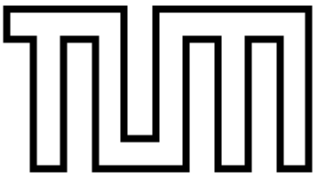


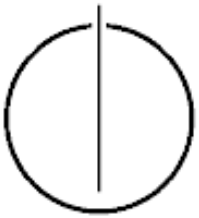
DEPARTMENT OF INFORMATICS

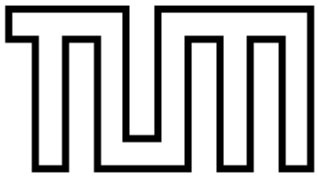
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Analysis of sparse-grids-based financial time series prediction

Ingo Mayer





DEPARTMENT OF INFORMATICS

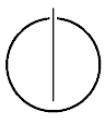
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Analysis of sparse-grids-based financial time series prediction

Analyse von dünngitter-basierter Zeitreihen Vorhersage mit Finanzdaten

Author:	Ingo Mayer
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	M.Sc. (Hons) Paul Cristian Sarbu
Submission Date:	15.10.2017



I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.10.2017 Ingo Mayer

Abstract

This thesis analyzes the forecasting of financial timeseries using sparse grids. Financial markets are complex, dynamical systems for which sparse grids are exceptionally suited. By applying the delay embedding approach the data will be transformed into a multi-dimensional regression problem that can be solved by sparse grids. The prediction will be tested on two different markets to evaluate the influence of various financial and grid parameters. Using historical data allows the immediate calculation of forecasting errors and provides insights into future performance. A pipeline for automated testing was built in Python, using the SG++ sparse grid framework.

Zusammenfassung

Diese Arbeit analysiert Vorhersage von Finanzmarkt-Zeitreihen mit Dünngittern. Finanzmärkte sind komplexe, dynamische Systeme. Dünngitter eignen sich hervorragend für die Prognose solcher Systeme. Die Methode der Verzögerungs-Einbettung wird verwendet um die Zeitreihen in ein multidimensionales Regressionsproblem zu transformieren, das von Dünngittern gelöst werden kann. Die Vorhersage wird auf zwei verschiedene Märkte angewendet um den Einfluss verschiedener Markt- und Gitterparameter zu evaluieren. Die Verwendung von historischen Daten erlaubt es direkt die Prognosefehler zu berechnen und Rückschlüsse auf die zukünftige Genauigkeit zu ziehen. Hierfür wurde eine Pipeline in Python programmiert, die das Dünngitter-Framework SG++ verwendet um automatische Tests zu ermöglichen.

Contents

1. Introduction

Timeseries forecasting describes the process of using a timeseries $s_{j=1}^{N}$ to predict its next value s_{N+1}. The timeseries is usually equidistant over time. A variety of machine learning approaches exist to solve this problem. This thesis will detail the approach used in forecasting a timeseries with sparse grids using the delay embedding method. For this purpose the open source general sparse grid toolbox SG++ will be used to apply sparse grids on financial markets.

1.1. Motivation and Problem

Machine learning theory has existed for more than 20 years, however computational limitations resulted in removing their scientific progress from the public eye. The recent hardware improvements have remedied this and allowed a reawakening of these theoretical concepts that have already been applied in numerous fields.

The prediction of financial markets is one of the most studied fields of time series forecasting. Due to its history and the tremendous impact capital markets have on both global and national economies [**8**], it is a dynamic system with countless influence factors. It takes a lifetime of study to fully understand these markets, if is possible at all. An accurate prediction in such a challenging environment is not only a scientific feat, it would be of significant economical value.

1.2. Related Works

Existing financial forecasting implementations show considerable promise with results already allowing fully automatic investment that gives positive returns. However, they are highly limited by the amount of data needed, the computing time and storage requirements due to the curse of dimensionality. Sparse grids can mitigate this problem by only scaling linearly with the dimension and have shown comparable results. Applying sparse grids to timeseries is introduced in [**1**] and applied to a number of test cases. To provide a comparison, some of these tests were implemented with the SG++ library.

In the Master's Thesis by [**9**], long short-term memory recurrent neural networks are used in a similar application that is tested on six randomly selected stocks on the S&P 500 index and achieved positive returns during a 400 hour investment scenario. Moreover, the nearest neighbor and random-forest concepts were applied to predicting U.S. inflation by Sebastian Lebert [**10**] and achieved results that are up to 15% better than related metrics.

1.3. Structure

Chapter 2 introduces the reader to the theoretical concepts behind sparse grids.

In chapter 3 the dynamics of finance markets and their history will be detailed. Two related processes will be reviewed and possible application scenarios of a forecast discussed.

The fourth chapter describes the concrete implementation.

The results for both tested markets are analyzed in level 5 and the influence of different grid parameters illustrated.

Finally, the conclusion summarizes the used approach and provides a few thoughts on possible improvements and future applications.

2. Sparse Grids

Sparse grids are a numerical discretization technique for multi-dimensional problems. Conventional methods are limited to discretizing differential equations with a maximum of 3 or 4 dimenions because of the so-called *Curse of Dimensionality*. The term was introduced by Bellmann in 1961 and describes the cost of computing and storage requirements of a discretized differential equation solution, which grows exponentially with the dimension.[**3**]

Regular sparse grids are still somewhat limited by the problem dimension. However, various approaches exist to further circumvent this issue such as spatially adaptive grids which will be analyzed in chapter 5. The following section will illustrate how a timeseries can be transformed into a multi-dimensional problem by using the delay embedding theorem by Taken and introduce the theoretical framework of sparse grids. A more detailed explanation can be found in [**11**].

