REAL-TIME STOCHASTIC PORTFOLIO OPTIMIZATION

Ph.D. dissertation of

SIPOS ISTVÁN RÓBERT, M.SC.

Supervisor:

Dr. Levendovszky János, D.Sc.
Doctor of the Hungarian Academy of Sciences

Department of Networked Systems and Services
Budapest University of Technology and Economics

Budapest, 2015

Acknowledgments

I would like to express my gratitude to my supervisor for his continuous support and encouragement. Special thanks also to dr. Norbert Fogarasi, Attila Ceffer, Farhad Kia and dr. Gábor Jeney for their helpful suggestions related to this work. Finally, I would like to acknowledge the support of my family and friends keeping me motivated along these years.

Summary

In modern finance, algorithmic trading plays an ever increasing role. Presently, of trading transactions performed on an algorithmic basis exceed 50% of total volume, which gains more and more ground at NYSE, LSE, etc. For example, high-frequency trading accounted for 77% of transactions in the U.K. markets in 2010 according to Tabb Group, or in 2008, algorithmic trading had a solid hold at 52% of market order volume at Frankfurt Stock Exchange [Hendershott et al, 2013], similarly, over 50% of all trades in the U.S. equity markets were algorithmic in 2010 [Avellaneda, 2011] implying a multi-trillion USD annual volume. Big financial services companies (such as Morgan Stanley, Goldman Sachs, Credit Suisse... etc.) invest a huge amount money into developing fast and reliable algorithms appropriate for the purpose of High Frequency Trading (HFT). The success and the profitability of these algorithms lie, on the one hand, in the application of sophisticated objective functions which can minimize the risk but may achieve a given profit, and in the speed of their operation, on the other. As a result, there is a great need for developing high speed and novel optimization methods which can strike a good trade off between complexity and running speed to ensure profitable results in the environment of HFT.

In the dissertation new methods are developed to broaden the set of mathematical tools used for algorithmic trading, in the following fields:

- different stochastic models are discussed and developed for portfolio optimization, including identifying mean reverting portfolios, modelling by traditional and autoregressive hidden Markov models (HMM) in order to capture the underlying characteristics of the financial time series;

- various estimation and learning methods are introduced and compared for model parameter identification. Training the HMMs is carried out by (i) Baum-Welch expectation maximization algorithm; (ii) simulated annealing; (iii) and hybrid methods [Sipos et al, 2013b]. To cope with the underlying complexity a probabilistic PCA based dimension reduction method and a novel clustering algorithm are also introduced [Sipos et al, 2013b];

- novel objective functions are developed for trading with mean reverting portfolios in the light of maximizing the achieved profit and minimizing the related transaction costs by optimizing subject to cardinality constraints. Portfolio optimization is then performed according to (i) maximizing the predictability; (ii) maximizing the average return; and (iii) maximizing the return achieved with a predefined probability;

- the selection of the optimal portfolio is carried out by stochastic search algorithms (when no analytical solution is available) including simulated annealing and feedforward neural networks [Sipos et al, 2013a]. The performance of the trading with mean reverting portfolios was further enhanced by generalizing it to a more flexible modelling with autoregressive HMMs. Consequently, novel parameter estimation methods and a corresponding objective function are also introduced [Sipos et al, 2015a]. The presented solutions satisfy the cardinality constraint in terms of providing a sparse portfolios which minimize the transaction costs (and, as a result, maximize the interpretability of the results).

In order to implement these methods for HFT we utilize a massively parallel GPGPU architecture. Both the portfolio optimization and the model identification algorithms are successfully tailored to be running on GPGPU to meet the challenges of efficient software implementation [Sipos et al, 2015b].

The performance has been extensively tested on historical data obtained from S&P 500, FOREX and U.S. SWAP rates. The results demonstrate that a good average return can be achieved by the proposed trading algorithms in realistic scenarios. Furthermore, the results have proven that profitability can also be accomplished in

the presence of secondary effects and the extensive speed profiling has demonstrated that GPGPU is capable of real-time performance in trading, thus it can cope with the requirements of HFT.

Contents

List of Figures

List of Tables

List of Abbreviations

AR-HMM: AutoRegressive Hidden Markov Model

BM: Brownian Motion

BW: Baum-Welch

CPU: Central Processing Unit

EM: Expectation Maximization

FFNN: FeedForward Neural Network

GMM: Gaussian Mixture Model

GPGPU: General-Purpose computing on Graphics Processing Units

GPU: Graphics Processing Unit

HFT: High Frequency Trading

HMM: Hidden Markov Model

MPT: Modern Portfolio Theory

MR: Mean Reversion

OU: Ornstein-Uhlenbeck

PCA: Principal Component Analysis

PDF: Probability Density Function

PPCA: Probabilistic PCA

SA: Simulated Annealing

SDE: Stochastic Differential Equation

VAR(1): Vector AutoRegression

Chapter 1

Introduction and motivations

Portfolio optimization was first investigated by Markowitz in the context of diversification. Investing into a portfolio instead of a single asset allows one to optimize the return and the associated risk according to various objectives. The so-called Modern Portfolio Theory (MPT) dates back to 1950's [Markowitz, 1952] aims at minimizing the associated risk for a given return or maximize the expected return for an accepted level of risk by portfolio selection. The portfolio is described by a vector, the components of which indicates the number of possessed quantity from the different assets. For example, if we trade with the stocks of the following S&P 500 companies: {GOOGL, MA, MS}, then the portfolio vector {20, 10, 40} implies having 20, 10 and 40 shares of Google Inc., Mastercard Inc. and Morgan Stanley, respectively. Portfolio optimization amounts to finding the best linear combination (the best portfolio vector), which satisfies some favourable random properties which makes trading profitable. For example, MPT follows the intuitive idea that the return of the portfolio is the linear combination of the return of the single assets, while the variance of the portfolio is getting lower by the diversification. In turn, the real profitability and the risk of a chosen portfolio does not directly result from the expected returns and variances. At the same time, the applied model must be robust, keep the transaction costs low and feature a scalable computational complexity for practical implementation on different hardware architectures. Hence the dissertation focuses on optimizing the portfolio vector.

However, finding the most profitable objective function and performing the corresponding portfolio optimization is a rather complex and thus challenging job. The dissertations aims at developing new methods for optimal portfolio selection. So far, the spectrum of algorithmic trading approaches includes (Kissell [2013], Chan [2009]):

- simple trend following strategies;
- pair trading;
- mean reversion;
- mathematical model based strategies (e.g. delta-neutral);
- arbitrage opportunities;
- index fund rebalancing;
- scalping...

All of these methods interpret portfolio construction as a mapping from the individual asset prices (as random processes, following a relatively "wild" stochastics) to a portfolio process which may be easier to predict or have reduced risk, according to the improved stochastic behaviour. Whatever way one selects a portfolio there is an evident trade-off between risk and profit according to mathematical finance, as a result there are different methods to strike a good balance between them. One approach is to seek a portfolio which maximizes its predictability, i.e. mean reversion is a good indicator of predictability, as a result, identifying mean reverting portfolios has become a key area of investigation (d'Aspremont [2011], Fogarasi et al [2013a]). Assuming that the asset price vector follows a VAR(1) process, portfolio optimization can be reduced to solving a generalized eigenvalue problem [d'Aspremont, 2011]. Another idea is to find two historically strongly correlated assets (usually from the same sector), and wait until a temporary divergence is observed. At this point, by investing into a portfolio consists of buying the underperformer and short sell the outperformer asset, profit can be realized when the spread tightens [Kissell, 2013]. Alternatively, one can select an optimal portfolio by minimizing its mean-absolute deviance as in

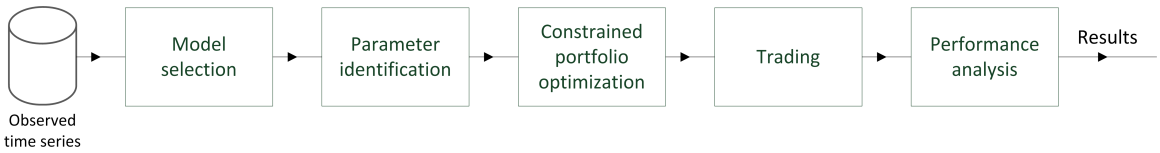

Figure 1-1: Computational approach

Konno [1991]. This approach is further elaborated in section 2. However, in the dissertation, we are not going to follow these approaches, because the mean reversion provides a more general model.

In general, the process of algorithmic trading can be summarized by the following flowchart (Fig. 1-1):

- parameter identification for the given model based on the observed data;
- portfolio optimization subject to an objective function;
- forming trading actions (and evaluate the performance achieved).

According to the general flowchart, the main endeavours of the present dissertation lie in developing:

- models which can better capture the underlying properties of the portfolio price process (e.g. OU, AR-HMM, etc.);
- fast model identification methods to estimate the unknown parameters;
- novel learning and optimization algorithms to support model identification;
- heuristics to cope with NP hard problems of constraints imposed on the cardinality of the portfolio vector,

giving rise to the following treatment.

Model selection

As far as the model selection is concerned, I focus on portfolios selected subject to mean reverting properties or to profit maximization by AR-HMM modelling.

While trading with mean reverting portfolios, we need to make a decision on whether the observed process exhibits mean reverting properties, or follows a Brownian motion or even is driven by mean aversion. Since these properties can change over time, a more flexible time dependent distribution model of financial time series is needed.

As with identifying mean reverting portfolios — and trading by the estimated long term mean of a given portfolio —, more generally, a trader can take advantage of a good quality prediction of the portfolio's future price as well. HMMs are widely used for predictions in fields as diverse as finance (Hassan et al [2005], Mamon et al [2007]), genomics [Durbin et al, 1998], linguistics [Jurafsky et al, 2006], and can better model stochastic properties changing over time. One of its possible generalizations is the autoregressive-HMM, which can describe an even wider range of random processes.

Learning, optimization and algorithmic complexity

As mentioned before, another aspect of algorithmic trading is related to algorithmic complexity. A widespread set of mathematical models and methods supports the needs for real-time and profitable trading. However, fast and profitable trading often proves to be difficult in the case of high number of assets, because the profit is considerably decreased by the transaction costs. On the other hand, to develop and run efficient trading algorithms in real-time requires striking a good balance between algorithmic complexity and runtime speed. Therefore one of the central issues of portfolio optimization is to optimize portfolios subject to cardinality constraint (which specifies the number of non-zero components not to exceed a given threshold) in real-time. This provides sparse portfolios to minimize the transaction costs and to maximize interpretability of the results. d'Aspremont [2011] analyzed the problem of finding mean reverting portfolios with cardinality constraints, however, with this constraint, the optimal portfolio selection becomes NP hard [Natarjan, 1995].

As far as the HMMs are concerned, efficient algorithms have not yet been developed for high speed and quality training of the free parameters, which are appropriate

for the purpose of high-frequency algorithmic trading [Hassan et al, 2007]. In order to optimize the parameters of the HMM, traditionally the Baum-Welch expectation maximization [Baum et al, 1966] algorithm is used. Unfortunately, although it has proven to be a computationally efficient algorithm, it can notoriously get stuck in one of the local optima. Thus, the quality of the achieved optimum is strongly based on the initialization.

Trading and performance analysis

Based on the identified portfolio and the predictions given by the model, the applied trading algorithm triggers the appropriate trading signal. Furthermore, secondary effects are also taken into account (like the bid-ask spread), which adapts the analysis to realistic market scenarios.

In order to evaluate the novel methods, I developed an extensive computational and back-testing framework for detailed performance analysis.

In this light, the main objectives of my dissertation are as follows:

- optimizing mean reverting portfolios by new numerical approaches and subject to novel objective functions (e.g. maximizing the expected mean return instead of predictability);
- choosing an optimal portfolio based on the hidden Markov modelling of the portfolio value time series;
- extending the underlying model to AR-HMM to further maximize the profit by better capture the stochastic behaviour;
- extensive speed profiling of the demonstrated algorithms.

This can be summarized by the following figure (Fig. 1-2).

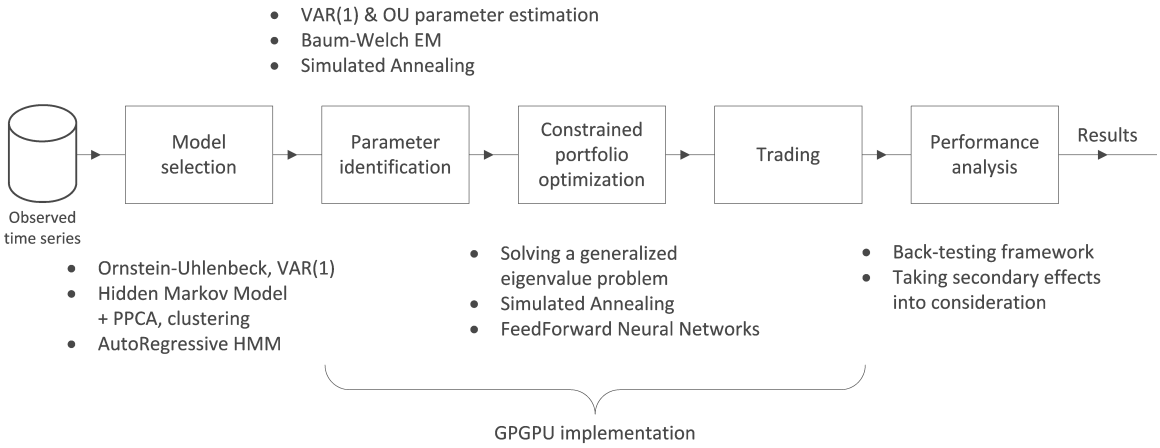

Figure 1-2: Application of the contributions

In each case, the objective is to maximize a given performance evaluation function, which in turn results in the optimal portfolio. Then, the identified portfolio is converted to a trading signal for taking the appropriate trading actions. In the phase of performance analysis, various numerical indicators are evaluated for the sake of comparing the profitability of different methods. Figure 1-2 also demonstrates that the dissertation contributed to each step of algorithmic portfolio optimization and trading.

For the purpose of performance analysis and for the comparison of the different methods detailed in the theses, I have developed numerous models and algorithms in MATLAB®. These algorithms are parts of an extensive simulation framework. The framework can also accommodate real financial time series to obtain back-testing results for verifying profitability.

1.1 Models and methods

In the course of the research I used modelling, analytical and simulation tools summarized by the following table (Table 1.1).

Throughout the dissertation, many stochastic models have been introduced which require different parameter identification methods, as detailed by the following points:

- VAR(1): least squares and maximum likelihood estimations;

Method	Use in dissertation	Sections
Analytical	Maximizing predictability	Section 2.2.1
Modelling	VAR(1) OU HMM AR-HMM	Chapter 2 Chapters 2 and 4 Chapter 3 Chapters 4 and 5
Algorithm	FFNN SA Baum-Welch EM PPCA Clustering	Section 2.3.2 Sections 2.3.1, 3.2, 4 and 5 Sections 3.2, 4 and 5 Section 3.2.1 Section 3.2.1
Simulation	Performance analysis	For each method

Table 1.1: Methods used in the dissertation

- Ornstein-Uhlenbeck SDE: sample mean, linear regression based, recursive, mean squared and pattern matching estimation techniques;
- FFNN: back propagation algorithm;
- HMM: Baum-Welch expectation maximization algorithm, simulated annealing and a novel hybrid training method;
- PPCA: an iterative expectation maximization algorithm;
- Clustering with GMM: an expectation maximization algorithm;
- AR-HMM: Baum-Welch expectation maximization algorithm.

Having identified the parameters of the specific model I applied, one can then optimize the portfolio, which leads to:

- solving a generalized eigenvalue problem;
- simulated annealing;
- HMM based prediction methods.

Chapter 2

Sparse mean reverting portfolios

In this section after defining the model and the underlying notations, I set on optimizing sparse mean reverting portfolios. The time series describing the prices of assets is denoted by $\mathbf{s}_t^T = [s_{1,t}, ..., s_{n,t}]$ where $s_{i,t}$ is the price of asset i at time instant t. The portfolio vector is denoted by $\mathbf{x}^T = [x_1, ..., x_n]$ where x_i gives the number of possessed quantity from asset i. The value of the portfolio at time t is denoted by $p(t)$ and defined as

$$p(t) = \mathbf{x}^T \mathbf{s}_t = \sum_{i=1}^{n} x_i s_{i,t}. \tag{2.1}$$

Our objective is to find the optimal portfolio $\mathbf{x}_{opt}$ which maximizes a pre-defined objective function, such as the average return:

$$\Psi(\mathbf{x}) = \max_{0 \leq t} E(p(t)) - p(0), \tag{2.2}$$

thus $\mathbf{x}_{\text{opt}} = \arg\max_{\mathbf{x}} \Psi(\mathbf{x})$, subject to cardinality constraint which specifies that the number of non-zero components in $\mathbf{x}_{opt}$ must not exceed a given number l.

$$\mathbf{x}_{opt}^T = [0, 0, ..., x_{i_1}, ..., 0, ..., x_{i_2} ..., x_{i_l}, ..., 0] \tag{2.3}$$

The optimal portfolio is sought under the assumption that the portfolio value $p(t)$ exhibits mean reverting properties, i.e. it follows an Ornstein-Uhlenbeck (OU) process [Ornstein et al, 1930]. This is a frequent assumption in trading (Fama et al [1988],

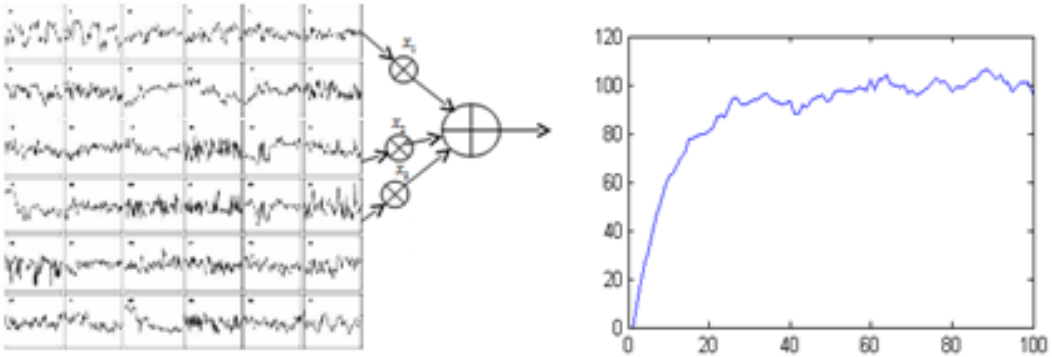

Figure 2-1: Individual assets with bad predictability, and their linear combination following an OU process

Manzan [2007], Ornstein et al [1930], Poterba et al [1988]), as the linear combination of individual assets shows a much better predictability (see Fig. 2-1), and as it is easy to trade with this portfolio in an intuitive way — buying below the long term mean, and selling it once it has reached it.

In practice there are other proposals for optimizing portfolios. For instance, as introduced in Konno [1991], one can minimize the mean-absolute deviance assuming an L_1 model as follows:

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \sqrt{\frac{2}{\pi}} \sigma\left(\mathbf{x}\right) E\left(\left|p\left(t\right) - E\left(p\left(t\right)\right)\right|\right) \tag{2.4}$$

This optimization problem can be solved efficiently by using linear programming approaches. Nevertheless, the dissertation aims to introduce more general and more sophisticated methods to better capture the characteristics of the underlying stochastic processes. Consequently, these approaches fall beyond the scope of the present investigation.

The OU process is characterized by the following stochastic differential equation [Ornstein et al, 1930]

$$dp(t) = \vartheta\left(\mu - p(t)\right) dt + \sigma dW\left(t\right), \tag{2.5}$$

where $W\left(t\right)$ is a Wiener process [Doob, 1953] and $\vartheta > 0$ (mean reversion coefficient),

μ (long-term mean) and $\sigma > 0$ (volatility) are constants. One can obtain its solution with the Itō-Doeblin formula [Ito, 1944], which is

$$p(t) = p(0)\, e^{-\vartheta t} + \mu\left(1 - e^{-\vartheta t}\right) + \int_0^t \sigma e^{-\vartheta(t-s)} dW(s). \tag{2.6}$$

This implies that

$$\mathbf{E}\left(p(t)\right) = \mu(t) = p(0)\, e^{-\vartheta t} + \mu\left(1 - e^{-\vartheta t}\right) \tag{2.7}$$

and asymptotically

$$\lim_{t\to\infty} p(t) \sim N\left(\mu, \sqrt{\frac{\sigma^2}{2\vartheta}}\right). \tag{2.8}$$

Parameter ϑ determines the convergence speed of the process towards the mean, and inversely indicating the level of uncertainty via the standard deviation of its asymptotic distribution. Hence, for convergence trading, larger ϑ implies a better portfolio, as it quickly returns to the mean yielding a minimum amount of uncertainty in stationary case.

