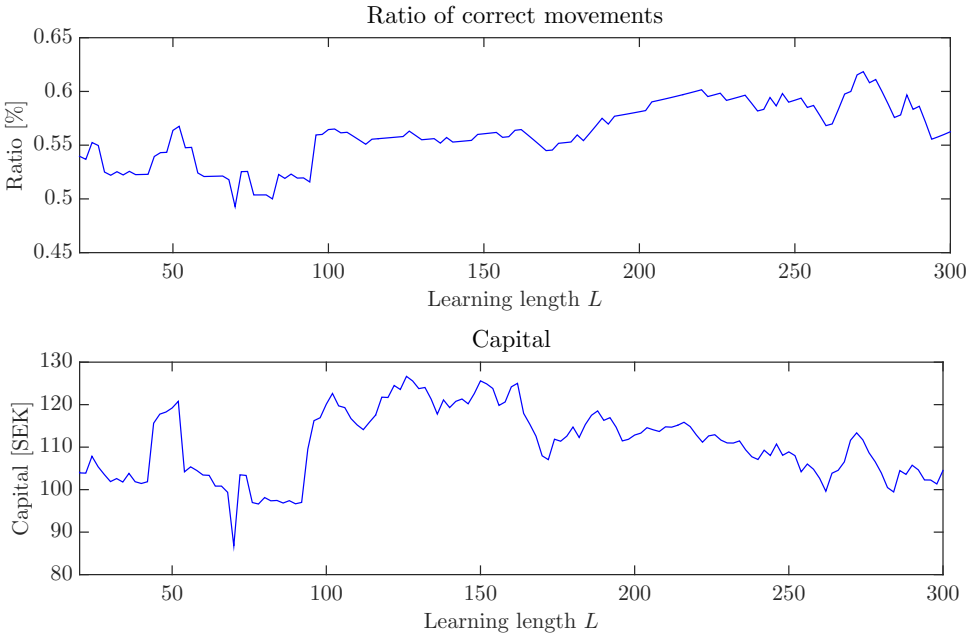

Figure A.1: Results where the algorithm has been iterated for different lengths L of the training set. The upper plot shows the ratio of correct movements in relation to the length of the training set. The lower plot shows the mean value of the capital as a function of L. From these two graphs, the training length was chosen to 150 trading days.

A.2 Delta

Figure A.2 shows the results from iterating the algorithm for different values of Δ. The upper plot shows the ratio of correct predictions and the lower plot shows the mean value of capital during the prediction phase. Once again, it is difficult to identify what value of Δ that will optimise the algorithm, however, a Δ of 2 SEK was chosen for the backtests.

The discussion about robustness in the previous section is applicable for delta as well. However, it seems as if the model is slightly more sensitive towards delta than the learning length. This is reasonable since the observable states are defined by δ. Hence, a too big delta could give rise to only a few observed states for a time series.

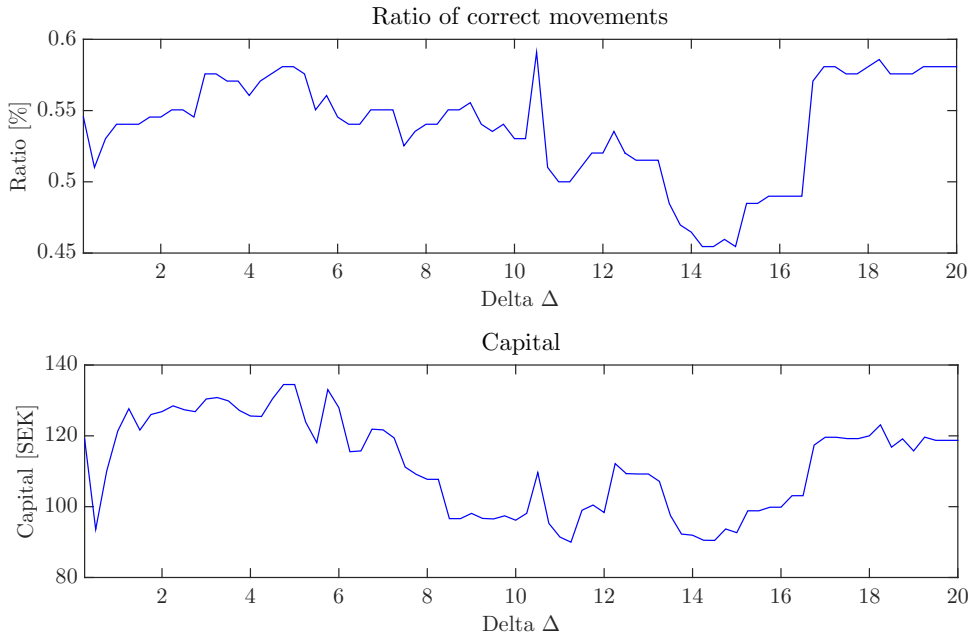

Figure A.2: Results where the algorithm has been iterated for different values of Δ. The upper plot shows the ratio of correct movements in relation to Δ. The lower plot shows the mean value of the capital during the prediction phase as a function of Δ. Δ was chosen to 2 SEK in the algorithm.