2.1. Delay Embedding

By using the delay embedding technique, timeseries forecasting can be turned into a regression problem that scales linearly with the length of the timeseries. This provides substantial computational benefits while retaining the desired accuracy. The following steps have to be performed to predict future timeseries values with a sparse grid approach [**1**]:

1. Estimate the dimension m of the underlying process
2. Turn the forecasting into a regression problem in $\mathbb{R}^d$ with $d = 2m + 1$
3. Train the regression solver in the discretized sparse grid space
4. Predict s_{n+1} by evaluating the delay vector $(s_{n-2m}, ..., s_n)^T$

Estimating the Embedding Dimension

The correlation dimension in non-linear dynamic systems can be calculated by the Grassberger Procaccia algorithm.[**4**]

Let the timeseries be described as $(s_j)_{j=1}^{\infty}$ with the goal of using the values $s_1, ..., s_n$ to predict s_{n+1}. The correlation dimension m of this timeseries, which is needed to apply the delay embedding technique, can be estimated by calculating the correlation integral of the timeseries for a positive number r:

$$C(r) = \frac{2}{N(N-1)} \sum_{i \neq j} \theta(r - |s_i - s_j|) \tag{2.1}$$

with $\theta(x)$ being the Heavyside step function.

If $C(r)$ decreases like a power law $C(r) \approx r^m$ then m is the correlation dimension of the timeseries[1] and can thus be used in the delay embedding.

[1] http://www.scholarpedia.org/article/Grassberger-Procaccia_algorithm, visited on 01.10.2017

Formulating the Regression Problem

According to the delay embedding scheme we can create the delay vectors t_j for a dimension d in $\mathbb{N}$, with the j-th delay vector being:

$$t_j := (s_{j-d+1}, s_{j-d+2}, ..., s_{j-1}, s_j)^T \in \mathbb{R}^d, j \geq d \tag{2.2}$$

Each vector contains d consecutive timeseries values. Taken's Theorem states that each s_{n+1} is completely described by the previous $d = 2m + 1$ values and a $\hat{g} : \mathbb{R}^d \to \mathbb{R}$ exists so that

$$\hat{g}(t_j) = s_{j+1} \forall j \geq d. \tag{2.3}$$

It is now a regression problem of the form $g(t_j) \approx \hat{g}(t_j)$ **[1]**.

If a historical timeseries is used, then the values $(s_j)_{j=1}^N$ values are known and can be used in constructing training vectors with which (2.3) will be approximated in the sparse grid space as detailed in the next section.

2.2. Regular Sparse Grids

Sparse grids operate in the d-dimensional hypercube $[0, 1]^d$. We will assume that the regression problem $g(t_j) \approx \hat{g}(t_j)$ lies in the same space. Therefore a timeseries $(s_j)_{j=1}^\infty$ first needs to be rescaled.

The interpolation is performed by using basis functions, which leads to the extended regression equation

$$g(t_j) \approx \hat{g}(t_j) = \sum_i \alpha_i \phi_i(t_j). \tag{2.4}$$

If we assume the approximated function to be 0 at the boundaries, then the standard hat function can be used as basis function

$$\phi(x) = max(1 - |x|, 0). \tag{2.5}$$

The basis function has to be restricted to the interval $[0, 1]$, which can be done for a 1-dimensional hat function by

$$\phi_{l,i}(x) := \phi(2^l x - i), \tag{2.6}$$

with l as the grid level and i the index.

Thus, the multi-dimensional basis function results in

$$\phi_{\vec{l},\vec{i}}(x) := \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j), \tag{2.7}$$

where $\vec{l}, \vec{i}$ are d-dimensional indeces, denoting the level and index for each dimension. The hierarchical subspaces W_l needed to construct a sparse grid of a particular level l can be obtained as follows [**2**]:

$$I_{\vec{l}} := \{\vec{i} : 1 \leq i_j \leq 2^{l_j}, i_j \text{ odd}, 1 \leq j \leq d\}, \tag{2.8}$$

which leads to

$$W_l = \text{span}\{\phi_{l,i} | i \in I_l\}. \tag{2.9}$$

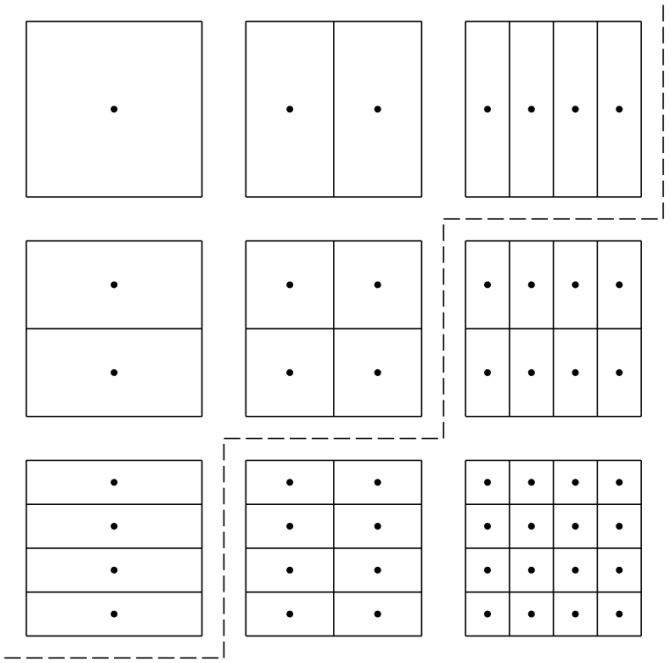


Figure 2.1.: Two dimensional subspaces [**3**]

The complete sparse grid is constructed by combining all subspaces above the dashed line. Using all subspaces would result in a regular grid.