Since in real trading environments the time is treated as a discrete quantity, in the following discussion we view the asset prices as a first order, vector autoregressive VAR(1) process. This approach is used in [d'Aspremont, 2011] to get a grasp at the continuous OU process, as a VAR(1) process exhibits mean reverting properties if $|a_1| < 1$ holds for the dominant eigenvalue of matrix $\mathbf{A}$ in (2.9). Assume that $\mathbf{s}_t^T = [s_{1,t}, ..., s_{n,t}]$ is subject to a first order vector autoregressive process, VAR(1), defined as follows:

$$\mathbf{s}_t = \mathbf{A}\mathbf{s}_{t-1} + \mathbf{W}_t, \tag{2.9}$$

where $\mathbf{A}$ is a matrix of type $n \times n$ and $\mathbf{W}_t \sim N(0, \sigma^2 I)$ are i.i.d.r.v.-s for some $\sigma > 0$.

Without loss of generality, we can assume that the portfolio evaluations have zero mean, then the variance of the portfolio is

$$\sigma^2(t) = E\left(p_t^2\right) = E\left(\mathbf{x}^T \mathbf{s}_t \mathbf{s}_t^T \mathbf{x}\right) = \mathbf{x}^T \mathbf{G} \mathbf{x}, \tag{2.10}$$

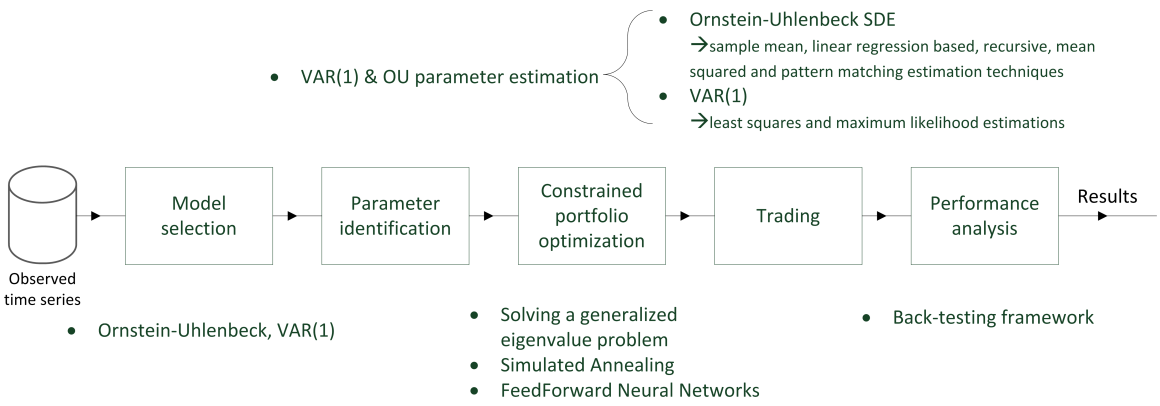

Figure 2-2: Computational approach — trading with mean reverting portfolios

while

$$\sigma^2(t-1) = E\left(\mathbf{x}^T\mathbf{A}\mathbf{s}_t\mathbf{s}_t^T\mathbf{A}^T\mathbf{x}\right) = \mathbf{x}^T\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x}. \tag{2.11}$$

$\mathbf{G}$ is the stationary covariance matrix of the VAR(1) process, and let $\mathbf{K}$ denote the covariance matrix of the noise. Defining the predictability of any selected mean reverting $\mathbf{x}$ portfolio can be formulated as [Box et al, 1977]:

$$\lambda = \frac{\sigma^2(t-1)}{\sigma^2(t)} = \frac{\sigma^2(t-1)}{\sigma^2(t-1)+\sigma^2_{noise}}. \tag{2.12}$$

It is clear that if λ is high then the contribution of the noise is small, as a result the portfolio is more predictable. This predictability measure also can be considered [d'Aspremont, 2011] as a proxy for the mean reversion coefficient ϑ, which is responsible for the uncertainty as shown in (2.8).

Furthermore — as an alternative objective function to maximizing the predictability of mean reverting portfolios — I maximized the average return on this type of portfolios. This is carried out assuming the underlying marginal probability density functions of the OU processes. In order to estimate the parameters of OU processes I have used multiple methods and their performance was compared. The novel portfolio identification methods with the corresponding trading strategies have been tested numerically on real financial time series, and the results exhibit profits.

The model and computational procedure is depicted by Fig. 2-2.

As a first step, the VAR(1) model parameters (matrices $\mathbf{A}$, $\mathbf{G}$, $\mathbf{K}$) have to be

identified based on the available observations (section 2.1.1). In the next block, different portfolio optimization methods (detailed in section 2.3) will produce an optimal sparse portfolio vector: $\mathbf{x}_{opt}^T = [x_1, ..., x_N]$ under the constraint $card(\mathbf{x}) \leq l$. Based on the identified portfolio, the selected trading strategy (section 2.4) should decide on which trading action is to be launched. For the sake of comparison of the profitability of different methods we made performance analysis the results of which are given in section 2.5.

In the next sections, I am going to detail the algorithms in each box on the computational model.

2.1 Parameter identification

This section gives details on various methods to estimate the OU model parameters and the parameters of the VAR(1) process used as its proxy.

2.1.1 VAR(1) model parameter identification

As explained in the preceding sections, with the knowledge of the parameters $\mathbf{A}$, $\mathbf{G}$ and $\mathbf{K}$, we can apply various heuristics to approximate the l-dimensional, optimal, sparse mean reverting portfolio [Fogarasi et al, 2013a]. However, these matrices must be estimated from the historical observations of the random process $\mathbf{s}_t$.

We recall from our earlier discussion that we assume $\mathbf{s}_t$ follows a stationary, first order autoregressive process. In most cases the linear system of equations is over-determined, hence $\mathbf{A}$ is estimated using least squares estimation techniques, as

$$\hat{\mathbf{A}} : \min_{\mathbf{A}} \sum_{t=2}^{T} \|\mathbf{s}_t - \mathbf{A}\mathbf{s}_{t-1}\|^2, \tag{2.13}$$

where $\|\cdot\|^2$ denotes the Euclidian norm.

Solving the minimization problem above, by equating the partial derivatives to zero with respect to each element of the matrix $\mathbf{A}$, we obtain the following system of

linear equations:

$$\sum_{k=1}^{n} \hat{\mathbf{A}}_{i,k} \sum_{t=2}^{T} s_{t-1,k} s_{t-1,j} = \sum_{t=2}^{T} s_{t,i} s_{t-1,j} \forall i, j = 1, ..., n. \tag{2.14}$$

Solving (2.14) for $\hat{\mathbf{A}}$ and switching back to vector notation for $\mathbf{s}$, we obtain

$$\hat{\mathbf{A}} = \sum_{t=2}^{T} (\mathbf{s}_{t-1} \mathbf{s}_{t-1}^T)^+ (\mathbf{s}_{t-1} \mathbf{s}_t^T), \tag{2.15}$$

where $\mathbf{M}^+$ denotes the Moore-Penrose pseudoinverse of matrix $\mathbf{M}$. Note that the Moore-Penrose pseudoinverse is preferred to regular matrix inversion, in order to avoid problems which may arise because of the potential singularity of $\mathbf{s}_{t-1} \mathbf{s}_{t-1}^T$.

Assuming that $\mathbf{s}_t$ is stationary for a $t > t_0$, we estimate $\mathbf{G}$ with the sample covariance matrix obtained as:

$$\hat{\mathbf{G}} = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{s}_t - \bar{\mathbf{s}}) (\mathbf{s}_t - \bar{\mathbf{s}})^T, \tag{2.16}$$

where $\bar{\mathbf{s}}$ is the sample mean vector of the assets defined as

$$\bar{\mathbf{s}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_t. \tag{2.17}$$

The covariance matrix $\mathbf{K}$ of random variable $\mathbf{W}_t$ can be estimated based on the assumption that the noise terms in equation (2.9) are i.i.d.r.v.-s with $\mathbf{W}_t \sim N(0, \sigma_W \mathbf{I})$ for some $\sigma_W > 0$, where $\mathbf{I}$ denotes the identity matrix. Then we obtain the following estimate for σ_W using:

$$\hat{\sigma}_W = \sqrt{\frac{1}{n(T-1)} \sum_{t=2}^{T} \left\| \mathbf{s}_t - \hat{\mathbf{A}} \mathbf{s}_{t-1} \right\|^2}. \tag{2.18}$$

In the more general case, when the terms of $\mathbf{W}_t$ are correlated, we can estimate the

covariance matrix $\mathbf{K}$, of the noise as follows:

$$\hat{\mathbf{K}} = \frac{1}{(T-1)} \sum_{t=2}^{T} (\mathbf{s}_t - \hat{\mathbf{A}}\mathbf{s}_{t-1})(\mathbf{s}_t - \hat{\mathbf{A}}\mathbf{s}_{t-1})^T. \tag{2.19}$$

Calculating the VAR(1) matrices by the abovementioned formulas is computationally intensive in general due to the large number of dimensions. Another approach is to determine $\mathbf{G}$ by the Lyapunov equation as in Fogarasi et al [2013a]. Nevertheless, reliable estimations can be obtained based on relatively short time windows, as numerically shown in Fogarasi et al [2013a], hence the running time is not a concern in our case. By the same token, the usage of short windows let us quickly react to new market situations. Buying portfolios much earlier than they converged close to their long term mean compensates the rough estimation of model parameters.

In this way the best fit VAR(1) model is selected, for the time series without cardinality constraint, which might have a large portfolio of potential assets (e.g. considering all 500 stocks which make up the S&P 500 index) from which the best sparse mean reverting portfolio is to be chosen.

2.1.2 OU parameter estimation

Determining the predictability of a given portfolio (2.12) is dependent on the estimated VAR(1) model parameters (i.e. matrices $\mathbf{A}$ and $\mathbf{G}$). If, in turn, we consider trading based on maximizing the average return — as it is going to be detailed in the next section —, only the parameters of the OU process are needed. This is another advantage of the new objective function, as these are scalar values in comparison with the matrix valued parameters of the VAR(1) model. Consequently, the estimation becomes more reliable in the case of a short time window of observations, and it also becomes plausible when dealing with high number of assets. However, these parameters are to be identified for each investigated portfolio, given by $p\,(t)$ process. For the sake of simplicity, let $\mathbf{p}^T = [p_1, ..., p_T]$ denote a vector constructed from the observed portfolio valuations.

Therefore, the objective of this section is to estimate parameter μ and ϑ in (2.7).

Estimating the long term mean (μ) of the process of portfolio valuations $p(t)$ is instrumental for mean reverting trading. Having an OU process at hand, this estimation can be performed in multiple ways [Fogarasi et al, 2012]. In addition to the estimation of μ, the least squares regression can estimate ϑ and σ parameters for any selected portfolio.

Sample mean estimation

As a benchmark for other methods, the long term mean can be estimated by simply taking the empirical average of the observed data:

$$\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}^T \mathbf{s}_t. \tag{2.20}$$

Least squares regression

Rewriting the Ornstein-Uhlenbeck equation stochastic differential equation (2.5) to the following way

$$p_t = \left(1 - e^{-\vartheta \Delta t}\right)\mu + e^{-\vartheta \Delta t} p_{t-1} + \sigma \sqrt{\frac{1 - e^{-2\vartheta \Delta t}}{2\vartheta}} dW(t) \tag{2.21}$$

results in a linear regression [van Gelder, 2009] in the form of

$$y = a + bx + \varepsilon_t. \tag{2.22}$$

Hence if we regress p_t against p_{t-1}, and derive $a = \left(1 - e^{-\hat{\vartheta}\Delta t}\right)\hat{\mu}$ and $b = e^{-\hat{\vartheta}\Delta t}$, then the estimation of the OU process parameters can be formulated as

$$\hat{\vartheta} = -\frac{\ln(b)}{\Delta t}, \quad \hat{\sigma} = sd(\varepsilon_t)\sqrt{\frac{2\hat{\vartheta}}{1 - b^2}} \quad \text{and} \quad \hat{\mu} = \frac{a}{1 - b}, \tag{2.23}$$

where $sd(\varepsilon_t)$ denotes the standard deviation of ε_t over t. Note that in the lack of the mean reverting property, when $\hat{\vartheta}$ is close to zero, the estimation of $\hat{\mu}$ could suffer from numerical instability.

Recursive solution

As the selected linear combination of VAR(1) processes is also VAR(1), then we can use the following recursive function:

$$p_1 = \mathbf{x}^T \mathbf{s}_1 \tag{2.24}$$

$$\widehat{\mathbf{s}}_{t+1} = \widehat{\mathbf{A}} \mathbf{s}_t \tag{2.25}$$

$$p_{t+1} = \mathbf{x}^T \widehat{\mathbf{A}} \mathbf{s_t} \tag{2.26}$$

This recursion should be performed until the difference between two steps is smaller then a given threshold, when the process is sufficiently close to the mean.

$$\hat{\mu} = p_{t+k}, |p_{t+k} - p_{t+k-1}| < \varepsilon \tag{2.27}$$

If the recursion is divergent that means that the process does not have the mean reversion property.

Minimizing the mean-square error

Among all possible Ornstein-Uhlenbeck processes described by equation

$$\begin{aligned} &\boldsymbol{\mu}\left(p_1, \mu, \vartheta, t\right) = \left(p_1 - \mu\right) e^{-\vartheta t} + \mu \\ &\mathrm{M} : \left\{ \boldsymbol{\mu}\left(p_1, \mu, \vartheta, t\right), \vartheta = \hat{\vartheta}, t = 1, ..., T \right\} \end{aligned} \tag{2.28}$$

we would like to select the one with least squared error against the observed portfolio valuation vector:

$$\hat{\mu} = \min_{\boldsymbol{\mu} \in M} \left\| \mathbf{p} - \boldsymbol{\mu} \right\|^2 \tag{2.29}$$

Solving the $\frac{\partial}{\partial \mu} \|\mathbf{p} - \boldsymbol{\mu}\|^2 = 0$ equation to find the optimal mean gives the following estimation:

$$\hat{\mu} = \frac{\sum_{t=1}^{T} \left(p_t - p_1 e^{-\vartheta t}\right)}{\sum_{t=1}^{T} \left(1 - e^{-\vartheta t}\right)}. \tag{2.30}$$

Pattern matching

Similarly to the previous method, pattern matching [Fogarasi et al, 2012] aims to find the maximum-likelihood estimation for the observed portfolio valuations among all possible processes (2.28). Taking into account that the samples follow a Gaussian distribution, this leads to the optimization problem

$$\max_{\boldsymbol{\mu} \in \mathrm{M}} \frac{1}{\sqrt{2\pi \det(\mathbf{U})}} e^{-\frac{1}{2}(\mathbf{p}_t - \boldsymbol{\mu}_t)^T \mathbf{U}^{-1} (\mathbf{p}_t - \boldsymbol{\mu}_t)} \tag{2.31}$$

being equivalent with

$$\min_{\boldsymbol{\mu} \in \mathrm{M}} \boldsymbol{\mu}_t \mathbf{U}^{-1} \boldsymbol{\mu}_t - 2 \boldsymbol{\mu}_t \mathbf{U}^{-1} \mathbf{p}_t, \tag{2.32}$$

where $\mathbf{U}$ denotes the covariance matrix of p_t: $\mathbf{U}_{i,j} = \frac{\sigma^2}{2\vartheta} \left(e^{-\vartheta |i-j|} - e^{-\vartheta (i+j)}\right)$. As the optimization problem is quadratic, it can be solved analytically

$$\frac{\partial}{\partial \mu} \boldsymbol{\mu}_t \mathbf{U}^{-1} \boldsymbol{\mu}_t - 2 \boldsymbol{\mu}_t \mathbf{U}^{-1} \mathbf{p}_t = 0 \tag{2.33}$$

resulting in the following closed form solution:

$$\hat{\mu} = \frac{\left(\mathbf{p}^{\mathrm{T}} - p_1 \mathbf{v}_2^T\right) \mathbf{U}^{-1} \mathbf{v}_1}{\mathbf{v}_1^T \mathbf{U}^{-1} \mathbf{v}_1}, \tag{2.34}$$

where $\mathbf{v}_1 = \left[\left(1 - e^{-\vartheta t}\right), t = 1, ..., T\right]$ and $\mathbf{v}_2 = \left[e^{-\vartheta t}, t = 1, ..., T\right]$ respectively.

Conclusions

For the sake of finding the best estimation procedure, the different methods were tested on artificially generated data. Each observation was with the following parameters $\sigma = 1.5$, $T = 8$ and $\mu, p_1 \in [0; 100]$, respectively. The comparison was done

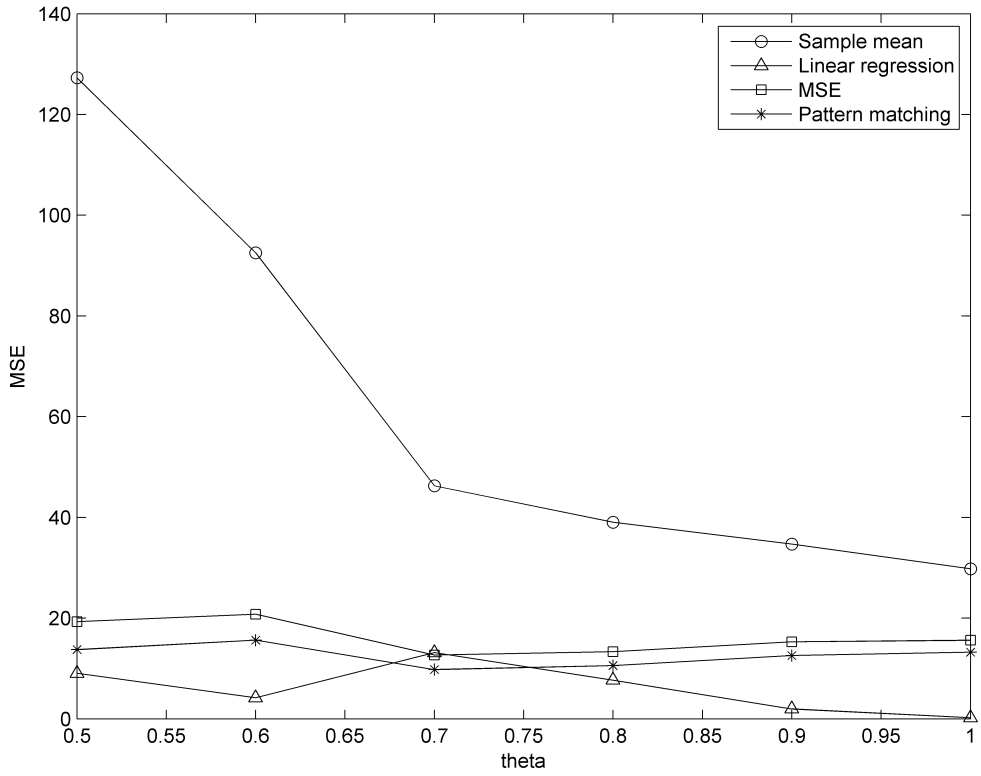

Figure 2-3: Comparison of different mean estimation techniques

independently for mean reversion coefficients in $\vartheta \in [0.5; 1]$, and in each case the mean squared error was taken into account for 100 generated processes. The results are shown by Fig. 2-3.

According to the numerical simulations, in the abovementioned testing procedure linear regression has shown the highest accuracy for estimating the long term mean, hence that method was chosen for trading throughout the chapter.

2.2 Objective functions for optimal portfolio selection

Having identified the parameters one can zoom down to portfolio optimization. In this section two objective functions are discussed for optimal portfolio selection:

1. Optimizing the portfolio subject to the traditional λ maximization [d'Aspremont, 2011];

2. Optimizing the portfolio subject to maximizing the average return as a novel approach.

2.2.1 Maximizing the predictability

The traditional way [d'Aspremont, 2011] to identify the optimal sparse mean reverting portfolio is to find a portfolio vector subject to maximizing its predictability.

One may note that

$$\mathbf{x_{opt}} = \arg\max_{\mathbf{x}} \lambda = \arg\max_{\mathbf{x}} \frac{\mathbf{x}^T\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x}}{\mathbf{x}^T\mathbf{G}\mathbf{x}} \tag{2.35}$$

is equivalent to finding the eigenvector corresponding to the maximum eigenvalue in the following generalized eigenvalue problem (Box et al [1977] and d'Aspremont [2011]):

$$\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x} = \lambda\mathbf{G}\mathbf{x}. \tag{2.36}$$

As a result, portfolio optimization is then cast as the following constrained optimization problem:

$$\mathbf{x_{opt}} : \max_{\mathbf{x}} \frac{\mathbf{x}^T\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x}}{\mathbf{x}^T\mathbf{G}\mathbf{x}}, card\,(\mathbf{x}) \leq l. \tag{2.37}$$

Introducing cardinality constraints in order to minimize the transaction costs, which involves optimizing sparse portfolios, turns out to be an NP hard problem [Natarjan, 1995]. Thus an efficient heuristic solution is to be developed to solve (2.37) [Fogarasi et al, 2013b].

2.2.2 Maximizing the average return

However, instead of maximizing the predictability a more general objective function having a direct impact on the trading profit can be introduced.