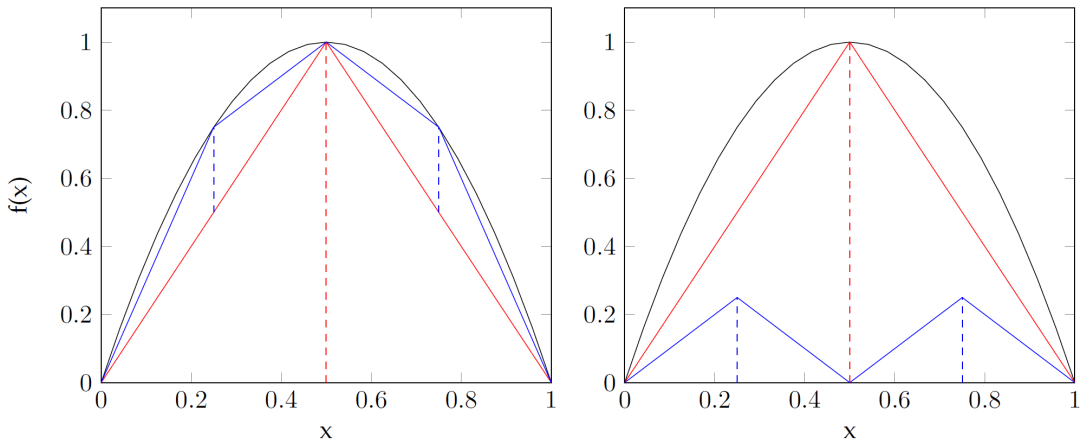


Figure 2.2.: Approximating a parabola function for a 1-dimensional grid with level 1 (red) and level 2 (blue) [**2**]

Modified Basis Functions

Timeseries are not always 0 at the boundary. In order to model this, additional grid points have to be added at the boundary of the sparse grid through an extra level ($l = 0$) with the basis functions. However, this results in a large number of new points for higher dimensions.

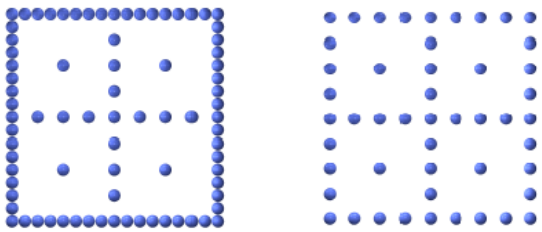


Figure 2.3.: One-dimensional hierarchical basis functions, with two basis functions located on level 0 on the boundary (left) and One-dimensional hierarchical basis functions with level 1 being extended by two extra level 0 basis functions located on the boundary (right)[**12**]

Therefore, in situations that do not require high accuracy at the boundaries, it is better to instead modify the interior basis functions to extrapolate towards the boundary [**12**] as can be seen in figure 2.3.

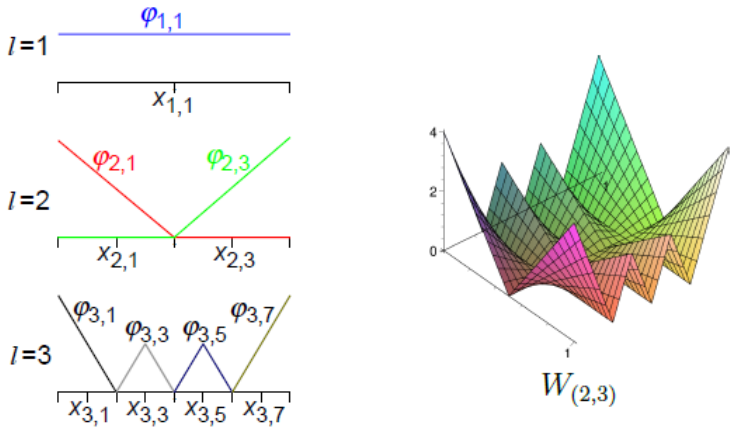


Figure 2.4.: Extrapolation towards boundary [**12**]

Figure 2.4 demonstrates how this modifies the basic hat functions at the border. This makes it possible to use non-zero timeseries boundary values, which will be used in the application of sparse grids on financial timeseries in chapters 4 and 5.

Linear System for Delay Embedded Vectors

The regression problem $g(t_j) \approx \hat{g}(t_j)$ can be written as the minimization problem

$$g = \arg\min_{f \in X} \mathcal{F}(f). \tag{2.10}$$

By using a Lebesque measurable cost function $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \cup \{\infty\}$ and the empirical density function $\hat{p}(x, y) = \frac{1}{N-d} \sum_{j=d}^{N-1} \delta_{t_j}(x) \delta_{s_{j+1}}(y)$, the minimization problem is then given by

$$g = \arg\min_{f \in X} \mathcal{F}(f) = \arg\min\left(\frac{1}{N-d} \sum_{j=d}^{N-1} c(s_{j+1}, f(t_j))\right). \tag{2.11}$$

Since this is an ill-posed problem, an additional constraint of the form $\Psi(f) \leq c$ is added, which leads to the following well-posed problem for positive λ values

$$g = \arg\min_{f \in X} \mathcal{F}(f) = \arg\min\left(\frac{1}{N-d} \sum_{j=d}^{N-1} c(s_{j+1}, f(t_j) + \lambda \Psi(f))\right). \tag{2.12}$$

Solving this equation in the discrete sparse grid space leads to the final linear system

$$\sum_{j=d}^{N-1} s_{j+1} \phi_{l,i}(t_j) = \sum_{k \in \mathbb{N}^d : n_d(k) \leq t, m \in I_k} \alpha_{k,m} \left(\sum_{j=d}^{N-1} \phi_{l,i}(t_j) \phi_{k,m}(t_j) + \eta h(\phi_{l,i}, \phi_{k,m}) \right), \tag{2.13}$$

for all $l \in \mathbb{N}^d : n^d(l) \leq t$ and $i \in I_l$.

2.3. Spatial Adaptivity

Regular sparse grids are often not the best approach to approximate functions in the sparsely populated hypercube space $[0,1]^d$. If a low level is used, the problem of having too few points for an accurate approximation can occur. Likewise, too many points at fixed locations can lead to overfitting, as can be seen in the results chapter.

It is therefore a reasonable approach to start with a grid of lower level and adaptively refine the points by adding more at locations of large error. This process, as described in **[1]** uses the simple error estimator

$$||\alpha_{l,i}\phi_{l,i}||_{L_\infty(H_d)} = |\alpha_{l,i}|. \tag{2.14}$$

The process starts with a low grid level and calculates the learning error at every grid point. The adaptive algorithm then adds all child nodes for those points, whose error $\epsilon_{l,i}$ is above a fixed threshold ϵ. If the percentage of points with an error above the threshold exceeds the specified adaptivity rate, then only the points with the highest errors are refined.

For a multi-dimensional grid, child($x_{l,i}$) is defined as

$$\begin{aligned} x_{k,m} |\ &\textit{There exists } j \in \{1,...,d\},\ \textit{s.t. } x_{k_j,m_j} \in \textit{child}(x_{l_j,i_j}) \\ &\textit{and } k_h = l_h, m_h = i_h \textit{ for all } h \in \{1,...,d\}\backslash\{j\}. \end{aligned} \tag{2.15}$$

It is important to keep in mind that it is not viable to only recalculate the errors of the adapted points in the next iteration, since the errors of all points may have changed.

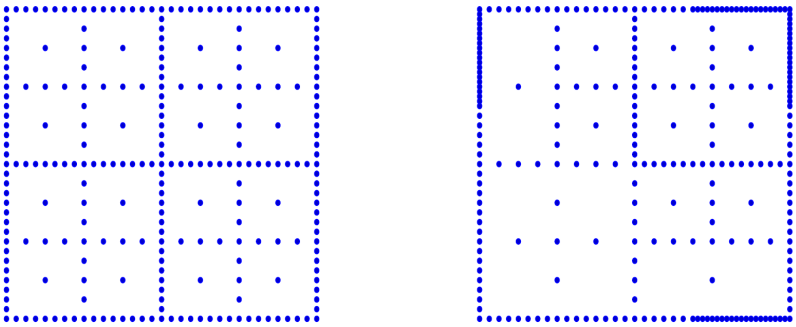


Figure 2.5.: Regular sparse grid of level 5 (left) and corresponding spatially adaptive grid (right) **[1]**

3. Financial Markets

Financial market forecasting is invaluable from an economical point of view. Financial markets link stronger elements to weaker ones and as such their development has a tremendous impact on economical growth. Studies have shown that "the financial market is not just capable of promoting economic growth, but a certain level of financial market development helps to prevent an economy from entering a crisis."[8] This clearly demonstrates the importance of capital markets in our globalized society and the associated benefit of an accurate prediction.

Moreover, the evolution of trade structures, like the introduction of currency as an intermediary, has continuously shaped financial systems and led to the global, capitalist market that exists today. The central principle continues to be an exchange of goods, a basic concept that has existed since the inception of humanity and can be described as one of the core building blocks of society. Tracking the development of this evolution and linking it to current events is therefore interesting from a sociological standpoint and may serve as a helpful tool in adapting to inevitable changes.

The following pages give an introduction into capital market development and the terminology used within financial publications. Afterwards the two markets that will be analyzed in this thesis will briefly be detailed along with an overview of market behavior theory that explains the used methodology and assumptions. As a final topic, this chapter we will summarize a few prominent examples of potential real-life financial timeseries forecasting applications.

3.1. History

Financial markets are complex systems with countless influencing factors. Many of these are the result of historical events. An in-depth look at its evolution is outside the scope of this thesis. However, a brief foray into the history of capital markets, which this section will provide, is necessary to gain a basic understanding.

The issuing of governmental and corporate loans can be traced all the way back into the 14th century, where Venetian lenders started selling debt issues[1] and the first stock exchange was already created in 16th century Belgium. Nevertheless, these are both relatively minor events in the evolution of capital markets. The major foundation stones for the current financial system were laid in the 17th century through the British East India Company and the infamous South Sea Bubble[2].

Originally, most trading companies that obtained goods in South-East Asia in the 17th century formed for only one trade journey and then disbanded. In order to finance their endeavors, the usual practice was to find an investor that was then given a share of the profits upon successful return.
The East India companies, which were trading companies with a government charter, changed this by paying dividends based on profits from all journeys to shareholders, instead of disbanding after a single trip.
The British East India company in particular was able to distribute significant dividends as they were the only company with a government-backed monopoly. The resulting financial boom was not regulated. This led to large investments into the new South Sea Company in 1711, which did not even start engaging in trading until 1717. Motivated by this illusory success, businessmen started selling shares for increasingly shady ventures. Because of various geopolitical factors The South Sea Company was unable to follow up on its absurd promises and caused a market crash that prompted the government to outlaw shares.
In the United States, the New York Stock Exchange (NYSE) founding in 1792 followed relatively quickly upon the end of the independence war in 1782. The NYSE had its share of peaks and troughs, most prominent among them the Great Depression of 1929.

This brief history overview shows that the development of the financial market system has always been dominated by economical factors and political events, which will almost certainly remain similar in the future. As previously stated, it is therefore of particular economical and sociological interest to study and predict its behavior.

[1] http://www.investopedia.com/articles/07/stock-exchange-history.asp, visited on 15.09.2017
[2] https://www.library.hbs.edu/hc/ssb/history.html, visited on 15.09.2017

3.2. Terminology

This chapter outlines a few important terms used in financial publications that will allow the reader to follow the description of used methods and the outlined real-life application possibilities.

Volatility is a "measure of a [financial goods'] stability. It is calculated as the standard deviation from a certain continuously compounded return over a given period of time. It is an important measure in quantifying risk; for example, a security with a volatility of 50% is considered very high risk because it has the potential to increase or decrease up to half its value." [3]
In order to determine its influence on a prediciton, this thesis will analyze both highly volatile timeseries as well as relatively stable series.