The maximization problem can be posed as follows:

$$\Psi(\mathbf{x}) = \max_{0 \leq t} E(p(t)) - p(0) = \max_{0 \leq t} (\mu(t) - p(0)) \tag{2.38}$$

or equivalently

$$\Psi(\mathbf{x}) = \max_{0 \leq t} \left((p(0) - \mu) e^{-\vartheta t} + \mu - \mathbf{x}^T \mathbf{s}_0 \right) \tag{2.39}$$

and

$$\mathbf{x}_{\text{opt}} = \arg\max_{\mathbf{x}} \Psi(\mathbf{x}). \tag{2.40}$$

Furthermore, another objective function can be obtained if we take into account that a risk free interest is available with interest rate r_f, which allows discounting the expected future portfolio value over time. In this way, we have the following expression by replacing the future value with its net present value:

$$\Psi^*(\mathbf{x}) = \max_{0 \leq t} \frac{E(p(t))}{(1 + r_f)^t} - p(0), \tag{2.41}$$

where the optimal solution for t, which is the optimal holding time, is given as

$$t = \frac{1}{\vartheta} \ln \left(\frac{(\mu - p(0)) (\vartheta + \ln(1 + r_f))}{\mu \ln(1 + r_f)} \right), \tag{2.42}$$

which comes from solving

$$\frac{\partial \Psi^*(\mathbf{x})}{\partial t} = \frac{\partial}{\partial t} \left(\frac{(p(0) - \mu) e^{-\vartheta t} + \mu}{(1 + r_f)^t} - \mathbf{x}^T \mathbf{s}_0 \right) = 0. \tag{2.43}$$

for t.

By using (2.38) or (2.41), portfolio optimization can again be reduced to a constrained optimization problem:

$$\mathbf{x}_{\text{opt}} = \arg\max_{\mathbf{x}} \Psi(\mathbf{x}), card(\mathbf{x}) \leq l. \tag{2.44}$$

Since (2.44) does not have a known analytical solution, as opposed to (2.35), I used stochastic search methods to find the optimal solution.

2.3 Optimization and dealing with the cardinality constraint

Having identified the model parameters, we can now focus on portfolio optimization. In the absence of proper analytical solutions for the constrained optimization problems posed in section 2.2.2, we use simulated annealing and feedforward neural networks for obtaining good quality (as close to the global optimum as possible) heuristic solutions (Kirkpatrick et al [1983] and Hornik et al [1989]).

2.3.1 Simulated annealing for portfolio optimization

Simulated annealing [Kirkpatrick et al, 1983] is a stochastic search method for finding the global optimum in a large search space as described in many related papers (Kirkpatrick et al [1983], Janaki et al [1996] and Fogarasi et al [2013b]).

In this context the negative energy function $J(\mathbf{x})$ is equivalent with the objective function by which the optimal portfolio vector is sought. One can use either the first objective function maximizing the eigenvalue (as in section 2.2.1) [Fogarasi et al, 2013b] or the second objective function maximizing the average return (as in section 2.2.2) for the selected portfolio:

1. $-J(\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x}}{\mathbf{x}^T\mathbf{G}\mathbf{x}}$;
2. $-J(\mathbf{x}) = \max\limits_{0\leq t}\left(\frac{(p(0)-\mu)e^{-\vartheta t}+\mu}{(1+r)^t} - \mathbf{x}^T\mathbf{s}_0\right)$.

In chapters 3, 4 and 5 further functions will be introduced enabled by new modelling and implementation approaches.

With an appropriate neighbour function the cardinality constraint $card(\mathbf{x}) \leq l$ is automatically fulfilled at each step of the algorithm. The neighbour function on each iteration generates a new portfolio on the L1-ball in a randomly chosen l dimensional subspace, the distance of which from the previous one depends on the current temperature T_{SA}. The algorithm automatically ensures that the available cash is not exceeded either with the long positions, or the short positions. Let $\mathbf{x}$ be an arbitrary

initialization vector, and then by calling a random number generator a new vector $\mathbf{x}'$ is generated randomly subject to the abovementioned neighbour function. We automatically accept the new vector if $J(\mathbf{x}') < J(\mathbf{x})$, or if $J(\mathbf{x}') \geq J(\mathbf{x})$ then due to random acceptance with $e^{-\frac{J(\mathbf{x}')-J(\mathbf{x})}{T_{SA}}}$ probability. The sampling is then continued while decreasing T_{SA} until zero. The last state vector is now the identified optimal sparse portfolio vector.

2.3.2 Applying FeedForward Neural Networks

As another approach for solving the optimization problem posed in (2.37), I develop here a fast approximation method for efficient optimal portfolio selection by using FeedForward Neural Networks (FFNN).

To estimate the optimal portfolio vector in (2.37) from the identified VAR(1) matrices, a universal approximator, in our case FFNNs can be utilized. FFNNs are described by the following mapping [Hornik et al, 1989]:

$$\mathbf{x} = Net(\mathbf{z}, \mathbf{w}) = \varphi\left(\sum_j w_j^{(L)} \varphi\left(\sum_l w_{jl}^{(L-1)} .... \varphi\left(w_{nm}^{(1)} z_m\right)\right)\right), \tag{2.45}$$

where $\mathbf{z}$ is the input vector and $\mathbf{w}$ vector denotes the free parameters. They have universal representation capabilities in L^2 (Hornik et al [1989] and Cybenko [1989]) in terms of

$$\forall \varepsilon > 0, f(\mathbf{z}) \in L^2 \rightarrow \exists \mathbf{w} : \|f(\mathbf{z}) - Net(\mathbf{z}, \mathbf{w})\| < \varepsilon, \tag{2.46}$$

hence they can approximate any continuous function on $\mathbf{R}^N$.

Furthermore, they can learn and generalize from a finite set of examples $\tau^{(K)} = \{(\mathbf{z}_k, \mathbf{d}_k), k = 1, ..., K\}$ by using the back propagation (BP) algorithm [Hagan et al, 1994]. As a result, one may look upon the optimal portfolio selection problem as a mapping from the identified, $\mathbf{A}, \mathbf{G}, \mathbf{K}$ matrices of the underlying VAR(1) process to the optimal sparse portfolio vector $\mathbf{x}$. In this case, the input vector of FFNN is $\mathbf{z} = (\mathbf{A}, \mathbf{G}, \mathbf{K})$ constructed by matrix flattening and the output is vector $\mathbf{x}$ with $card(\mathbf{x}) \leq l$ to fulfil the sparsity constraint (Fig. 2-4).

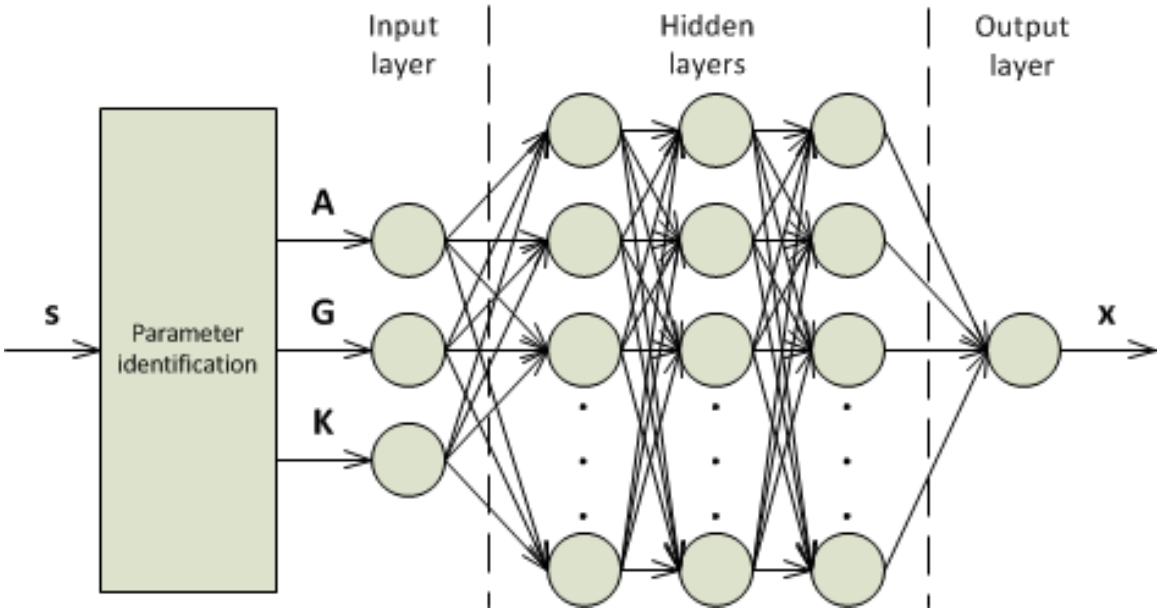

Figure 2-4: FFNNs for finding the optimal sparse portfolio vectors

One can construct the training set $\tau^{(K)}$ by finding the optimal sparse portfolio vectors for some input matrices by exhaustive search. Unfortunately as the input layer has a size of $3N^2$, where N is the number of assets, this solution can be used only for lower dimensional spaces. The construction of the training set is done according to the following computational model (Fig. 2-5).

Once the training set has been constructed, a back propagation algorithm is used to optimize the weights (the free parameters) of the corresponding FFNN by globally minimizing the following objective function [Hagan et al, 1996, chapters 11 and 12]:

$$\mathbf{w}_{opt} : \min_{\mathbf{w}} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{z}_k - Net\left(\mathbf{z}_k, \mathbf{w}\right)\|^2. \tag{2.47}$$

Following the identification of the weights, which has to be done prior to the trading once, the optimal portfolio vector is given by:

$$\mathbf{x}_{opt} = Net\left(\mathbf{z}, \mathbf{w}_{opt}\right). \tag{2.48}$$

The sparsity constraint was enforced by truncation, i.e. setting the $N-l$ elements with the smallest absolute value to zero. The neural network had three hidden layers.

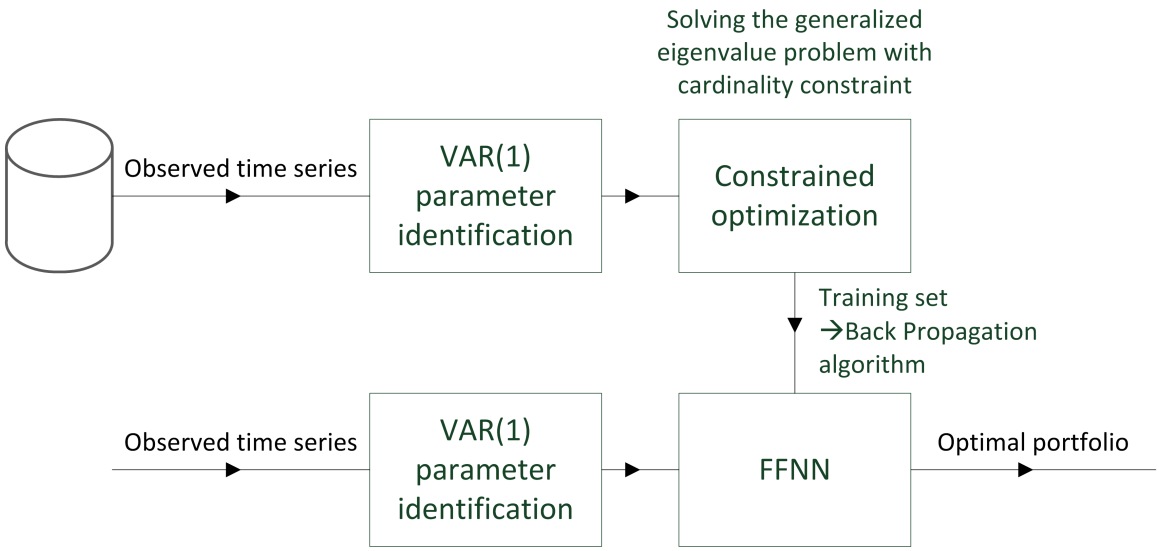

Figure 2-5: Training and using FFNNs for portfolio optimization

I enhanced the performance of the previous methods in terms of convergence speed by combining a feedforward neural network with simulated annealing. In this method the output of the trained FFNNs was used as the starting point of the simulated annealing. Note that the time consuming training of the FFNN has to be performed only once, and this trained FFNN can then be reused in each time instance for finding the optimal portfolio vector. By providing an already optimized portfolio vector for the SA to begin with, better quality results can be achieved with a considerable speedup in the operation.

2.4 Trading algorithms

Having the optimal portfolio at hand, the developed trading algorithms are introduced in this section to carry out the corresponding trading. In the proposed trading algorithms, trading is described as a walk in a binary state space in which either we already have a portfolio at hand or cash at hand. The transitions between the two states are only affected

1. either by the state of the portfolio valuation $p(t)$;

2. or by the evaluations of the potentially owned and the newly identified optimal

portfolio by (2.38) or (2.41).

Two alternatives of the trading strategy are introduced which differs in whether is it allowed to sell an owned portfolio before it reached its expected mean and invest to a new portfolio with higher expectations instead. Each trading algorithm is then formalized by a state chart (Fig. 2-6 and Fig. 2-7).

2.4.1 Trading by the λ of OU process

After the optimal portfolio is selected based on maximizing the predictability proxy λ, the trading algorithm is responsible for interpreting the selected portfolios as a sequence of trading actions. In the first place it has to be decided whether the portfolio is below or above its long term mean, or in its stationary state. This is a fundamental part of the trading strategy as this determines whether to buy or sell a mean reverting portfolio.

However, we do not know the exact value of μ, only an estimate (various identification methods are listed in section 2.1.2), therefore this can be perceived as a decision theoretic problem. In stationary state the portfolio has a normal distribution

$$p_t \sim N\left(\mu, \frac{\sigma}{\sqrt{2\vartheta}}\right) \tag{2.49}$$

thus we can formulate hypotheses based on the probability that the observed portfolio value is in the stationary state as follows [Fogarasi et al, 2012]:

- Hypothesis 1, the portfolio is under the mean ($p(t) < \mu - \alpha \rightarrow p(t) < \mu$) with the probability of

$$\int_{-\infty}^{\mu-\alpha} \frac{\sqrt{2\vartheta}}{\sqrt{2\pi}\sigma} e^{-\frac{(u-\mu)^2}{\frac{\sigma^2}{\vartheta}}} du = \frac{\varepsilon}{2}; \tag{2.50}$$

- Hypothesis 2, the portfolio is above the mean ($p(t) > \mu + \alpha \rightarrow p(t) > \mu$) with the probability of

$$\int_{\mu+\alpha}^{\infty} \frac{\sqrt{2\vartheta}}{\sqrt{2\pi}\sigma} e^{-\frac{(u-\mu)^2}{\frac{\sigma^2}{\vartheta}}} du = \frac{\varepsilon}{2}; \tag{2.51}$$

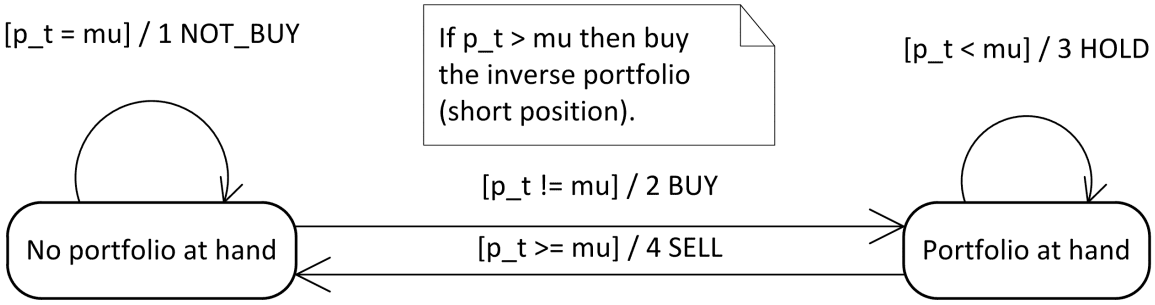

Figure 2-6: State chart of `nochange` trading strategy

- Hypothesis 3, the portfolio is in stationary state ($p(t) \in [\mu - \alpha; \mu + \alpha] \rightarrow p(t) = \mu$) with the probability of

$$\int_{\mu-\alpha}^{\mu+\alpha} \frac{\sqrt{2\vartheta}}{\sqrt{2\pi}\sigma} e^{-\frac{(u-\mu)^2}{\frac{\sigma^2}{\vartheta}}} du = \varepsilon. \tag{2.52}$$

By adjusting the parameter ε one can set the probability of making the wrong choice if the portfolio has already reached its stationary state. Adjusting this parameter affects how cautious the trading will be.

In this case the agent buys the portfolio only if the identified portfolio has not reached its stationary state. Note that if the portfolio is above its estimated long term mean, then the agent buys the inverse portfolio instead in which every long and short position is switched to its opposite. And after having a portfolio at hand the agent holds it until it reaches its stationary state (estimating the long term mean and making the decision is recalculated in every time instance according to the sliding window used) then sells it.

2.4.2 Trading by maximizing the average return

In the second case, the portfolios are evaluated by calculating the corresponding objective function. Positive evaluation indicates a profitable portfolio, while negative evaluation indicates that the portfolio may produce a loss. Based on this, the agent buys or holds a portfolio only if it has a positive evaluation. A new trading action is taken if a newly identified portfolio (`x_opt`) has higher and also positive evaluation

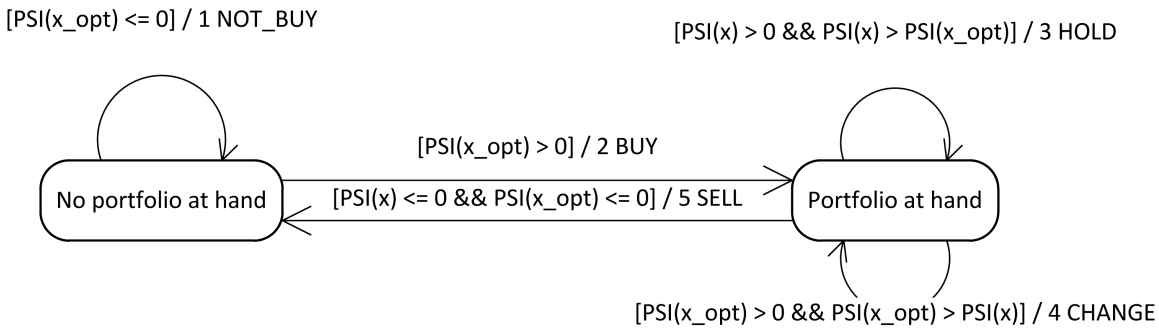

Figure 2-7: State chart of `change` trading strategy

than the present one (`x`). In this case one can sell the owned portfolio and buy the new one with higher expectations instead. This approach treats the present portfolio as a sunk cost, thus only the future expectations are taken into consideration. Hence, we do not have to give up the best available portfolio in favour of a presently unfavourable portfolio (Fig. 2-7).

2.5 Simulation results and performance analysis

An extensive back-testing framework was created to provide numerical results for comparison of different methods on different financial time series.

For a detailed comparative analysis the following performance measures were calculated for each simulation: (i) minimal value $G_{\min} = \frac{1}{c_0} \min_{0 \le t \le T} c_t$; (ii) final value $G_{final} = \frac{c_T}{c_0}$; (iii) maximal value $G_{\max} = \frac{1}{c_0} \max_{0 \le t \le T} c_t$; (iv) average value $G_{avg} = \frac{1}{c_0} \frac{1}{T} \sum_{t=1}^{T} c_t$, where c_t denotes the sum of owned cash and the market value of the owned portfolio at time instance t, while c_0 denotes the initial cash (in each case the agent started with \$10,000).

Details on the used financial time series are given in the Appendix 7. Regarding the sparsity constraint, 3 assets were selected in each transaction. This constraint is further motivated by taking into account realistic market scenarios, e.g. if round lots are present (the smallest amount of assets that can be purchased) it would not be feasible to hold dense portfolios consisting only a small number of assets. Another example is, when trading is done on an order book, some of the assets might not

be available to buy at the same time or at required quantity. These considerations are taken consistently throughout the dissertation, and, as a result, the numerical results obtained for sparse portfolios has not been compared to the performance of unconstrained portfolios which are unrealistic in practice.

In this period, the U.S. SWAP rates had a decreasing tendency with simple buy and hold strategy (-12.11% on average). The bar chart (Fig. 2-8) shows that all of the introduced methods outperform this tendency, and in the scenario when a FFNN was deployed the trading was profitable with a 13.49% yearly profit. It is noteworthy that the trading was performed without major drawdowns, i.e. the current owned capital has not dropped considerably under the initial amount, and the final position is very close to the best instance in the investigated period.

In the studied period the S&P 500 asset prices rose only by 12.03%, while maximizing the λ results in 33.40%; and maximizing the average profit (with the first trading algorithm) results in 53.55%. On the other hand, changing the portfolios before they could reach the long term mean proved to be a bad strategy (Fig. 2-9).

The FOREX rates in this period showed only a slight increase during the year (1.52%), our methods achieved up to 21.66% profit (Fig. 2-10).

As one can see, the novel portfolio optimization methods outperform the traditional eigenvalue maximization, and also deploying an FFNN can further increase the performance.

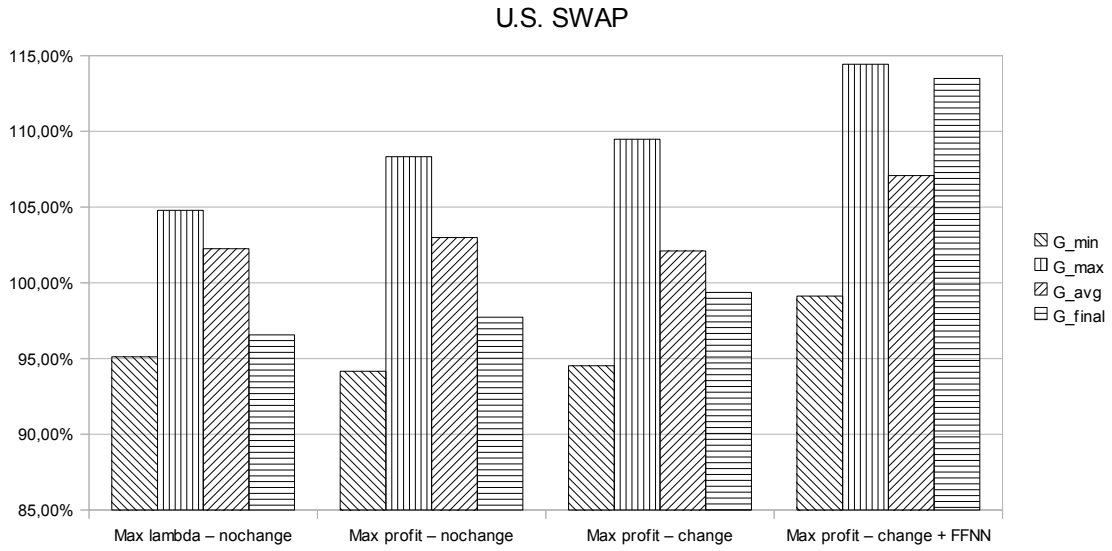

Figure 2-8: Trading results on SWAP with mean reverting portfolios

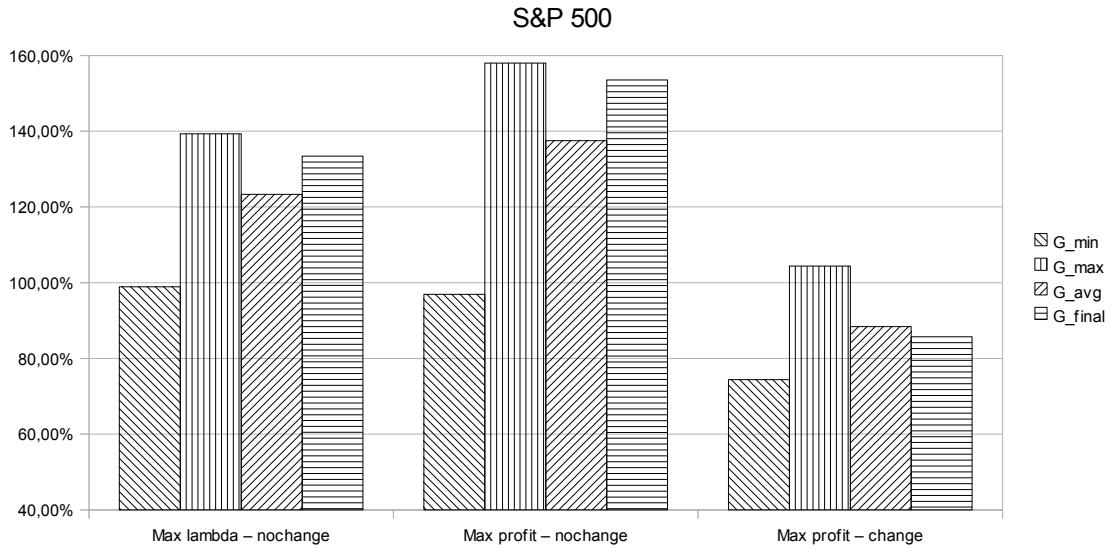

Figure 2-9: Trading results on S&P 500 with mean reverting portfolios

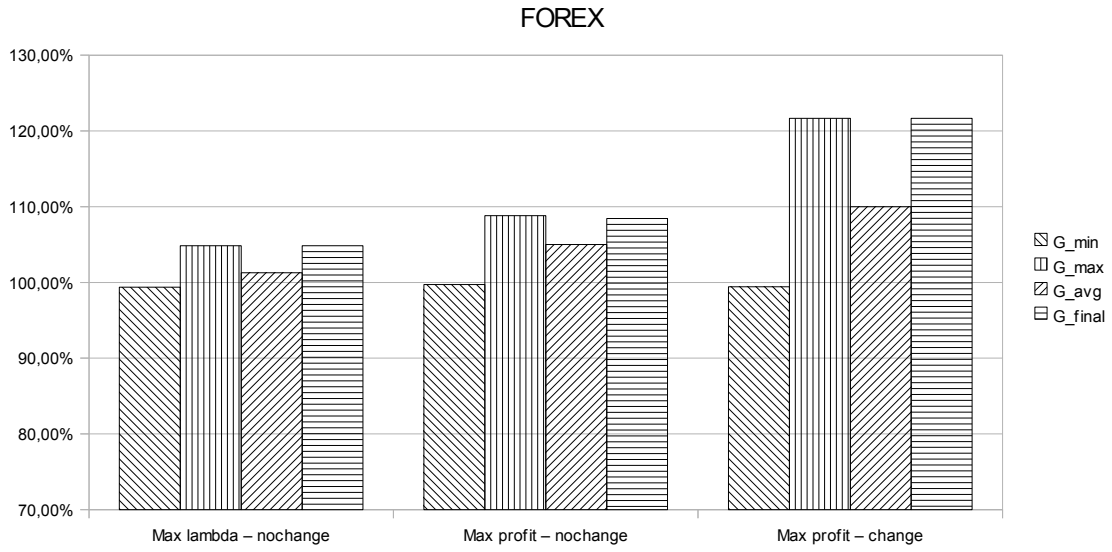

Figure 2-10: Trading results on FOREX with mean reverting portfolios

Chapter 3

Trading with Hidden Markov Models

Identifying mean reverting portfolios, as introduced in the previous chapter, aims to perform trading actions based on the estimated long term mean. Similarly, one can use other models for taking advantage of predicting the future behaviour of a given portfolio.

A Hidden Markov Model (HMM) [Baum et al, 1966] is a statistical model which is an extension of Markov chains. In this model, the current state is no longer directly visible to the observer, but each state emits an observable output quantity denoted by

$$\mathbf{X} = \{o_1, o_2, ..., o_T\} \tag{3.1}$$

and each emission depends only on the hidden state

$$\mathbf{Q} = \{q_1, q_2, ..., q_T\}\,. \tag{3.2}$$

The probability of an emitting a specific output is determined by the conditional probabilities

$$P\left(o_t|\mathbf{Q}\right) = P\left(o_t|q(t) = q_t\right), \tag{3.3}$$

while the transition probabilities of the underlying Markov chain, describing the jumping probabilities from one state to another is given by the transition probability matrix $A_{ij} = P\left(q(t+1) = q_i\,|q(t) = q_j\,\right)$. $\boldsymbol{\pi}_N$ denotes the initial distribution vector.

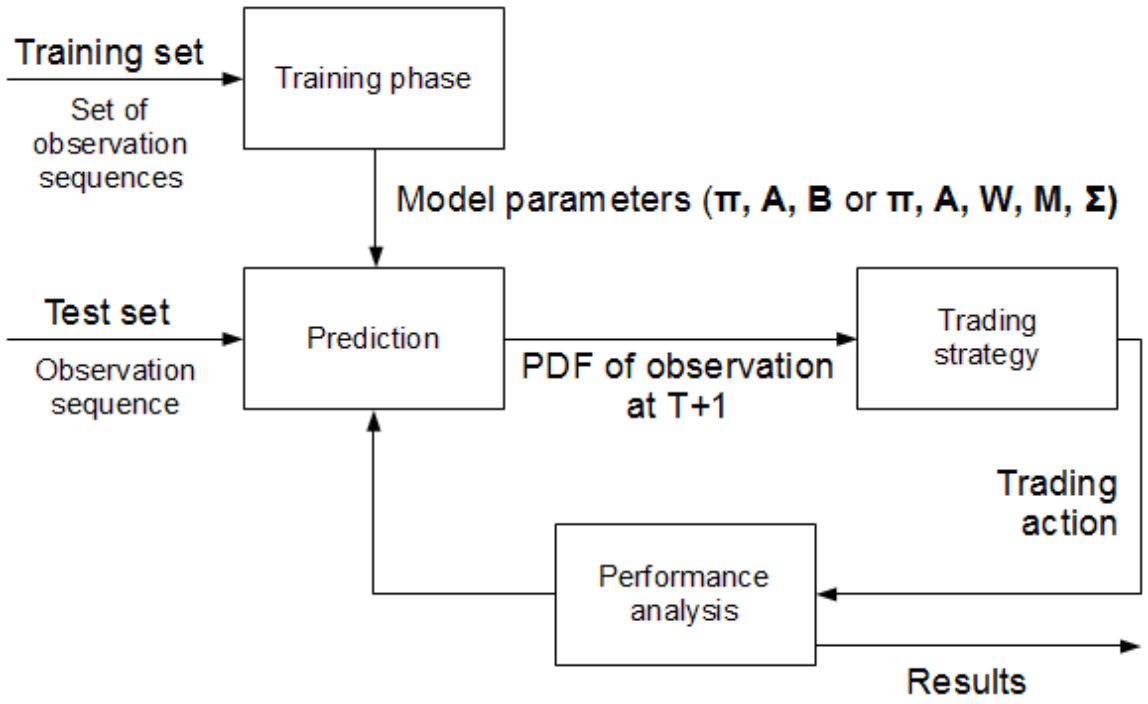

Figure 3-1: Computational approach — trading by HMMs

HMMs are commonly used in various fields, for instance in bioinformatics [Durbin et al, 1998] or speech recognition (Jurafsky et al [2006], Rabiner [1989]). Such models are also widely used in econometrics [Hamilton, 1990]. In our case they are used to predict future values of financial time series.

The computational model of trading by HMM is shown in Fig. 3-1.

Having the identified model parameters at hand, the model allows us to make predictions for the next time instance. Then the trading signal can be formed accordingly. The predictive performance of HMMs is investigated on financial time series in section 3.4, which demonstrated that a good average return can be obtained.

3.1 Description of an HMM

The observable output o_t of an HMM can be treated as a discrete or as a continuous value.

Discrete case

If we treat the observations having discrete values over a given alphabet

$$o_t \in \{k_1, k_2, ..., k_M\}, \tag{3.4}$$

then the observation probability matrix, $\mathbf{B}_{NxM}$ describes the probability distribution over the possible values for each state. For continuous data, pre-processing is needed to quantize the data to the discrete alphabet. The easiest alphabet has a three-element code indicating that the asset price is {increasing, stagnating, decreasing}. However, from the learning point of view, it is usually more efficient to form subsets with similar number of observations in each. In the discrete case, HMM is described by

$$\Theta = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\}. \tag{3.5}$$

Vector valued observations can be coded as n-tuples, note that in this case the alphabet size will grow according to $O(M^n)$.

Continuous case

When the observable output (o_t) is continuous, the observation probabilities are described by probability density function, instead of a probability matrix. For this purpose a multivariate Gaussian mixture model [Dasgupta, 1999] can be used, in which the density function is composed of a weighted sum (according to the weight matrix $\mathbf{W}$) of M independent Gaussian functions (depicted by Fig. 3-2:

$$P(o_t|q(t) = q_j) = \sum_{k=1}^{M} w_{jk} b_{jk}(o_t), \tag{3.6}$$

where $b_{jk}(o_t) \sim N\left(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\right)$. Note that this approach can handle multivariate observations as well.

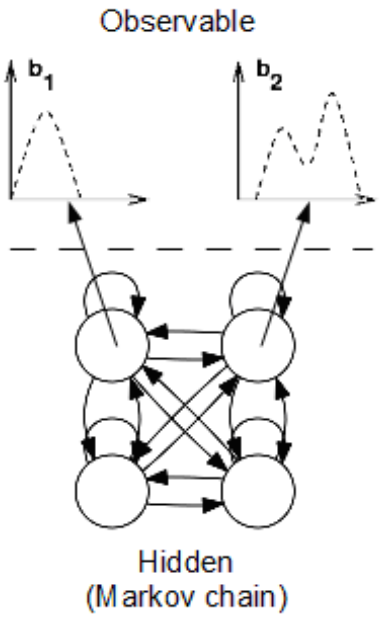

Figure 3-2: A hidden Markov model with continuous observations

In the continuous case, HMM can then be described as

$$\Theta = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{W}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}\,. \tag{3.7}$$

3.2 Training algorithms for identifying financial time series by HMMs

The model parameter estimation (learning) is a key aspect when we are using HMMs for prediction during high-frequency trading. We need a learning algorithm providing good quality results enough for profitable trading, and being fast enough at the same time to fit the available time window. Finding such a method with a good trade-off between accuracy and speed proves to be the main challenge when using HMMs.

First, the construction of a training set is needed. This is carried out by sampling the history of the input time series with a sliding window of length T: $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_K\}$, where $\mathbf{X}_t = \{\mathbf{r}_{t-T+1}, ..., \mathbf{r}_t\}$ and $\mathbf{r}_t$ is the daily return at the tth time instance. In the discrete case, the daily returns were partitioned into M equal sized subsets. In the case of the daily returns handled as continuous values, a normalization

step is applied:

$$r^*_{t,i} = \frac{r_{t,i} - \bar{\mathbf{r}}^{(i)}}{std(\mathbf{r}^{(i)})} \tag{3.8}$$

During training, the likelihood (or in practice, due to the small order of magnitude of such probabilities, the log-likelihood) of the model is maximized based on the given observations [Rabiner et al, 1986]:

$$\Theta_{opt} = \arg\max_{\Theta} P\left(\mathbf{X}|\Theta\right). \tag{3.9}$$

The optimization problem described in (3.9) can be solved in multiple ways. However, there is no computationally efficient algorithm known to find the global optimum. Thus, I have aimed at overcoming the limitations of the Baum-Welch algorithm. In the dissertation the following results have been achieved:

- I have used the traditional Baum-Welch (BW) algorithm.
- I provided good quality solutions by the use of Simulated Annealing (SA) to optimize the model parameters.
- I have combined the BW algorithm with SA to yield a fast and efficient hybrid learning algorithm to increase the efficiency of the underlying stochastic search, as a result find the global optimum of the model parameters faster.
- To further ease the underlying computational complexity, I have employed a new clustering algorithm to pre-process the training set.
- I have carried out dimensional reduction by probabilistic PCA (PPCA) which in turn allows one to control the model degree. In this way, one can avoid overfitting and also control the computational time, at the same token.

Baum-Welch algorithm

The Baum-Welch expectation maximization (BW EM) algorithm [Baum et al, 1970] is a mechanism to iteratively update the model (given by (3.5) or (3.7)) starting from

an arbitrary initial value and iterating until the likelihood of the model converges to a certain value. Since this is an iterative method, which can use the forward-backward algorithm, implemented in an efficient way by dynamic programming [Rabiner, 1989], this proves to be relatively fast. On the other hand, it may get stuck in one of the local maxima making the final result highly dependent on the initialization.

Simulated annealing

In the absence of analytical solutions for finding the global optimum of the likelihood of model parameters, one can use simulated annealing (SA) to obtain good quality heuristic solutions. Simulated annealing [Kirkpatrick et al, 1983] is a stochastic search method for finding the global optimum in a large search space. In this context the energy function $J(\Theta)$ is the log-likelihood of the selected model:

$$J(\Theta) = L(\Theta). \tag{3.10}$$

Let Θ be an arbitrarily initialized model, and then by calling a random number generator Θ' is generated subject to its constraints. The neighbour function on each iteration modifies one of the probability vectors or matrices with an amount according the current temperature T_{SA}. Accept the new model if $J(\Theta') > J(\Theta)$, or otherwise with $e^{-\frac{J(\Theta)-J(\Theta')}{T_{SA}}}$ probability. Continue the sampling while decreasing T_{SA} until zero. The last state now describes the identified model.

This method, in theory [Kirkpatrick et al, 1983], can provide us the best fitting model. It is a rather slow method due to the large dimension of the search space.

Hybrid method

Unfortunately, the BW algorithm tends to get stuck in one of the local maxima. On the other hand, in the case of applying simulated annealing, the convergence is rather slow in large dimensional search spaces. Therefore there is a need for a speedy but good-quality solution.

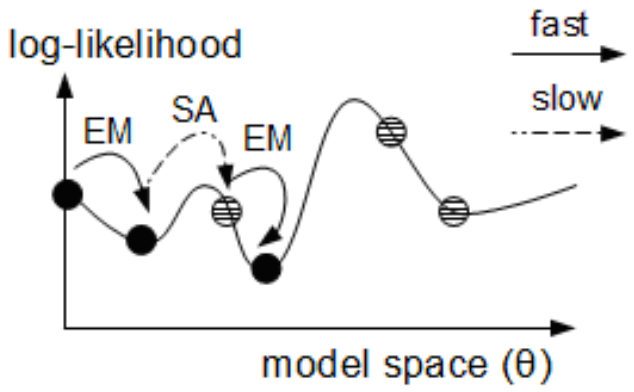

Figure 3-3: Optimization with the hybrid algorithm

Our objective is then to couple the BW algorithm with SA. The new algorithm operates as follows: first we start with SA randomly choosing a new accepted state. Then the BW algorithm takes over quickly searching for a local maximum around the state accepted by SA. Once this local maximum is found, then SA generates the next random state subject to a given schedule, from which BW searches for the local maximum again (see Fig. 3-3). This cycle is repeated until a given number of steps have been carried out (Fig. 3-4).

In this way, good quality solutions can be achieved in relatively short time. This approach can bring more reliable results for trading, which are less dependent on the initialization than the traditional Baum-Welch. The real-time nature of this algorithm can be ensured by limiting the number of steps of SA making it fit into a predefined time interval.

3.2.1 Reducing the complexity of the training

Another way of easing the computational complexity of the optimization problem posed in (3.9) is to reduce its state space. This is going to be treated by two different approach:

- assign the observations to hidden states with a clustering algorithm, and identify the HMM model parameters (Θ) cluster-wise;

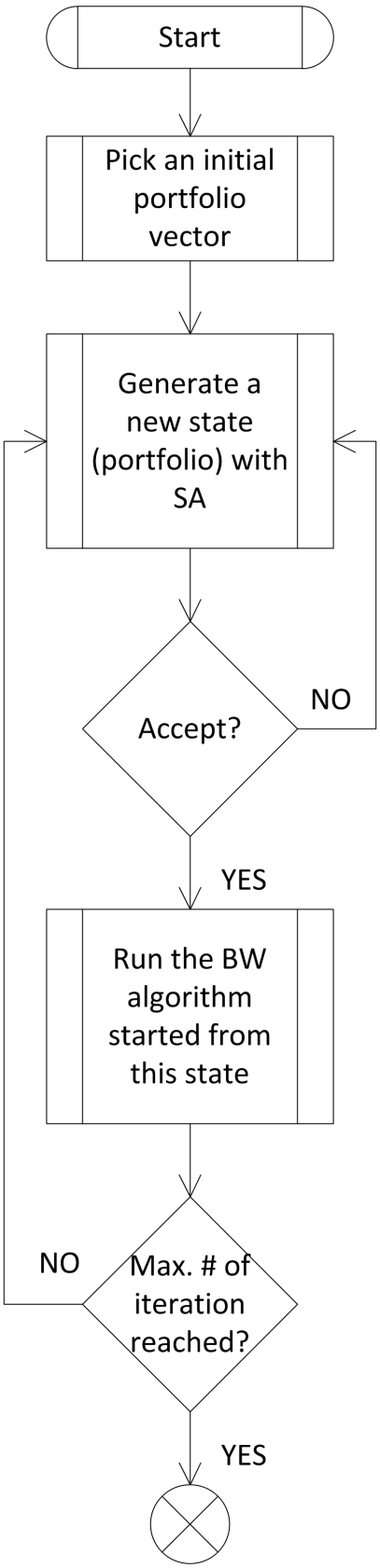

Figure 3-4: Flowchart of the hybrid algorithm

- perform dimension reduction to optimize in a smaller dimensional space.

Clustering algorithm

Optimizing model likelihood in the full space of Θ including every available observation is computationally exhausting. But by forming clusters out of the observations, and assigning them to the corresponding hidden states, the p.d.f. describing the observation probabilities can be estimated for each cluster, separately. In this way, the complexity can be reduced.

Former attempts to utilize clustering in the HMM training process proved to be beneficial in the field of speech recognition [Rabiner, 1989], however, applications for financial time series has yielded much more moderate performance [Idvall et al, 2008]. Rabiner addressed this topic in his work [Rabiner, 1989] suggesting the following algorithm in continuous case:

1. Arbitrary initialization of model parameters.

2. Segment the observations in the training set based on the optimal state sequence (Viterbi path) calculated by the Viterbi algorithm (Viterbi [1967] and Forney [1973]) and assign each data point to the corresponding maximum likelihood hidden states.

3. Perform k-means clustering [Bishop, 1995] over the observation vectors within each state resulting M clusters per state.