Trends identify specific market phases. There are three main types of trends: The upwards trend (often called "bullish market" since a bull strikes upwards with its horns) that indicates an increase in asset price; the downwards trend (often called "bearish market" since a bear strikes downwards with its paws) that indicates a decrease in asset price and a sideways trend, which shows little change in asset price. Trends can be identified for different periods of time and are frequently used in the field of technical analysis.

Stocks are financial assets that are traded on stock exchanges. Stocks are divided into shares that represent ownership of a company in proportion to the total number of shares and entitle the shareholder to partake in the companies successes through an increased stock valuation and dividends. In addition, common stocks grant voting rights in shareholder meetings.

Stock Data is provided by stock exchanges. In the context of this paper this refers to various different timeseries, which contain specific stock values. In addition to current and closing prices, stock exchanges provide opening, high and low price values, as well as information on traded volume. This price data will sometimes show significant, immediate drops that are the result of stock splits and paying out dividends. The adjusted version of each timeseries, which is also provided by stock exchanges, takes this into consideration and should always be used for forecasting. In this thesis only adjusted closing prices will be considered.

[3] "Volatility." Farlex Financial Dictionary. 2009. Farlex 30 Sep. 2017 http://financial-dictionary.thefreedictionary.com/Volatility, visited on 16.09.2017

Market Indeces combine several stocks that aggregate their total value compared to a base value to represent an entire stock market and track its development over time. The SP500 index is a prominent example of this and will be used for the analysis.

Strategies are procedures that determine investment or trading decisions in order to make a profit on financial markets and can be categorized by their assumed risk and return. This will be further detailed during the discussion of a real-life application.

3.3. The U.S. Stock Market

The financial timeseries forecasting analysis will deal with two different markets. The first is the U.S. stock market, one of the biggest in the world. In particular, the SP500 index, which aggregates the value of the largest 500 U.S. companies traded on stock exchanges, and the SP500 company stocks themselves will be analyzed.

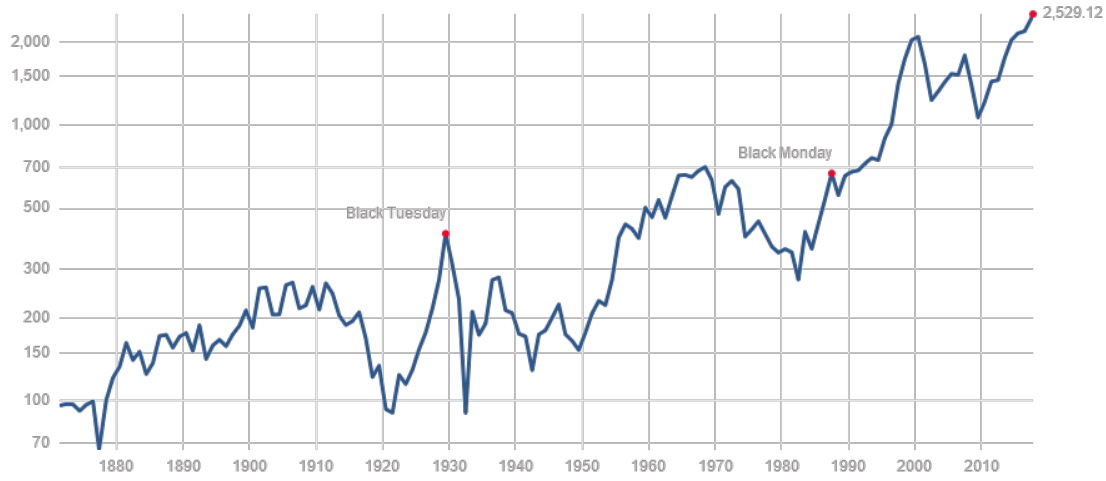


Figure 3.1.: SP500 Value Development (adjusted for inflation)
Source: http://www.multpl.com/inflation-adjusted-s-p-500

Most financial forecasting methods use the SP500 index as a validation data set as it is a good representation of the overall stock market. It has the most publicly available stock data and is one of the most avidly watched indices in the world. Moreover, the index has been relatively stable in recent years, which makes it an ideal candidate to attempt a prediction.

3.4. Cryptographic Currencies

Cryoptgraphic currencies like Bitcoin are described as a "purely peer-to-peer version of electronic cash [that] would allow online payments to be sent directly from one party to another without going through a financial institution."[**11**]. They use the Blockchain technology to store and verify transactions in a digital and distributed manner.
The market capitalization of the over 1100 existing cryptocurrencies is approximately $ 60 billion. Since Bitcoin was the first such currency, others are often summarized under the term "altcoin".

Figure 3.2.: Bitcoin Value Development
Source: https://www.buybitcoinworldwide.com/price/

Cryptographic currencies have seen an extreme increase in value. They are highly volatile due to controversial opinions and a lack of government backing. A selection of cryptographic currencies will be used to determine the impact of volatility on the prediction.

3.5. Financial Timeseries Forecasting

With a market capitalization of roughly $ 70 trillion in 2015[4] the stock market is one of the largest global markets and arguably the most prominent global man-made economical influencer. Applying a relatively young (in terms of applicability, because of computational limitations) technology to a non-linear and dynamic system that impacts the daily life on a global scale, presents a considerable challenge. Comparing the results of a sparse grid-based prediction to state-of-the-art methods such as long-short term memory recurrent neural networks (LSTM RNN) will provide an overview of the applicability of the delay embedding approach in finance.

In the information age, everyone has access to financial markets and their data. The continuous growth of the global stock market reflects a growing number of active participants that use it to secure their financial future. Nevertheless, it is currently necessary to either hire a professional or to spend large amounts of time on both investment and trading, in order to receive a substantial benefit. This leads to the additional motivation of providing a new method of securing personal finances. Having a predictive support tool can not only increase investment security, but also make the stock market accessible to those that cannot afford a professional asset manager or do not have the time to manage their own portfolio.

Moreover, the number of use-cases for predictive tools in this sector are vast, e.g. a validation tool for economic cycle identification, a support tool for identifying hidden influences on financial markets that will lead to a more objective understanding or, in light of the recent financial crisis caused by speculative decisions, a regulatory application to limit excessive risk. In a later chapter we will discuss using the prediction for asset trading in quantities that do not influence the market.

We will now take a look at financial market behavior, for which various different hypotheses exist. Prominent among them is the Efficient Market Hypothesis which claims that financial markets are systems of complete information. Chaos theory, on the other hand, proposes that financial markets are non-linear systems and as such not designed to ensure fair pricing. Both the (weak) Efficient Market Hypothesis and Chaos Theory follow the approach that future behavior can, to some degree, be linked to past data, thus providing the theoretical foundation of financial timeseries forecasting. The following sections will briefly introduce both concepts.

[4] http://www.visualcapitalist.com/all-of-the-worlds-stock-exchanges-by-size/, visited on 02.09.2017

3.5.1. Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH), sometimes referred to as Random Walk Theory, assumes that all information available is already considered in current stock prices. This makes it highly unlikely to outperform the general market performance, except through exploiting highly localized inefficiencies. This efficient market exists because of the large number of analysts searching for these inefficiencies and acting upon them to make profits. Therefore, efficiency grows in proportion to market participants. The term first appeared in 1965 by E.F. Fama, who states that "the evidence in support of the efficient market model is extensive, and (somewhat uniquely in economics) contraditory evidence is sparse."[**5**] The theory proposes three forms of efficiency (weak, semi-strong, strong) that propose different degrees of information inclusion. The weak form states that the current price incorporates historical prices only, while the strong form is based on complete information incluson.[**6**] Based on this theory, it is evident that only weak market efficiency allows for a useful prediction.

3.5.2. Chaos Theory

In contrast to the Efficient Market Hypothesis, Chaos Theory follows the approach that "prices seem to be highly random but contain some type of trend. The amount of trend varies with the type of market and the time frame being analyzed."[**7**] A commonly used concept in this context are fractals, objects that are individually different but related to the whole. A good example for this is "the branching network in a tree [...]. Each branch is different from the others; however, the branches are similar to the structure of the whole tree."[**7**] Chaos Theory studies overall behavior of a non-linear system - the underlying rules cannot be identified. Since financial markets are non-linear, dynamic systems, "deterministic chaos represents the best trade-off to establish fixed rules in order to link future dynamics to past results of a time series without imposing excessively simple assumptions."[**13**]

3.5.3. Related Methods

The finance market is one of the most prominent research topics for machine learning and predictive analytics. Thus, numerous papers exist describing various approaches and application fields. Two of these will be summarized briefly to give an example of other approaches used in this area.

LSTM RNN For Financial Timeseries Paper

The Master's Thesis by Qiyan Gao **[9]** analyzes the feasibility of using Long short-term memory recurrent neural networks (LSTM RNN) for financial forecasting. LSTM RNN are designed to predict timeseries data with long time delays between vital events and has seen a rise in prominence through its great results in speech and handwriting recognition. It is interesting to note that Gao tested his prediction results in an investment scenario and achieved remarkable returns with a profit of $ 413,233.33. using a $ 6,000,000 initial investment and a time period of 400 hours.

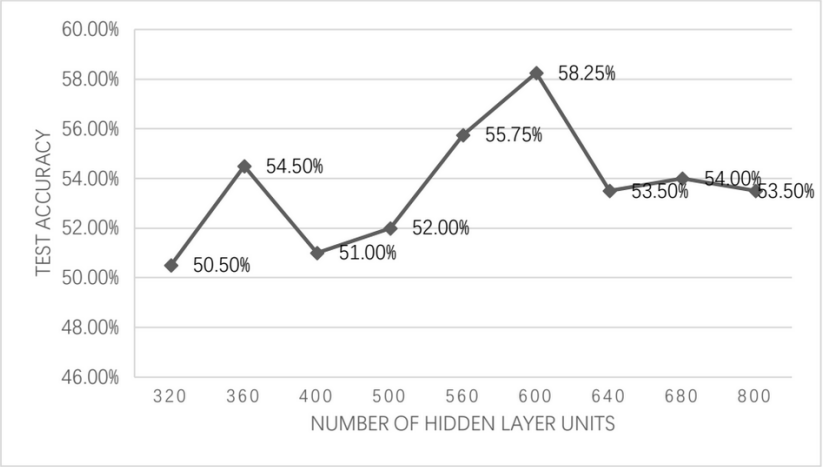


Figure 3.3.: QUALCOMM Test Accuracy with Various Number of Hidden Layer Units **[9]**

Gao chose six randomly selected stocks from different industries, all of which are listed in the SP500 index: Black Rock, Inc., Alphabet, Inc., QUALCOMM, Inc., Exxon Mobil Corp., International Business Machines Corp. and JPMorgan Chase & Co. Additionally, Gao used stock index and competitor hourly data for feature creation. The prediction accuracy using LSTM RNN is highly dependant on the number of neurons in hidden layers. Gao tested a range of 320 - 800 neurons with a peak accuracy at 600. An average maximum accuracy of 54.83% was achieved for the selected stocks.

Machine Learning for Inflation Paper

The Master's Thesis by Sebastian Lebert [**10**] analyzes the use of several machine learning approaches, such as the nearest neighbor algorithm and random forests, to forecast U.S. inflation. This paper clearly demonstrates that financial forecasting is not limited to the stock market but allows for a great variety of possible application scenarios.