4. Re-estimate the model parameters in a standard manner (calculate the relative frequency of observations in each state and the relative frequency of transitions, estimate sample mean and covariance for each cluster).

5. Calculate statistical similarity with the previous model, if we can assume convergence then stop, continue from step 2 otherwise.

The main drawback of Rabiner's algorithm is that it performs a separate and linear clustering phase besides a non-linear GMM fitting phase per hidden state.

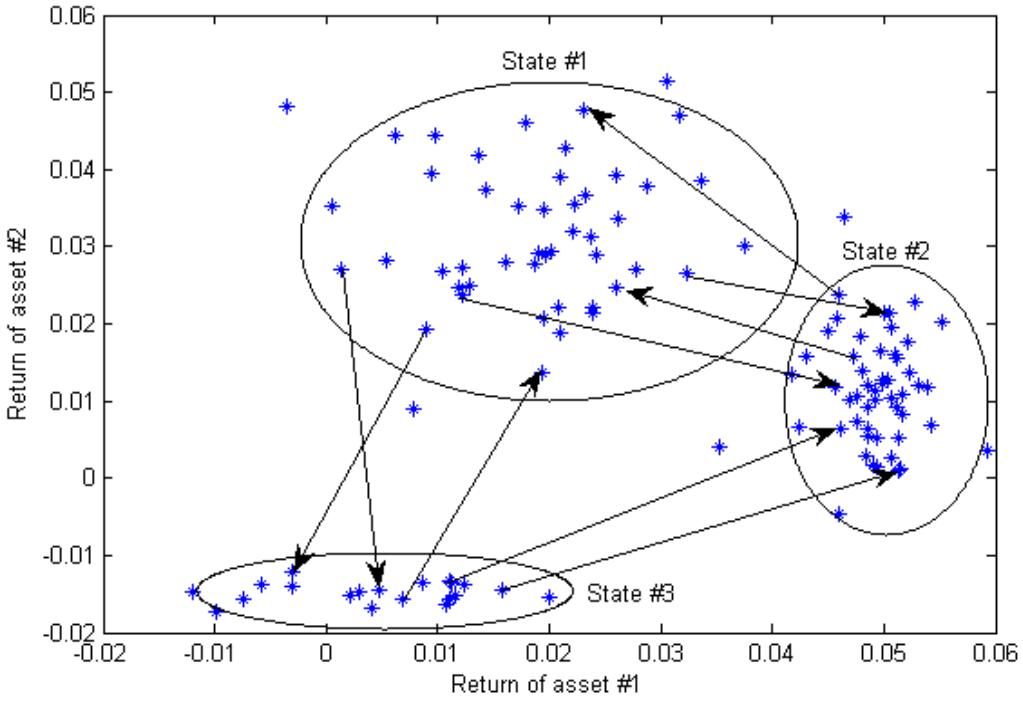

Figure 3-5: Clustering by GMM and transitions between the identified hidden states

This drawback motivated me to develop the following algorithm (depicted in Fig. 3-5):

1. Fit a Gaussian mixture distribution using every data point in the training set.
2. Assign each mixture component to a corresponding hidden state, index the price vectors accordingly.
3. Calculate the transition probabilities and the initial distribution vector from relative frequencies (note that the remaining model parameters, namely the p.d.f. of the observations, are already estimated during the first step).

As data points having similar distribution can differ in this aspect, the performance of the algorithm can be further enhanced by also taking into consideration the price vector in the next time instance besides the current one during clustering. In this case we need to perform the Gaussian mixture estimation (step 1.) on the following data set:

$$\tau = \{\{\mathbf{r}_i, \mathbf{r}_{i+1}\}, 0 \leq i < K\}, \tag{3.11}$$

which embed the return vectors into a $2n$ dimensional space. After the clustering is done, in step 2, we need to consider only the top-left quarter of the matrices describing the p.d.f., which belongs to the first n dimensions.

Dimension reduction by PPCA algorithm

Transforming the data from a higher dimensional space into a space of fewer dimensions as a pre-processing step for the HMM yields the following advantages:

- optimization in a smaller dimensional space requires much less computational time, which enables one to do higher frequency trading, or to involve a larger number of assets, which was not feasible formerly;
- one can control the model degree which is vital to avoid overfitting.

The goal of dimension reduction is to relate a d-dimensional return vector ($\mathbf{r}$) into a q-dimensional ($q < d$) latent vector denoted by $\mathbf{x}$. If we assume a linear relationship with non-zero mean, we get the basic model of factor analysis [Bartholomew, 1987]:

$$\mathbf{r} = \mathbf{W}\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}, \tag{3.12}$$

where matrix $\mathbf{W}_{d\times q}$ describes the transformation between the two spaces, $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\varepsilon}$ represents the noise in the model. Having the mapping matrix and the mean of the data, the reduction can be formulated as

$$\mathbf{x} = (\mathbf{r} - \boldsymbol{\mu}) \left(\mathbf{W}^T\right)^{-1}. \tag{3.13}$$

Deploying a feature extraction algorithm into to computational framework introduced in Fig. 3-1 can be done in a straightforward manner. As it is shown in figure 3-6, the dimensional reduction (as in (3.13)) and the back-projection to the original space (given in (3.12)) is simply precedes and follows the training and prediction modules, while the trading strategy is the same as before.

Probabilistic PCA (PPCA) is a derivation of principal component analysis (PCA),

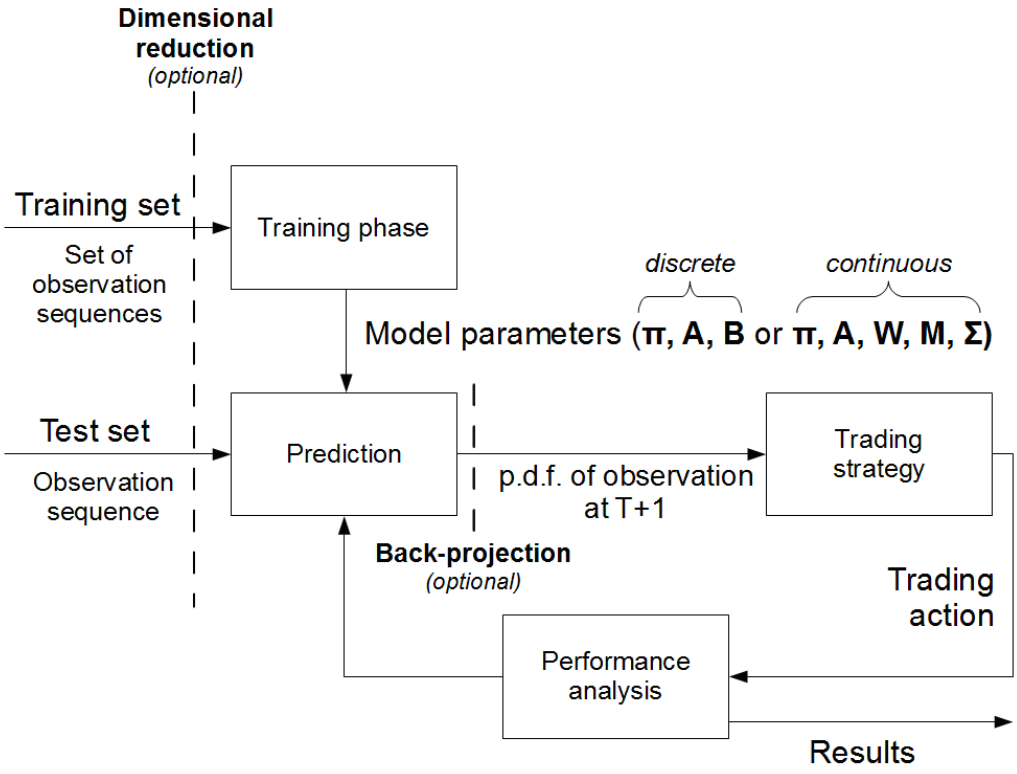

Figure 3-6: Extended computational approach for trading by HMMs

a well-known technique for dimension reduction [Jolliffe, 1986], having a proper density-estimation framework [Tipping et al, 1999]. PPCA model is described by the following conditional probability distribution:

$$\mathbf{r}|\mathbf{x} \sim N\left(\mathbf{W}\mathbf{x} + \mu, \sigma^2\mathbf{I}\right), \tag{3.14}$$

where the marginal distribution for the latent variables is Gaussian by convention with $\mathbf{x} \sim N(0, \mathbf{I})$ causing the marginal distribution for the observed data to be

$$\mathbf{r} \sim N\left(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}\right). \tag{3.15}$$

Due to lack of a closed-form solution, an iterative expectation-maximization algorithm was used to obtain the maximum-likelihood estimator of the model parameters [Tipping et al, 1999].

3.3 Prediction based trading

After the training phase has been completed, the model is able to predict future stock prices based on a window of observed previous data denoted by $\mathbf{X}$. The forward algorithm [Rabiner, 1989] can be used to calculate the forward values, i.e. the conditional probabilities being in each hidden state:

$$\alpha_t(i) = P(\mathbf{X}|\Theta, q(t) = q_i). \tag{3.16}$$

The forward algorithm calculates these values in a memory- and running time efficient manner by using a dynamic programming table. In our case, we are interested only in the probability vector belonging to the last state: $\alpha = \left[\alpha_T^{(1)}, ..., \alpha_T^{(N)}\right]$. Having this at hand, one can formulate the probability density functions belonging to observations in the next step as follows:

- discrete case: $\omega = \alpha\mathbf{AB}$, where $\omega_i = P(o_{T+1} = k_i)$;

- continuous case:

$$\xi \sim N\left(\sum_{i=1}^{N}(\alpha\mathbf{A})_i \sum_{j=1}^{M} w_{ij}\boldsymbol{\mu}_{ij}, \sum_{i=1}^{N}(\alpha\mathbf{A})_i \sum_{j=1}^{M} w_{ij}\boldsymbol{\Sigma}_{ij}\right) \tag{3.17}$$

  and $P(\xi = \mathbf{y}) = P(\mathbf{o}_{T+1} = \mathbf{y})$.

These p.d.f.-s are our predictions for the future asset prices. In each step, the asset with the highest probability of increasing (or decreasing) is selected which determines the trading action, whether to buy or short sell for the sake of maximizing the profit. In the discrete case, the trading action is determined by the movement coded by different states, we can de-quantize it as $E(\xi) = \omega^T \mathbf{v}$ accordingly, where $\mathbf{v}$ denotes the quantization vector used to map the alphabet (in the case of vector data, represented as n-tuples, an additional decoding step is needed to sum up the probabilities belonging to each asset). While, in the continuous case, a de-normalization step is needed as per (3.8). Thus, the trading signal is described as:

$$i := \arg\max_i |E(\xi_i)|\,; \quad \begin{array}{c} E(\xi_i) > 0 \rightarrow buy \\ E(\xi_i) < 0 \rightarrow shortsell \end{array}. \tag{3.18}$$

3.4 Performance analysis

The performance of the methods described above has been tested accordingly to the measures introduced in 2.5. The bar charts (Fig. 3-7 and Fig. 3-8) show that in the case of the U.S. SWAP rates all of the introduced methods beat the market tendency (-39.27%), and in the scenario when SA was used to train the continuous mode HMM the trading was profitable with a 108.52% yearly profit. Regarding the FOREX rates, our methods achieved up to 26.62% profit (Fig. 3-8).

During the training phase, an EM step takes around 900ms, while one step of SA consumes 250ms using the continuous training sets on an Intel M330@2.13GHz processor. The prediction based trading step is quasi real-time.

As one can see, the novel HMM optimization methods outperform the traditional EM algorithm in most scenarios, while the hybrid approach speeds up the convergence

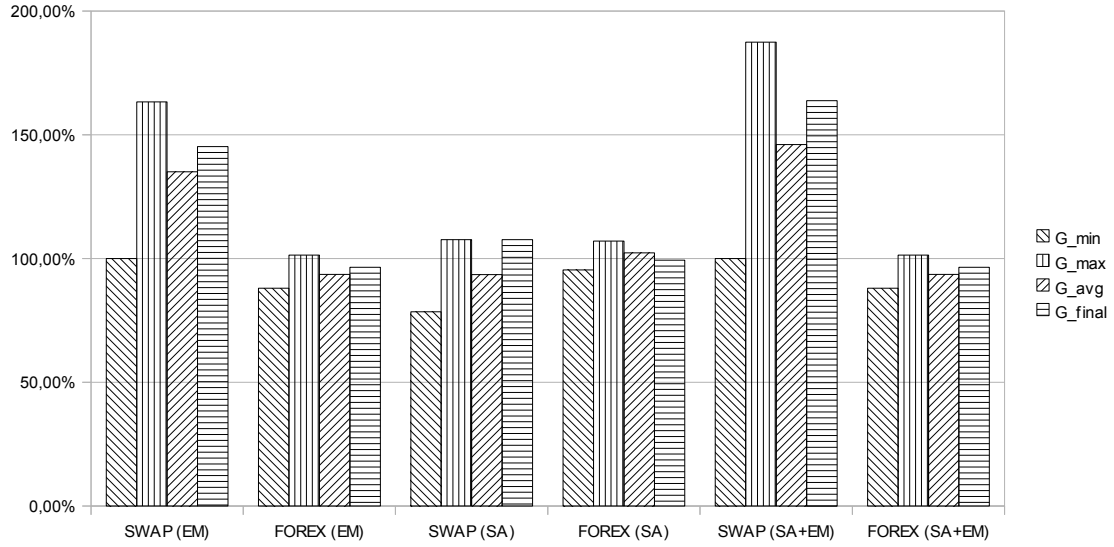

Figure 3-7: HMM trading results in discrete case

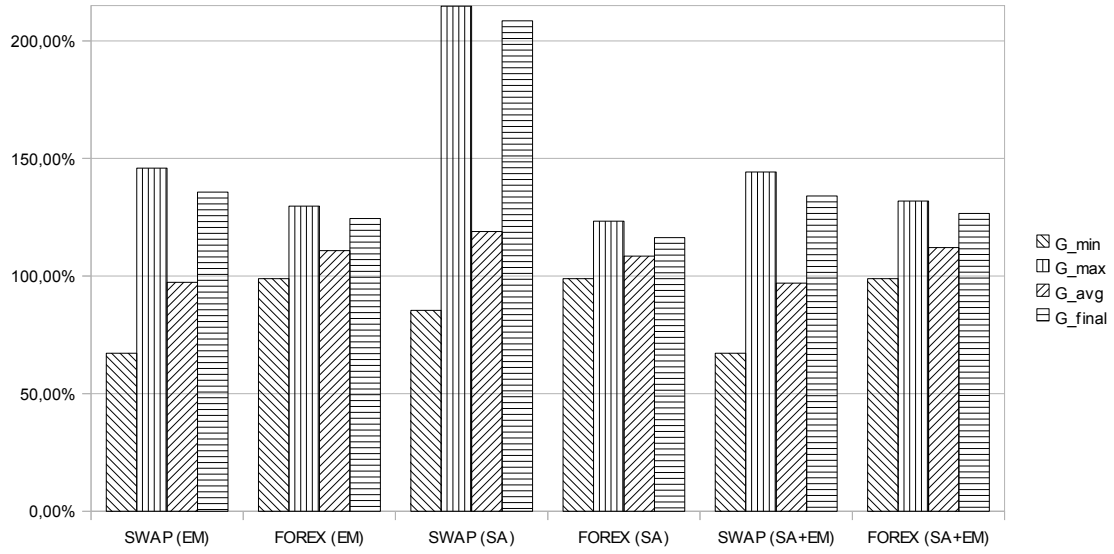

Figure 3-8: HMM trading results in continuous case

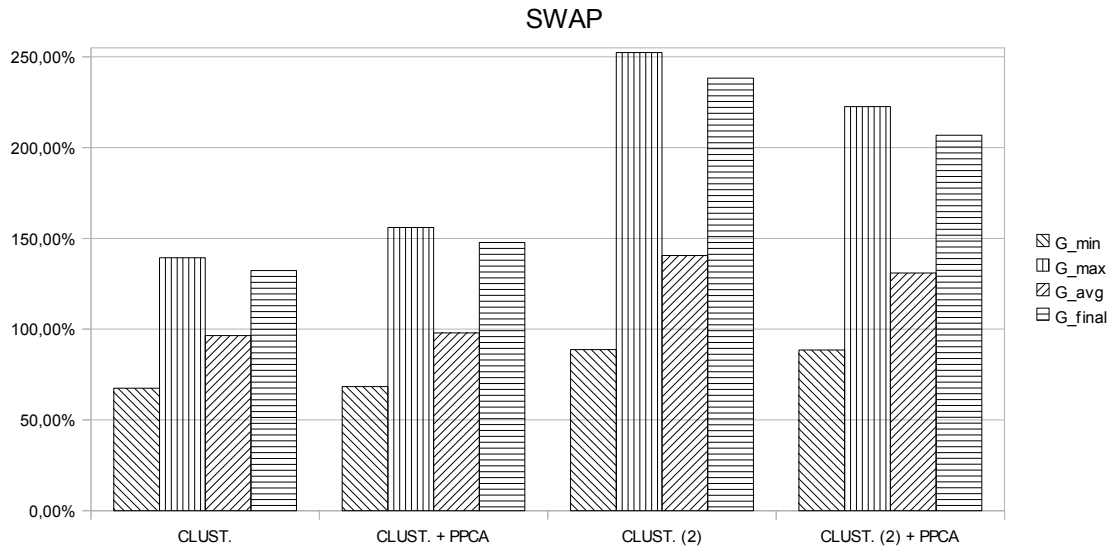

Figure 3-9: HMM trading results on SWAP using clustering and PPCA

with one order of magnitude.

As it is shown in figure 3-9 and 3-10, clustering and dimension reduction methods were also proven to be profitable on the tested time series, outperforming the uniform portfolio. The numerical results show that applying PPCA is beneficial with normal clustering (`CLUST`), and less profitable with the enhanced clustering version (`CLUST (2)`).

In comparison with the results achieved without pre-processing, the range of the realized profits are comparable, reaching up to total 138.28% yearly profit on SWAP. Nevertheless, the applied pre-processing steps resulted in a considerable speed-up (another order of magnitude) compared to the previous methods.

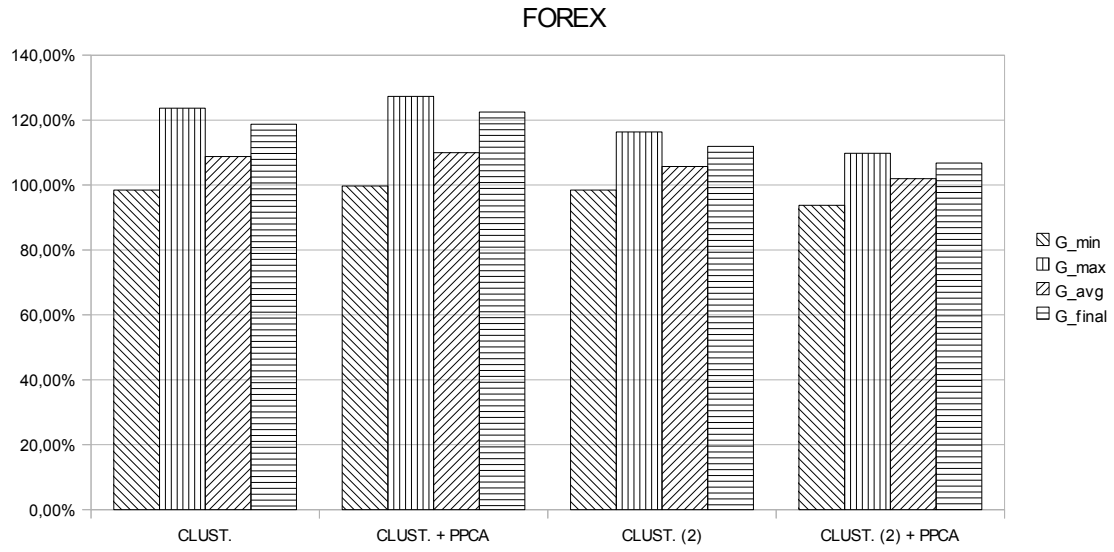

Figure 3-10: HMM trading results on FOREX using clustering and PPCA

Chapter 4

Optimizing sparse portfolios by AR-HMMs

There are several ways of modelling mean reverting stochastic processes, but the traditional methods fall short of capturing the true characteristics of the underlying time series. Therefore, we need to introduce more sophisticated modelling tools which can approximate a wide range of random processes. In this way, trading can become even more efficient and profitable. The modelling to achieve this goal is going to be the autoregressive hidden Markov model (AR-HMM).

As I demonstrated in 2.1.2, linear regression based methods proved to be sufficient for OU parameter estimation. However, the shortcomings of these methods are that they sometimes do not provide high accuracy for the purpose of algorithmic trading, and also they may be unstable under certain circumstances (e.g. when the level of mean reversion is low).

In the forthcoming discussion I will demonstrate that the AR-HMM process is a generalization of mean reverting processes, as a result AR-HMM can totally model mean reverting processes.