Lebert chose 8 different quarterly datasets:

1. Consumer price index for all urban consumers (CPI)
2. Global price of brent crude in USD per Barrell
3. Median sales price for new houses sold in the United States in USD
4. Producer price index for all commodities
5. Unemployment rate: 20 years and over in percent
6. All employees: Total nonfarm payrolls in thousands of people
7. Closing price of the S&P 500
8. Spread between the 10-year treasury constant maturity rate and the 3-month equivalent in basis points

Datasets 1, 5 and 6 are seasonably adjusted. He first attempts a prediction with each individual variable, then combines single variables with the Consumer Price Index (CPI) and finally tests a combination of three. The use of S&P 500 data in this scenario is further evidence of the stock market influence on national and global economics.

He concludes that machine learning approaches are "at least as as accurate as univariate benchmark models"[**10**]. For specific variable combinations the results are up to 15% more accurate than comparable benchmarks.

Furthermore, Lebert addresses the downside of timeseries predictions. An algorithms goal is a maximum of accuracy by finding correlations. This allows no interpretation of causality and is therefore of limited use in understanding why the timeseries behaves in the predicted manner.

3.6. Applications of Financial Forecasting

The focus of this thesis is analyzing the results of a financial timeseries prediction. Nevertheless, it is of importance for such an analysis to consider potential applications and the needed prediction accuracy to realize these. This chapter will briefly detail two potential use-cases that will discussed further in chapter 5.

Forecasting is a "planning tool that helps management in its attempts to cope with the uncertainty of the future, relying mainly on data from the past and present and analysis of trends."[5] Separating this into different time horizons allows us to define two application scenarios: trading (short-term) and investment (long-term). However, since financial market dynamics tend to be highly localized in time and space, a prediction for investment time horizons, which are typically between 5 and 15 years, is quite difficult.
Thus, the much more lucrative field of application is short-term financial asset trading, where a prediction accuracy of more than 50% over a short period of time is sufficient to achieve positive returns under the assumption that each trade is performed with the same amount of money and transaction fees are not applied. This, in turn, leads to two possible types of predictive trading applications:

Forecast as Strategy The prediction can be used as a direct strategy that indicates when to buy and sell assets. The percentages of predicted gain or loss that trigger a buy/sell signal will have to be globally optimized or customized for each trader, based on their preferred risk. Customer risk preference categorization is a widespread methodology in financial asset management.

Forecast as Risk Classification Financial asset traders use a variety of different information to build their custom strategies. Data streams are combined to signal buy/sell decisions. The timeseries forecast does not have to be used as a direct strategy, instead it can be used to reinforce decisions by indicating whether the buy/sell decision is likely to correlate with the asset price movement in the near future. This allows for a less accurate prediction to be used.

These two applications are merely examples for asset trading. As previously stated, the number of possible applications in general are quite numerous but require an increasing degree of accuracy or a longer time frame, which is the main limitation of a timeseries prediction.

[5] http://www.businessdictionary.com/definition/forecasting.html, visited on 26.09.2017

4. Implementation

This chapter will provide a documentation of how the results in chapter 5 were reached. First, the SG++ library, which was used to construct the sparse grids, will be introduced and the chosen datasets and their sources explained. Finally, the implementation itself will be detailed in three parts: data aquisition, sparse grid construction and results evaluation.

4.1. The SG++ Library

The SG++: General Sparse Grid Toolbox[1] is a universal, open source toolbox for spatially adaptive sparse grid methods and the combination technique. It provides various low-level and high-level sparse grid functionality allowing one to start using sparse grids with minimal initial implementation effort. The functionality ranges from interpolation and quadrature via the solution of differential equations to regression, classification, and more. The SG++ toolbox supports multiple operating systems and multiple programming languages.

[1] D. Pflüger, Spatially Adaptive Sparse Grids for Higher-Dimensional Problems. Verlag Dr. Hut, München, 2010. ISBN 9-783-868-53555-6, visited on 27.09.2017

4.2. Data Sources

All stock and Bitcoin data is obtained from Quandl[2], a data platform that delivers financial, economic and alternative data and is gaining popularity in financial data science because of its free, high quality datasets. Additionally, a large number of professional datasets are purchasable.

The historical Ethereum prices were obtained from Etherscan.[3]

As mentioned in chapter 3, two datasets will be evaluated in this thesis. A list of both datasets can be found in appendix XY. The companies listed on the SP500 index are a good representation of the global stock market with a relatively limited volatility and serve as the main evaluation set. However, a proper assessment of all 500 would be outside the scope of this paper. Therefore, a random selection of 50 companies listed in the index were chosen as a representation. Their datasets are end of day stock quotes taken from the Quandl EOD Wiki.

The cryptographic currencies evaluation aims to analyze the prediction performance on new markets that are highly unstable. For this purpose the currencies Bitcoin and Ethereum will be analyzed.

Table 4.1.: Prediction Timeframes

#	Learning	Prediction
1	01.01.2000 - 31.12.2015	01.01.2016 - 31.12.2016
2	01.01.2000 - 31.12.2015	01.01.2016 - 31.01.2016
3	01.01.2010 - 31.12.2015	01.01.2016 - 31.12.2016
4	01.01.2015 - 31.12.2015	01.01.2016 - 31.01.2016

Four time frames were selected for the evaluation to determine the influence of both the training and testing length on the forecast. Cryptographic currencies were first established in the mid- to late 2000s, therefore less data exists for this prediction. The smallest time frame (4) was used in this evaluation.

[2]https://www.quandl.com/, visited on 01.07.2017
[3]https://etherscan.io/chart/etherprice, visited on 01.07.2017

4.3. Financial Predictor Pipeline

The pipeline is written in Python to make use of its numerical libraries and quick prototyping times. In order to make testing faster and to provide a basis for later application implementation, a simple graphical user interface was built with PyQt5 for changing the sparse grid parameters and the data source.