More precisely, the new contributions of this chapter lie in the following points: (i) instead of the predictability parameter the average return is maximized; (ii) instead of the OU modelling, autoregressive HMM (AR-HMM) is used.

In addition to that, a multi state AR-HMM can be considered as a generalization

of these PDFs, which enables to perform prediction based algorithmic trading on financial time series. The performance analysis shows that the results exhibit good average returns.

As a type of mixture models, endowed enough degree of freedom (number of hidden states) HMMs are capable of modelling a large class of distributions. Moreover, by AR-HMM we can capture both the long and short range dependencies of time series, as it combines a Markov chain on the hidden states, and giving rise to statistical dependencies on the observed variables [Berchtold, 1999]. Unlike the standard HMM assumption, in the case of AR-HMMs the emissions are conditionally not independent given the hidden state [Murphy et al, 2012]. The observable output was treated as a continuous value, described by a univariate Gaussian probability density function. Then the probability of emitting a specific output is determined by the conditional probability

$$P\left(p\left(t\right)|p\left(t-1\right),q_t=j,\Theta\right)=N\left(p\left(t\right)|\varphi_j p\left(t-1\right)+\mu_j,\sigma_j\right). \tag{4.1}$$

In other words, the observation depends on the hidden state, and on the previous observation through an additive autoregressive component.

In order to identify the model parameters

$$\Theta=\{\boldsymbol{\pi},\mathbf{A},\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\varphi}\} \tag{4.2}$$

the Baum-Welch expectation maximization algorithm can be utilized similarly as introduced in 3.2. Compared to the standard HMM, in the case of an AR-HMM, the forward-backward algorithm remains unchanged, while in the EM algorithm only a slight modification needed [Dey et al, 1994].

I have shown that we can consider the AR-HMM as a time dependent generalization of the OU and Brownian motion (BM) SDEs. Apart from the mean reverting modelling it is also a common approach in the literature to model financial time series with BM [Taylor, 2014]. However, that approach assumes the efficient-market hypothesis, which in reality might not hold. The novel approach introduced manages

to overcome the problems arising from the fact that the traditional OU approach is unable to handle processes changing their characteristic over time. Besides providing a more general model, it gives more accurate and stable predictions for the future portfolio prices.

4.1 Modelling OU processes with AR-HMMs

From the OU process, defined in (2.5), one can obtain the following probabilistic model [Gillespie, 1996]:

$$P\left(p\left(t\right)|p\left(t-1\right),\vartheta,\mu,\sigma\right)=N\left(p\left(t\right)\left|e^{-\vartheta\Delta t}p\left(t-1\right)+\left(1-e^{-\vartheta\Delta t}\right)\mu,\sigma\sqrt{\tfrac{1-e^{-2\vartheta\Delta t}}{2\vartheta}}\right.\right). \tag{4.3}$$

This equation describes the OU process in the same form as (4.1), where the conditional observation probability of an AR-HMM is defined. By carrying out the following mappings

$$\varphi_1=e^{-\vartheta\Delta t},\quad \mu_1=\left(1-e^{-\vartheta\Delta t}\right)\mu\quad\text{and}\quad \sigma_1=\sigma\sqrt{\frac{1-e^{-2\vartheta\Delta t}}{2\vartheta}}, \tag{4.4}$$

we can consider a single state AR-HMM being equivalent with an OU process. However, this can be either a mean reverting or a mean averting process. Furthermore, setting $\vartheta=0$ in (2.5) yields a BM, more generally, an AR-HMM state can represent BM with a possible linear drift as well.

Based on this approach, a multi state AR-HMM can be considered as a generalization of the OU and BM SDEs. Furthermore, having multiple states brings even more flexibility into modelling the price distributions, and also captures if the process is driven by different models over different intervals in time (often called as regimes [Hamilton et al, 1994]).

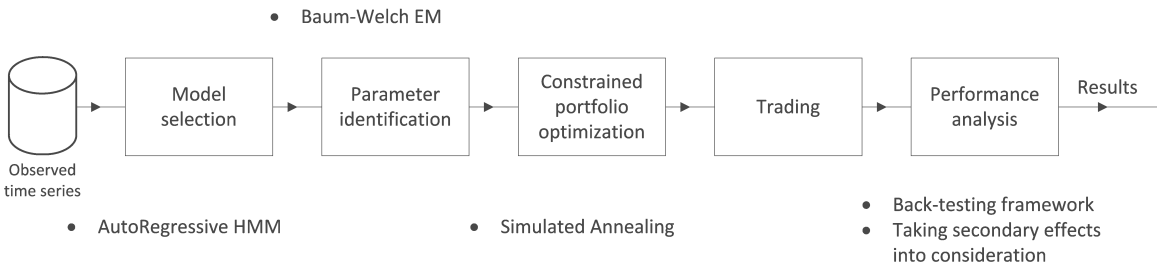

Figure 4-1: Computational approach — AR-HMM

4.2 Portfolio optimization

I adapted the objective function introduced in 2.2.2 to maximize the expected return for the case of modelling by AR-HMMs. In addition, I extended it to cope with the bid-ask spread. Similarly to (2.38), the objective function is expressed as follows:

$$\Psi(\mathbf{x}) = \max_{0 \leq t} E(p(t)) - p(0), \tag{4.5}$$

but in this case

$$E(p(t)) = (E(p(t-1))\,\boldsymbol{\varphi} + \boldsymbol{\mu})^T \gamma_t \tag{4.6}$$

and $\gamma_t = \mathbf{A}^T \gamma_{t-1}$ is described recursively, where $E(p(0)) = p(0) = \mathbf{x}^T \mathbf{s}_0$ and $\gamma_t^{(j)} = P(q_t = j \,|\mathbf{X})$.

First, we need to identify the AR-HMM model parameters for an observed process, and then, as a novel approach, optimize the prediction based average return (Fig. 4-1).

The bid-ask spread can be included into this model in a straightforward manner: the current and the future prices, for fitting the AR-HMM, of the portfolio should be taken into consideration as the corresponding bid and ask prices.

4.3 Performance analysis

In this section a performance analysis is given for AR-HMM based methods. First, the accuracy of an AR-HMM as an estimator of the long term mean is compared to the traditional methods for mean reverting processes (as described in 2.1.2). Second,

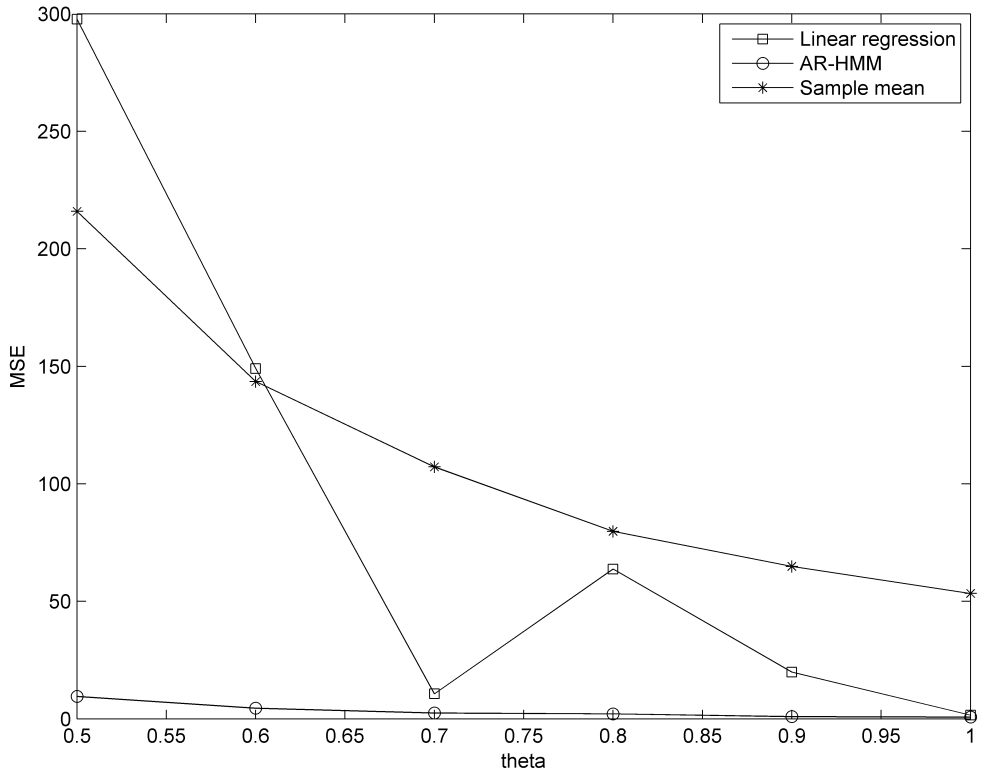

Figure 4-2: Comparison of different mean estimation techniques

AR-HMM based trading results are detailed on real financial data sets.

4.3.1 Mean reversion parameter estimation by AR-HMM

As it was detailed in 4.1, the single state AR-HMM can be treated as a generalization of the OU process, hence it is suitable as a parameter estimation method. Estimating the long term mean (μ) of the process of portfolio valuations is instrumental for mean reverting trading. For the sake of comparison with other estimation procedures (see section 2.1.2), different methods were tested on artificially generated data.

As one can see (Fig. 4-2), the newly proposed way of OU parameter identification gives the most precise estimations, outperforming the traditional methods by an order of one magnitude.

At the same time, using the OU parameter estimation techniques for identifying AR-HMM model parameters is not directly possible due to its more general nature.

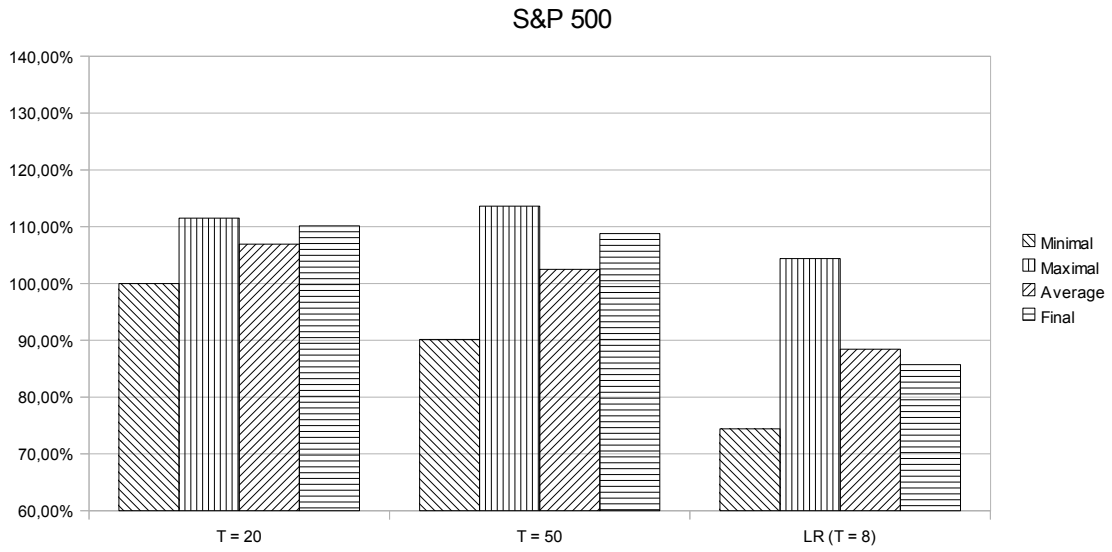

Figure 4-3: AR-HMM trading results on S&P 500

In turn, initializing the BW algorithm with a special case of AR-HMMs representing a simple OU process that is fit to the observations might be beneficial in terms of its faster convergence. However, this investigation is left as an area of future research.

4.3.2 Trading results

Fig. 4-3 compares the results achieved on S&P 500 with different lengths of sliding window. As a benchmark, these results are compared to the performance achieved by the same objective function and trading algorithm, but using linear regression for parameter estimation instead of AR-HMM (denoted by `LR`, further details are given in section 2.1.2). Due to the large number of assets, $N = 7$ hidden states were used. Unfortunately, there is no analytical method known to find the optimal value of this parameter [Hassan et al, 2007].

The novel AR-HMM method proved to be profitable and outperformed the linear regression by 24.42%.

For back-testing on FOREX time series, we used the MetaTrader® 5 platform providing a real environment with secondary effects, like the bid-ask spread. During the simulations, the leverage was set to 10. During the simulations $T = 50$ days were used for model identification, while the effect of different number of hidden states in

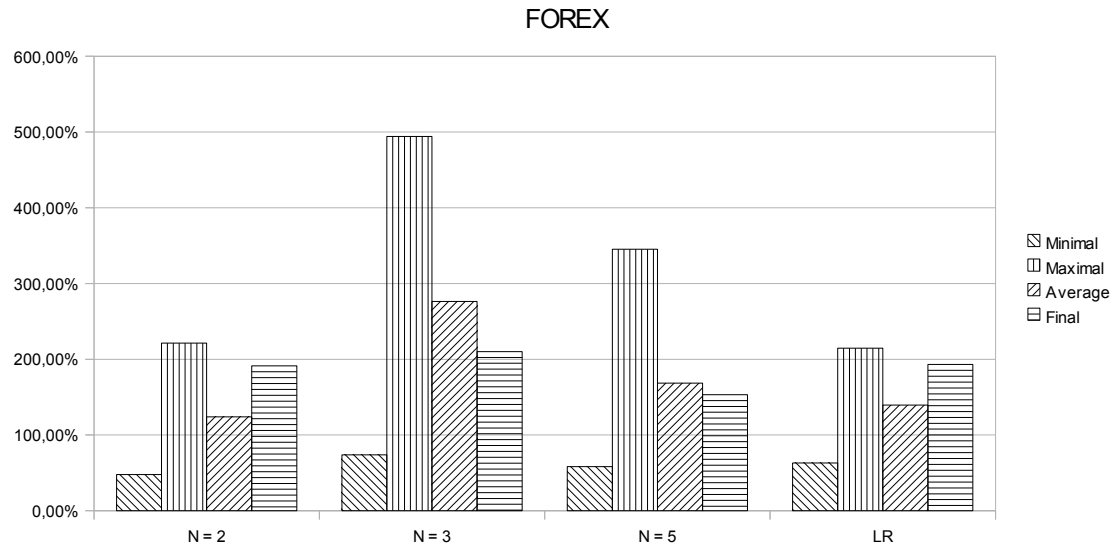

Figure 4-4: AR-HMM trading results on FOREX

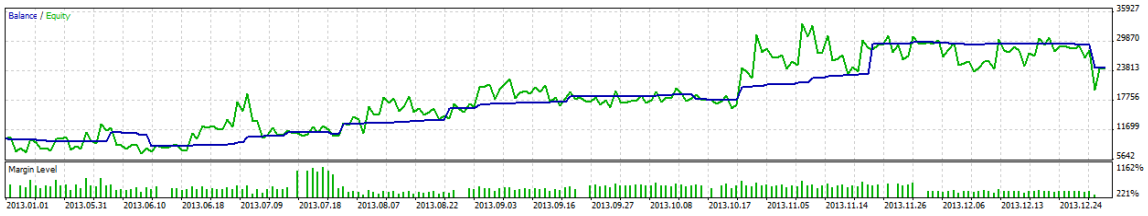

Figure 4-5: Detailed AR-HMM trading results on FOREX

the trading results is shown in Fig. 4-4.

In this period, the FOREX rates had a slightly decreasing tendency, an equally weighted portfolio would result in -2.24%. The bar chart (Fig. 4-4) shows that the introduced methods beat this tendency, they achieved up to 110.12% yearly profit.

In the scenario when a longer time window was used, the trading was even more profitable with a 144.10% yearly profit including every secondary effects. However, considering longer sets of data, the training of the AR-HMM requires substantially longer computational time.

As it is shown in figure 4-5, using this strategy not only results in a favourable profit, but without major drawdowns, the balance is almost monotonously increasing.

Chapter 5

Real-time parallel implementation on GPGPUs

Not only the profit but the computational time is also an important aspect in real environment. Reacting earlier than the competitors on the market, and being able to participate in more trading opportunities establish its direct impact on the profitability. Along with the algorithmic approaches developed for fast trading, the software implementation on massively parallel architectures can also be used to decrease the running time and ensure a real-time algorithmic trading. This paves the way towards high-frequency trading, as well as trading on a larger number of assets (e.g. S&P 500). Furthermore, it helps perform a deeper analysis providing better quality results. Note that in the case of the applied stochastic search algorithms (e.g. SA), the response time can be scaled in an arbitrary way by setting the number of steps. Naturally, this way the quality of the solution is going to be compromised.

The nature of the methods and algorithms introduced in the previous chapter for AR-HMM based trading, including linear algebraic operations, simulated annealing and the forward-backward algorithm for HMMs allows parallel implementation and requires lesser data transfer for the input and output variables. These algorithmic properties make a massively parallel implementation feasible and beneficial. In this light, General-Purpose computing on Graphics Processing Units (GPGPU) must be further investigated for trading.

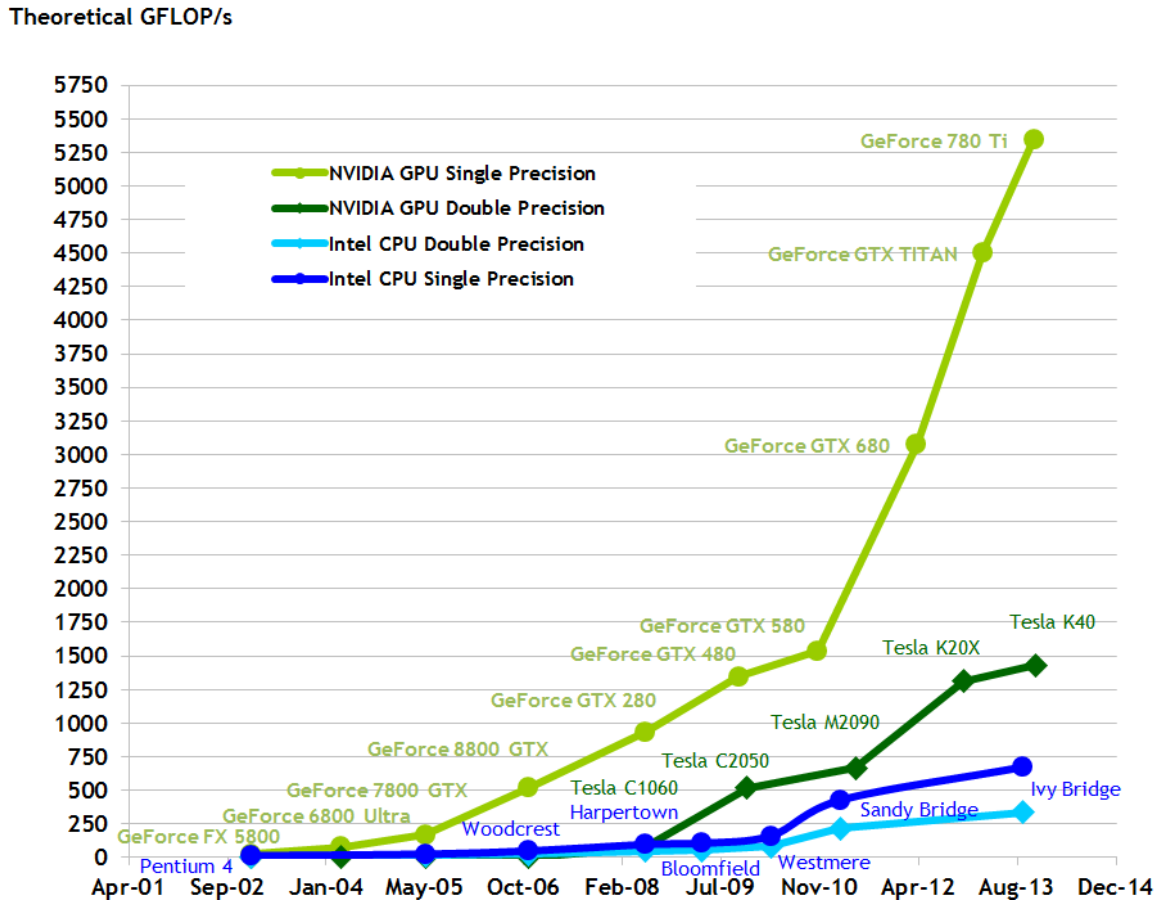

Figure 5-1: Comparison of peak floating-point performance of Nvidia GPUs and Intel CPUs [NVIDIA, 2014]

GPGPU is a computing concept which utilizes the massively parallel architecture of a Graphics Processing Unit (GPU) to solve arbitrary numerical problems (NVIDIA [2014] and Cook [2013]). In terms of floating-point operations per second, which is the primary measure of hardware performance for scientific or engineering computations, GPUs are superior compared to traditional CPUs (Fig. 5-1).

On the other hand, GPU based implementations are inadequate when dealing with serial algorithms, or when a significant amount of data need to be transferred between the CPU and GPU memory space.

GPUs are also outperforming the CPUs in power consumption per floating-point operation (Fig. 5-2), hence if energy efficiency is a concern then this approach proves to be favourable.

Consequently, using the GPU shows growing interest in the field of computational

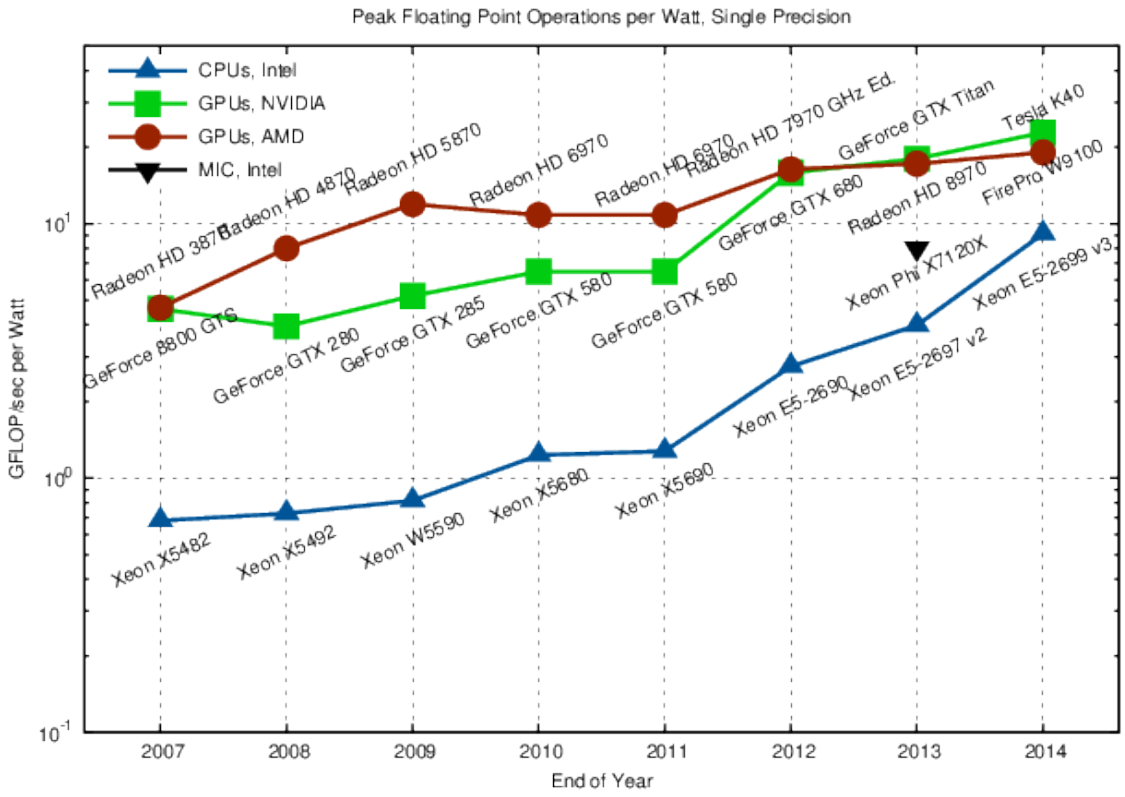

Figure 5-2: Comparison of energy consumption per floating-point operation [Rupp, 2013]

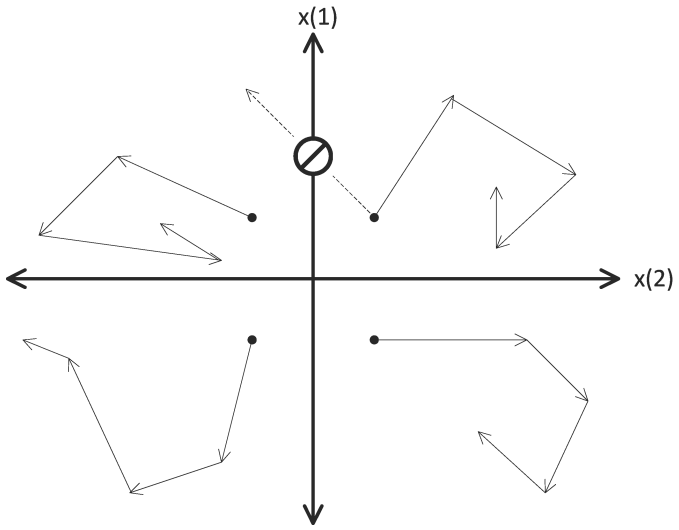

Figure 5-3: Parallel simulated annealing

finance also [Pages et al, 2011]. There are several applications that clearly proven the success of highly parallel programming, such as option pricing, risk calculation and time series prediction. Therefore, the challenge for algorithmic trading is to develop methods implemented in parallel. Its computational power here is going to be used for time series modelling, identification and portfolio optimization.

After identifying the AR-HMM model parameters on the given time series, the optimal portfolio selection is carried out in a similar manner as in 4. However, by taking advantage of the higher computational power enabling Monte Carlo methods, the objective function introduced in (4.5) can also be generalized. Instead of the expected return, the complete probability density function of the future portfolio evaluations is modelled by the AR-HMM fit to the time series as detailed in section 4.1. The portfolio optimization itself was driven by SA. Motivated by the high number of possible parallel threads the SA was extended to cope with multiple portfolios simultaneously as specified in 5.2 (Fig. 5-3).

Besides the new algorithmic approaches to utilize the GPGPU environment the methods introduced in chapter 4 have also been ported to this architecture (see section 5.2). As a result, in the same time span a larger population of portfolio candidates can be evaluated based on a more precise objective function compared to the traditional implementation. This improvement leads to higher profits and also casts trading

feasible on higher frequency data as demonstrated in section 5.3.

5.1 Portfolio optimization

In this section I discuss the optimal portfolio selection subject to a new objective function. This goal function maximizes the lower bound on return achieved with a pre-defined probability η, i.e.:

$$\omega(t) : P(p(t) \geq \omega(t)) = \eta. \tag{5.1}$$

However, attention should be given when the future portfolio price distribution is asymmetric. Especially, in the case when its expected value implies loss on the portfolio, while at the same time the median ($\eta = 0.5$) would falsely prompt us to invest into this specific portfolio. In order to avoid such situations, a constraint is formed accordingly. As a result, the objective function is expressed as follows:

$$\begin{gathered} \Psi(\mathbf{x}) = \max_{0<t} \omega(t) - p(0) \\ \text{under the constraint that } E(p(t)) - p(0) > 0 \end{gathered} \tag{5.2}$$

Taking into account the cardinality constraint as well, portfolio optimization can be reduced to a constrained optimization problem:

$$\mathbf{x}_{\text{opt}} = \arg\max_{\mathbf{x}} \Psi(\mathbf{x}), card(\mathbf{x}) \leq l. \tag{5.3}$$

This objective function may prove to be much more beneficial for the portfolio holder than just optimizing the predictability (as in (2.37)) or maximizing the average return (introduced in (2.44)), however, its solution does not lend itself to analytical tractability (as opposed to the maximal predictability which leads to a generalized eigenvalue problem shown in (2.36)). Thus, to come to grip with maximizing $\omega(t)$ for a given portfolio, we identify AR-HMM model parameters for an observed process, and then, as a novel approach, we calculate the prediction based CCDF.

To obtain the CCDF of the future values of $p(t)$, we have used Monte Carlo method. K simulations were run on the identified AR-HMM starting from the observed initial value of $p(0) = \mathbf{x}^T \mathbf{s}_0$ in the $0 < t \leq T_{\max}$ interval. Section 5.2 describes methods for estimating $\Theta = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ and γ_0.

Parameter η allows the investors to adjust their willingness to take risk. One can see that $\eta < 0.5$ results in a braver strategy, while $\eta > 0.5$ yields a rather cautious one. The bid-ask spread can be included to the model alike as in 4.2.

5.2 Parallel implementation

In this section we investigate the implementation of AR-HMM based portfolio optimization on parallel architectures. This implementation may pave the way towards a more sophisticated high frequency trading using AR-HMMs on GPGPU. However, there are some algorithmic challenges to be resolved when implementing the portfolio optimization on multi-core architectures.

The initial parameter estimations of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in (4.2) are based on the K-means clustering algorithm [Bishop, 1995]. The input of K-means algorithm consists the time series for each portfolios and the outputs are the cluster centres and variances. By giving multiple initial points, more precisely giving the time series values in different orders, for the K-means algorithm we might obtain different HMMs for a given portfolio, because K-means might converge into different state distributions. This algorithm is used only to create a reasonable starting point for local optimization, and the data points are simply treated as samples from the equilibrium distribution, hence their order is not taken into consideration.

After identifying the initial state distribution, we use the Baum Welch algorithm (as described in chapter 4) to estimate the final states, together with the autoregressive $\boldsymbol{\varphi}$-s, and transition matrix per HMM. Training HMMs by the traditional Baum-Welch algorithm on a single core architecture is straightforward. It uses an Expectation-Maximization algorithm to find the model parameters for a given set of observation. The computationally most time consuming step of this process is the

forward-backward algorithm.

In the parallel implementation, the number of parallel threads are the same as the number of states, on the other hand, updating the probability distribution vector is still to be done sequentially. As a result, the degree of parallelism is only determined by the number of states for a given HMM. This can be further extended by running several HMMs simultaneously. As it was mentioned before, one can initialize the K-means algorithm and, as a result, the BW algorithms in multiple ways. Training more than one HMM per portfolio yields a better fit model, if the one with the highest likelihood (see equation (3.9)) is chosen. In this way, the Baum Welch algorithm can be performed independently on N (number of HMMs per portfolio times the number of hidden states in each portfolio) number of threads for each portfolio. In turn, the number of states is chosen less than 32 because a further increase will not considerably improve the trading performance.

The conventional implementation of Baum-Welch and K-means algorithm run until they meet a given stopping criteria (e.g. convergence detected or the objective function falls below a given threshold). Due to the nature of parallel computing, we use a fixed number of iteration, because the runtime is determined by the slowest thread.

Regarding the portfolio optimization, we perform parallel and independent simulated annealing in a given number of randomly selected subspaces. Each subspace is l dimensional, to fulfill the cardinality constraint $card(\mathbf{x}) \leq l$. More precisely, we used closed orthants in $\mathbf{R}^l$, constraining each coordinate of the subspace to be nonnegative or nonpositive. This approach further increases the level of parallelism, resulting N threads per subspace in total. Depending on the properties of the actual GPU used for the calculations a so called block can be formed from one or more of these threads to be scheduled to the multiprocessors.

As an initialization we generate some portfolios (each of them satisfies the cardinality constraint), and for each portfolios we fit at least one HMM. Then we let the parallel optimization algorithm run until the temperature reduces to zero. After reaching the predefined iteration number, we pick the best portfolio with respect to

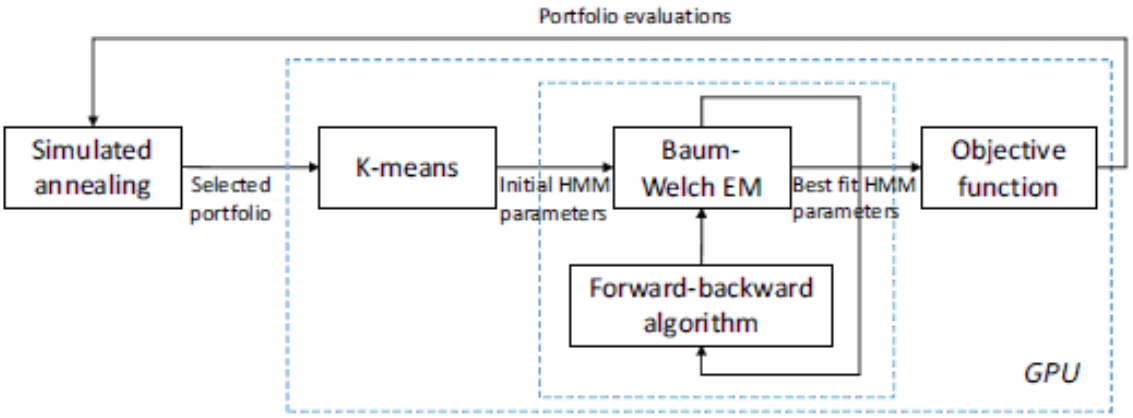

Figure 5-4: GPGPU computational approach

the objective function.

The parallelism of the proposed method is twofold with respect to granularity:

- Coarse-grained parallelism: as opposed to the sequential SA in which in each run we start from a given initial condition, in the case of our parallel implemented SA several algorithms run from several initial points at the same time, i.e. AR-HMM parameter fittings in each subspace are done simultaneously.

- Fine-grained parallelism: in each step of SA, (i) the algorithm for AR-HMM parameter estimation is parallel along the hidden states, and (ii) the evaluation of the objective function through the Monte Carlo simulations for estimating the complementary cumulative distribution function is also carried out in a parallel manner.

Accordingly, the computational framework is shown by the block diagram Fig. 5-4.

5.3 Performance analysis

Similarly to the previous chapters, an extensive back-testing framework was used to provide numerical results for the sake of comparing the performance of CPU and the GPGPU. In this section, we show the numerical results obtained on the following data sets:

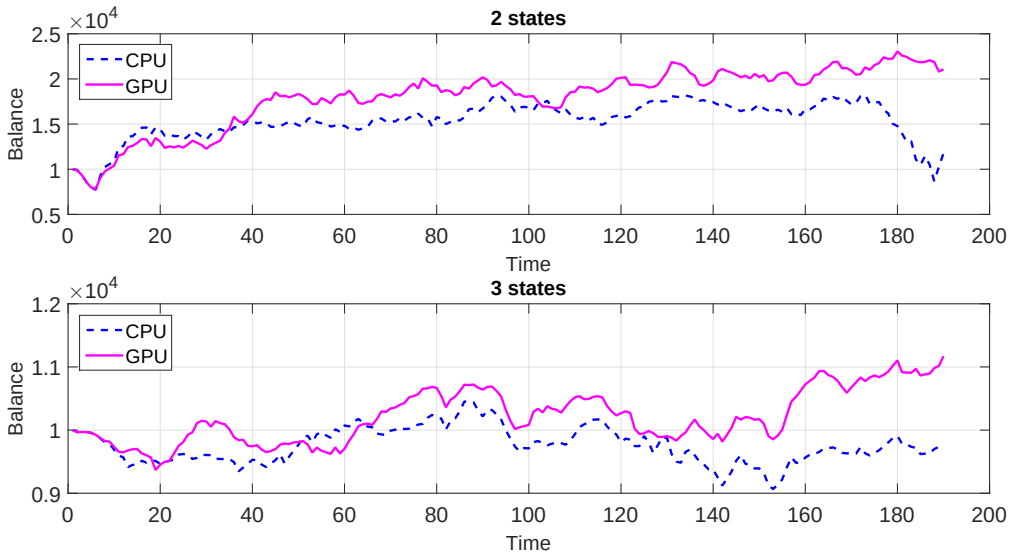

Figure 5-5: Trading performance on generated data

- AR-HMM generated time series;
- FOREX rates (EUR/USD, GBP/USD, AUD/USD, NZD/USD, USD/CHF, USD/CAD from the year of 2013 in daily and 15 minutes resolution) [Metatrader, 2014].

Regarding the sparsity constraint, 3 assets were selected in each transaction. The artificial data was generated by on AR-HMM with two and three hidden states, respectively. The numerical experiments lasted the same time interval on both architectures. In each case 3 hidden states were used, we used 65 observations from the past to fit AR-HMMs, while the prediction was projected to the next 8 time instances.

5.3.1 Trading results

As the GPGPU is much faster and thus capable of visiting much more portfolios in the course of the optimization a much better value of the objective function has been achieved and the trading performance has surpassed the one achieved by the CPU on generated data (Fig. 5-5).

The next figures indicate real tests running on FOREX data obtained at 15 minute (Fig. 5-7) and daily (Fig. 5-6) tick time. The duration of tests have been 1 year and 3 days, respectively.

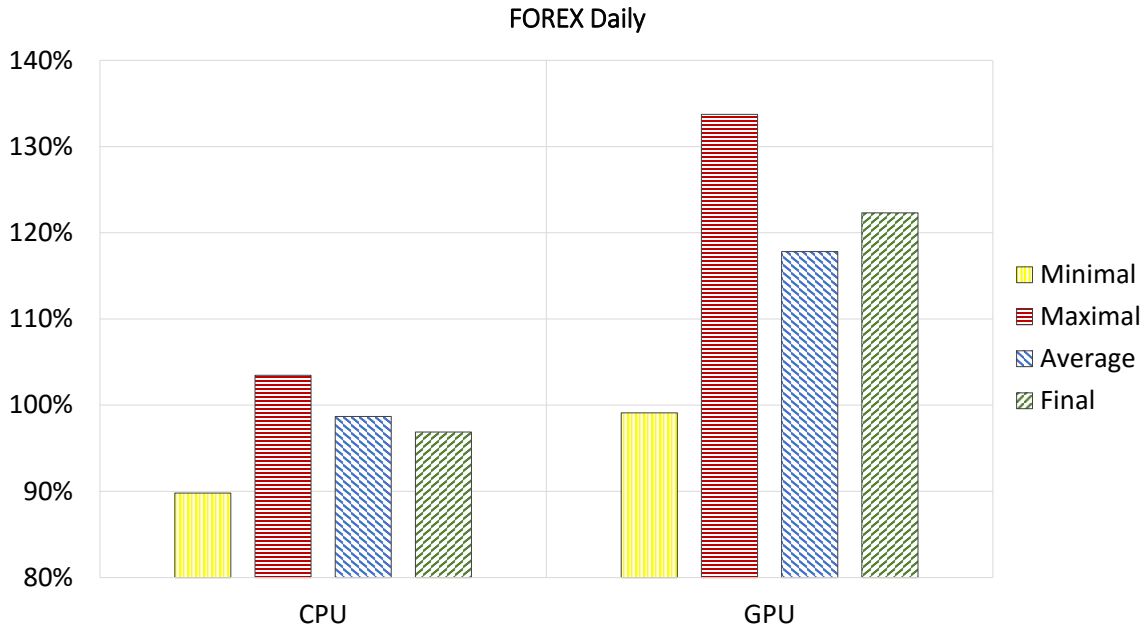

Figure 5-6: Trading performance on daily FOREX data

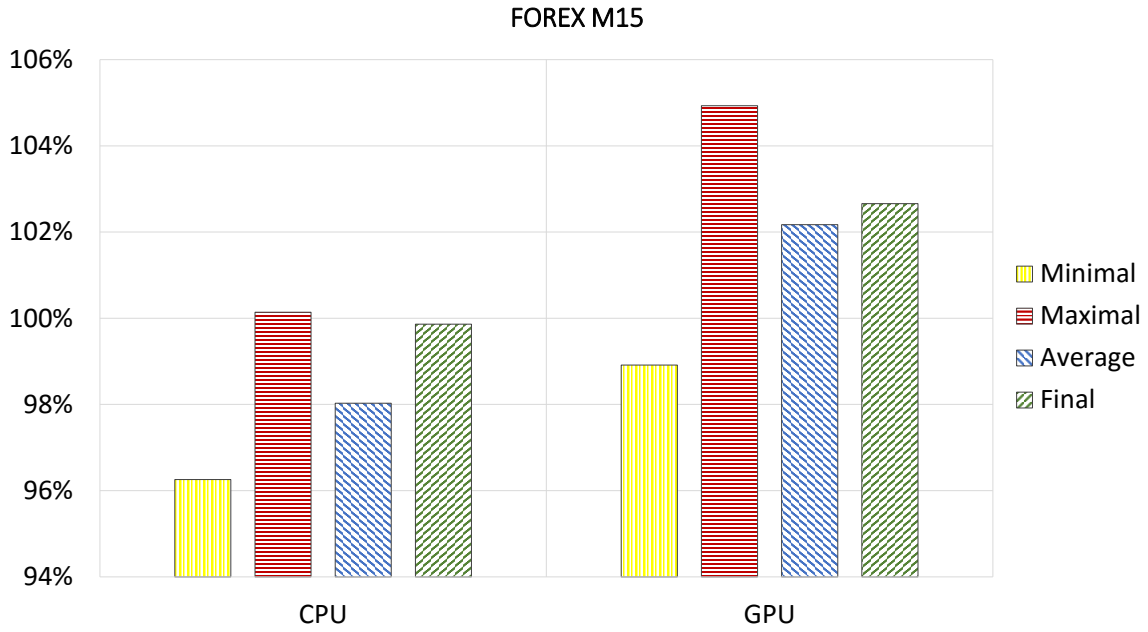

Figure 5-7: Trading performance on 15 minutes FOREX data

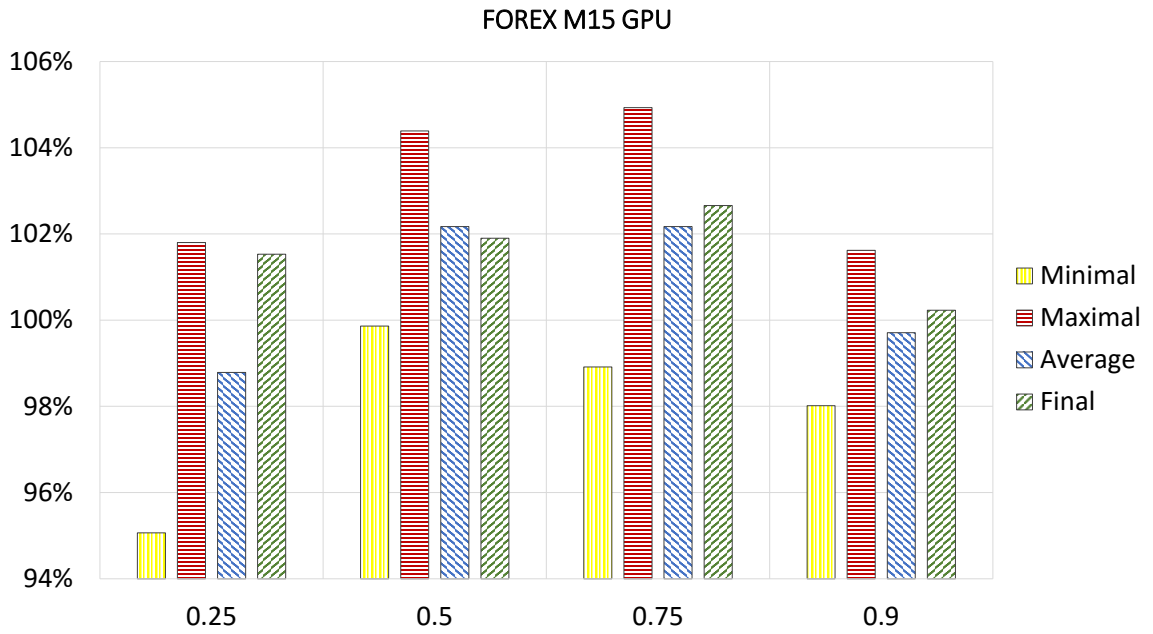

Figure 5-8: Comparison of trading results using different η on 15-minute FOREX data

One can see, in each case we managed to secure positive profit. On the 15 minute tick time the GPGPU has achieved 2.7% profit, while 0.1% loss achieved by CPU in a 3 days interval (annualization is often misleading on high-frequency data). On the daily averages the system performed more modestly, but even in this case the GPGPU secured a 22.3% yearly profit as opposed to the 3.1% loss achieved by the CPU. The reason for the poorer performance on daily averages may lie in the fact that in the case of more infrequent samples one cannot exploit the advantage of nature of the time series exhibited at a finer scale.

As demonstrated by Fig. 5-8, we also compared the performance when using different η parameters in (5.2), which controls the level of risk taking. As one can see, it proved to be favorable to follow a cautious strategy with $\eta = 0.75$, but not an overly cautious one.

5.3.2 Speed profiling

The experiments were conducted on a PC with a second generation Intel i7 CPU and on one NVIDIA GeForce GTX 570 GPU. The comparison between the computation

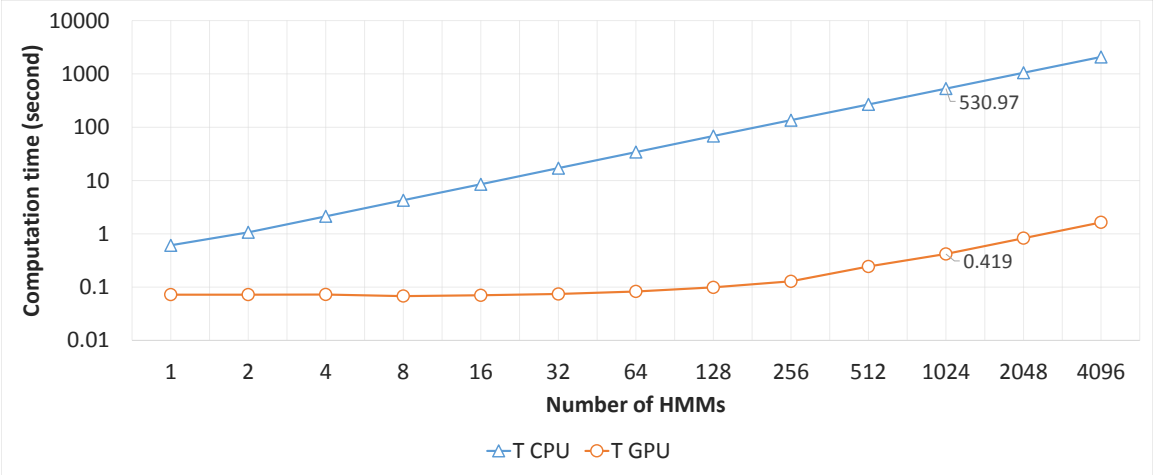

Figure 5-9: Comparison between the computation time of CPU and GPU

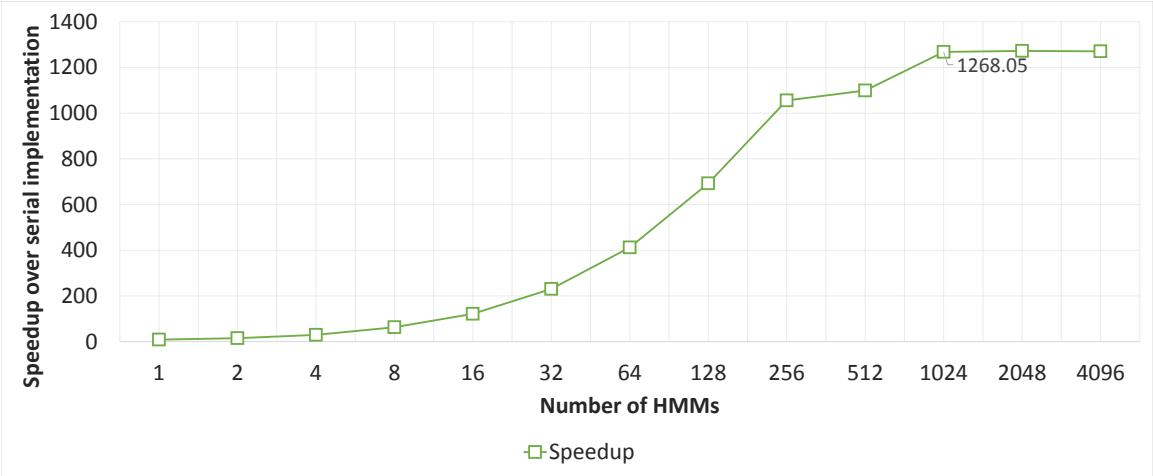

Figure 5-10: Speedup over serial implementation

time of CPU and GPU is shown by Fig. 5-9.

Fig. 5-10 shows the speedup over serial implementation, that is the ratio between the computation time on CPU and the computation time on GPGPU.

The performance in the case of small number states of HMM is relatively low, which is not surprising in the light of the fact the GPGPU computing power remains underutilized. If we increase the number of optimized HMMs then more and more blocks will become active on the GPGPU which will increase the performance, thus the GPU will be much faster than the CPU with a factor 1024 up to 1270. This increase in speed is imperative to perform high frequency trading.

To demonstrate the difference between CPU and GPGPU, in the M15 simulations

(as shown in Fig. 5-9) the portfolio optimization took 478 ms on average, which means about 10 minutes on CPU casting it unfeasible to use in this time scale.

Chapter 6

Conclusions and summary of results

The dissertation is focused on stochastic portfolio optimization, which is the core issue of algorithmic trading. As was mentioned before, the novel contributions are centred around the following topics:

- model selection (mean reverting portfolios; hidden Markov modelling; extending the underlying model to autoregressive HMMs);
- model parameter identification;
- optimization (SA; FFNN; Baum-Welch EM and a hybrid method);
- trading (massively parallel GPGPU implementation).

More precisely:

- I have developed novel approaches for mean reverting portfolio optimization subject to cardinality constraints, namely a new objective function — maximizing the mean return — and a corresponding trading algorithm have been introduced. The optimization itself is carried out by stochastic search algorithms and FeedForward Neural Networks (FFNNs). By this method a better profit can be achieved as compared to the traditional ones.
- I have increased the performance of prediction based algorithmic trading by fitting Hidden Markov Models (HMM) to the asset price series. To enhance its

performance, I have introduced a novel hybrid algorithm for learning. To further ease the underlying computational complexity, I have employed a new clustering algorithm and I have carried out dimensional reduction by probabilistic PCA (PPCA).

- I generalized the mean reversion based trading from the OU SDE to a more flexible modelling with autoregressive HMMs (AR-HMM) which includes the OU SDE. In this way, AR-HMM can better capture the stochastic nature of $p(t)$. I have introduced novel parameter estimation methods and a corresponding objective function.

- I provided a real-time GPGPU based parallel implementations for the described methods to further enhance the practical usability of results. This allows one to involve more assets into the analysis or perform an even higher frequency trading. Moreover, through the decreased power consumption, it reduces operating costs and makes the algorithmic trading more eco-friendly.

Table 6.1 summarizes the profit achieved by the novel methods trading on different assets. The final result is shown at the end of the year (proportionally to the original capita) achieved by the traditional methods, namely:

- MR: maximizing predictability;

- HMM: training by the Baum-Welch EM;

- AR-HMM: using linear regression,

besides the profit obtained by the novel algorithms. One can see that the new method performed better than the traditional one almost in each case.

The proposed trading algorithms has proven to be profitable on real financial time series taking into account the bid ask spread as well. An extensive performance analysis demonstrated that the trading based on the novel parameter identification algorithms and objective functions could increase the trading efficiency and the profit compared to the traditional methods. It is also shown that the efforts from both

algorithmic (e.g. clustering and PPCA, hybrid training algorithm for HMMs) and implementation (using GPGPUs) side can further speed up the algorithms giving rise to high frequency trading applications. Consequently, I have managed to improve the performance of algorithmic trading.

While the dissertation focuses on algorithmic trading, the results developed here can be generalized to any other framework where modelling and prediction of time series are concerned. For instance, HMMs are widely used in speech recognition and synthesis and bioinformatics (e.g. DNA motif discovery, protein folding).

Since algorithmic trading tends to be the backbone of modern financial industry, the novel methods can contribute to reliability of financial processes, as a result they can also be beneficial for the economy and the society, on the whole.

6.1 Future work

One could further improve the results by (i) extending the underlying modelling tools to more general models with less assumptions about the underlying time series; (ii) or introducing more sophisticated novel objective functions to get a better grasp at balancing between the expected profit and the associated risks; (iii) and utilizing novel portfolio optimization methods which can more efficiently find the optimal portfolio by taking into consideration the nature of the objective function.

Method	**FOREX**	**SWAP**	**S&P500**
Traditional MR (2.2.1)	104.86%	96.57%	133.40%
Novel MR (2.5)	121.66%	113.49%	153.55%
Traditional HMM (3.2)	124.47%	135.65%	-
Novel HMM (3.2)	126.62%	208.52%	-
Novel HMM (3.2.1)	122.49%	238.28%	-
Traditional AR-HMM (2.1.2)	193.02%	-	85.73%
Novel AR-HMM (4)	244.10%	-	110.15%
Traditional AR-HMM (HFT) on CPU (5)	100.3%	-	-
Novel AR-HMM (HFT) on GPGPU (5)	100.7%	-	-

Table 6.1: Comparison of profitability of novel methods compared to the traditional ones

The reliability of the results could be highly improved if the step-by-step nature of the general computational approach (as shown in Fig. 1-1) would be redesigned. This is motivated by the fact that the evaluation for a given portfolio based on the optimal (for example in terms of maximizing the likelihood) estimation of the model parameters can greatly differ from the one belonging to a slightly less probable parameter set. Similarly, in the introduced objective functions we assumed that we will know the exact time instance to sell the portfolio. Thus, I would suggest further investigation into the effects of incorporating the trading strategy into the objective function itself.

Chapter 7

Appendix

Throughout the performance analysis parts of the dissertation, the shown numerical results obtained on the following data sets:

- Daily closing prices of 500 stocks from the S&P 500 (between July 2010 and July 2011) [Yahoo, 2011]
  - Chapters 2 and 4.
- U.S. SWAP rates in daily resolution [Morgan, 2010]
  - Chapter 2: from the year of 1998
  - Chapter 3: between August 2008 and August 2010
- FOREX rates in daily resolution (EUR/USD, GBP/USD, AUD/USD, NZD/USD, USD/CHF, USD/CAD) [Metatrader, 2014]
  - Chapter 2: mid-prices between October 2005 and September 2006
  - Chapter 3: mid-prices between December 2009 and 2011
  - Chapters 4 and 5: bid and ask prices from the year of 2013

Bibliography

Publications of the author

Journal articles

SIPOS, I. Róbert; LEVENDOVSZKY, János. Optimizing sparse mean reverting portfolios. Algorithmic Finance, 2013, 2.2: 127-139. DOI: 10.3233/AF-13021

SIPOS, I. Róbert; LEVENDOVSZKY, János. Optimizing Sparse Mean Reverting Portfolios with AR-HMMs in the Presence of Secondary Effects. Periodica Polytechnica Electrical Engineering and Computer Science, 2015, 59(1), 1-8. DOI: 10.3311/PPee.7352

SIPOS, I. Róbert; CEFFER, Attila; LEVENDOVSZKY, János. Parallel optimization of sparse portfolios with AR-HMMs. Submitted to: Computational Economics.

Conferences

SIPOS, I. Róbert; LEVENDOVSZKY, János. High frequency trading with Hidden Markov Models by using clustering and PPCA algorithms. In: Proceedings, 15th Applied Stochastic Models and Data Analysis (ASMDA2013), pp. 67-80. Barcelona, 2013.

References

AVELLANEDA, M. Algorithmic and High-frequency trading: an overview. Quant Congress, USA, 2011.

BARTHOLOMEW, D. J. Latent variable models and factor analysis. Charles Griffin & Co. Ltd., London, 1987.

BAUM, L. E. and PETRIE, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. The Annals of Mathematical Statistics 37 (6): 1554–1563, 1966.

BAUM, L. E., PETRIE, T., SOULES, G. and WEISS, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist., vol. 41, no. 1, pp. 164–171, 1970.

BERCHTOLD, Andre. The double chain Markov model. Communications in Statistics-Theory and Methods, 1999, 28.11: 2569-2589.

BISHOP, C. M. Neural networks for pattern recognition. Oxford university press, 1995.

BOX, G.E., TIAO, G.C. A canonical analysis of multiple time series. Biometrika 64 (2), 355–365, 1977.

CHAN, E. Quantitative trading: how to build your own algorithmic trading business. Vol. 430. John Wiley & Sons, 2009.

COOK, S. CUDA programming: A Developer's Guide to Parallel Computing with GPUs. Elsevier, 2013.

CYBENKO, G. (1989) Approximations by superpositions of sigmoidal functions. Mathematics of Control, Signals, and Systems, 2 (4), pp. 303-314.

DASGUPTA, S. Learning Mixtures of Gaussians. Proceedings of Symposium on Foundations of Computer Science (FOCS), 1999.

D'ASPREMONT, A. (2011) Identifying small mean-reverting portfolios. Quantitative Finance, 11:3, pp. 351-364.

DEY, S., KRISHNAMURTHY, V., SALMON-LEGAGNEUR, T. Estimation of Markov-modulated time-series via EM algorithm. Signal Processing Letters, IEEE, 1994, 1.10: 153-155.

DOOB, J. L. (1953). Stochastic processes (Vol. 101). Wiley: New York.

DURBIN, R., EDDY, S. R., KROGH, A. and MITCHISON, G. Biological sequence analysis, Cambridge University Press, 1998.

FAMA, E. and FRENCH, K. (1988) Permanent and Temporary Components of Stock Prices. The Journal of Political Economy, 96(2), pp. 246-273.

FOGARASI, Norbert; LEVENDOVSZKY, János. Improved parameter estimation and simple trading algorithm for sparse, mean reverting portfolios. In: Annales Univ. Sci. Budapest., Sect. Comp. 2012. p. 121-144.

FOGARASI, Norbert; LEVENDOVSZKY, János. A simplified approach to parameter estimation and selection of sparse, mean reverting portfolios. Periodica Polytechnica Electrical Engineering and Computer Science, 2013, 56.1: 21-28.

FOGARASI, Norbert; LEVENDOVSZKY, János. Sparse, mean reverting portfolio selection using simulated annealing. Algorithmic Finance, 2013, 2.3: 197-211.

FORNEY, G. D. The Viterbi algorithm. Proc. IEEE, vol. 61, pp. 268-278, Mar. 1973.

VAN GELDER, P. (2009) Statistical modeling of financial markets. In: TU Delft, UFS Workshop, Bloemfontein, March 4-5 2009.

GILLESPIE, D. T. (1996). Exact numerical simulation of the Ornstein-Uhlenbeck process and its integral. Physical review E, 54(2), 2084.

HAGAN, M.T. and MENHAJ, M.B. (1994) Training feed-forward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks, Vol. 5, No. 6, pp. 989–993.

HAGAN, M. T., DEMUTH, H. B., & BEALE, M. H. Neural network design. Boston: Pws Pub., 1996.

HAMILTON, J. D. Analysis of time series subject to changes in regime. Journal of econometrics, 1990, 45.1: 39-70.

HAMILTON, James D.; SUSMEL, Raul. Autoregressive conditional heteroskedasticity and changes in regime. Journal of Econometrics, 1994, 64.1: 307-333.

HASSAN, R. and NATH, B. Stock Market Forecasting Using Hidden Markov Model: A New Approach. Proceedings of the 2005 5th International Conference on Intelligent Systems Design and Applications.

HASSAN, R., NATH, B. and KIRLEY, M. A fusion model of HMM, ANN and GA for stock market forecasting. Expert Systems with Applications 33 (2007) 171–180.

HENDERSHOTT, T., RIORDAN, R. Algorithmic trading and the market for liquidity. Journal of Financial and Quantitative Analysis, 48(04), 1001-1024. 2013.

HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. Neural networks, 1989, 2.5: 359-366.

IDVALL, P. and JONSSON, C. Algorithmic Trading — Hidden Markov Models on Foreign Exchange Data. Master's Thesis, Department of Mathematics, Linköpings Universitet, 2008.

ITO, K. (1944) Stochastic Integral. Proc. Imperial Acad. Tokyo, 20, pp. 519-524.

JANAKI, R. D., SREENIVAS, T.H., GANAPATHY, K. S. Parallel Simulated Annealing Algorithms. Journal of Parallel and Distributed Computing, 37, pp. 207-212, 1996.

JOLLIFFE, I. T. Principal Component Analysis. Springer-Verlag, New York, 1986.

JURAFSKY, D. and MARTIN, J. H. Speech and Language Processing: An Introduction to Natural Language Processing. Computational Linguistics, and Speech Recognition, 2nd Edition, Pearson Prentice Hall, London, 2006.

KIRKPATRICK, S., GELATT, C.D. and VECCHI, M.P. (1983) Optimization by Simulated Annealing. Science, 220, pp. 671-680.

KISSELL, R. The science of algorithmic trading and portfolio management. Academic Press, 2013.

KONNO, H. and YAMAZAKI, H. (1991). Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market. Management science, 37(5), 519-531.

MAMON, R. S. and ELLIOTT, R. J. Hidden Markov Models in Finance, Springer Science Business Media, 2007.

MANZAN, S. (2007) Nonlinear Mean Reversion in Stock Prices. Quantitative and Qualitative Analysis in Social Sciences, 1(3), pp. 1-20.

MARKOWITZ, H. (1952) Portfolio Selection. The Journal of Finance, Vol. 7 (1), pp 77–91.

METATRADER (2014) FOREX time series [WWW]. Available from: `http://www.metaquotes.net/en/metatrader5` [Accessed 02/2014].

MORGAN STANLEY MARKETONE (2010) SWAP series [DATABASE].

MURPHY, Kevin P. Machine learning: a probabilistic perspective. MIT Press, 2012.

NATARJAN, B.K. (1995) Sparse approximate solutions to linear systems. SIAM J. Comput., 24(2), pp. 227-234.

NVIDIA (2014) CUDA C programming guide [WWW]. Available from: `http://docs.nvidia.com/cuda/cuda-c-programming-guide` [Accessed 08/2014].

ORNSTEIN, L.S. and UHLENBECK, G.E. (1930) On the Theory of the Brownian Motion. Physical Review, 36(5), p. 823.

PAGÈS, G., WILBERTZ, B. GPGPUs in the computational finance: massive parallel American style options. Concurrency and computation: Practice and experience, 24, pp. 837-848, 2011.

POTERBA, J.M. and SUMMERS, L.H. (1988) Mean reversion in stock prices: Evidence and implications. Journal of Financial Economics, 22(1), pp. 27-59.

RABINER, L. R. and JUANG, B. H. An introduction to hidden Markov models. IEEE ASSP Magazine 3:4-16, 1986.

RABINER, L. R.. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77 (2), p. 257–286, February 1989.

RUPP, K. CPU, GPU and MIC Hardware Characteristics over Time [WWW]. Available from: `http://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/` [Accessed 06/2015].

TAYLOR, H. M. and KARLIN, S. (2014). An introduction to stochastic modeling. Academic press.

TIPPING, M. and BISHOP, C. Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B, 61, part 3, pp. 611-622, 1999.

VITERBI, A. J. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. IEEE Trans. Informat. Theroy, vol. IT-13, pp. 260-269, Apr. 1967.

YAHOO (2011) S&P 500 asset series [WWW]. Available from: `http://finance.yahoo.com` [Accessed 07/2011].