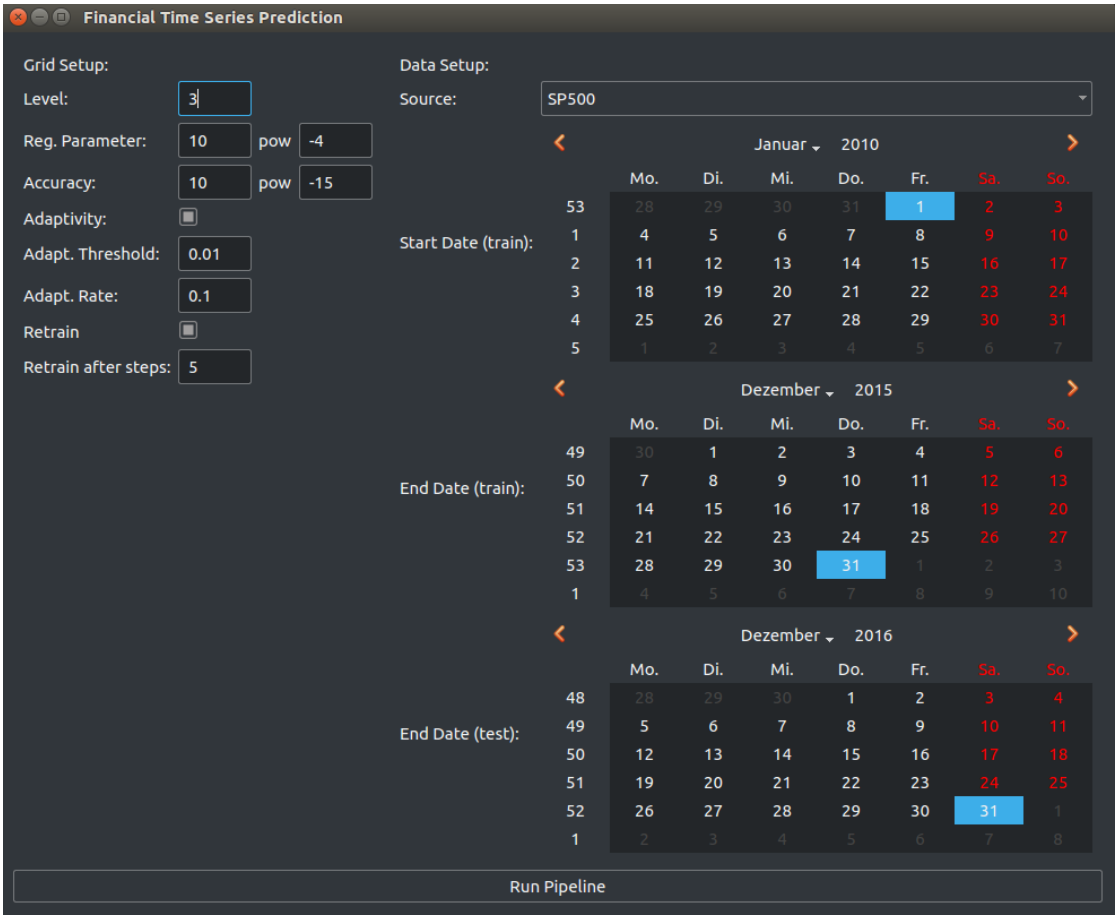


Figure 4.1.: Pipeline GUI

As the name suggests, the pipeline provides automated functionalities for each step from data aquisition to results evaluation. Analyzing the selected amount of data would otherwise have been impossible in the given time.

4.3.1. Data Aquisition & Preprocessing

The data obtained from Quandl cannot be immediately be used to predict future stock values. It first has to undergo a few preprocessing steps that turn the stock price timeseries into delay vectors as detailed in chapter 1.

Streaming 16 years of stock market data for a prediction would take an enormous amount of time, which is why all datasets were downloaded once, saved to a CSV file and then loaded locally as a Pandas dataframe. Additionally, the embedding dimension for each dataset was calculated with the Python package *nolds* upon download and saved in a separate CSV file. These two changes significantly reduced testing time.

```
def transform_timeseries_to_datatuple(timeseries, dimension):
        # data can exceed [0,1] sparse grid range later -> 25% padding
        max = dataframe.max()
        min = dataframe.min()
        padding = (max - min) / 4
        modify max/min with padding
        rescaled_series = []
        for (value in dataframe):
                rescaled_value = rescale values into sparse grid range [0, 1]
                rescaled_series.append(rescaled_value)
        length = len(timeseries) - dimension
        # build list of values to compare the prediction to
        value_list = rescaled_series[dimension:]
        # build delay vector list
        delay_vectors = []
        for (i in range(len(timeseries) - dimension delay vectors)):
                delay_vector = []
                # each delay vector contains |dimension| elements
                for(i in range dimension):
                        delay_ vector.append(rescaled_series[0])
                        rescaled_series = rescaled_series[1:]
                delay_vectors.append(delay_vector)
        return delay_vectors, value_list
```

Figure 4.2.: Transform Timeseries into Delay Vectors

Henon & Jump Map

The pipeline used in obtaining the comparison results for the Henon and Jump Map differs only in the timeseries creation. Instead of downloading stock data from a data provider, these values had to be calculated.

The Henon Map is a well-studied example of a dynamical system with chaotic behavior. For its calculation the standard parameters $a = 1.4$ and $b = 0.3$ were chosen. Additionally the two start values were set to $x_0 = 0.1$, $x_1 = 0.2$. A total of 20000 values was calculated.

```
def calculate_henon():
        values = []
        values.append(x0)
        values.append(x1)
        for (i in range(2, 20000)):
                value = a - pow((values[i - 1], 2) + b*values[i - 2]
                values.append(value)
        return values
```

Figure 4.3.: Henon Map Calculation

The Jump Map start values were set to $x_0 = 0.1$, $x_1 = 0.2$. A total of 20000 values was calculated.

```
def calculate_jumpmap():
        values = []
        values.append(x0)
        values.append(x1)
        for (i in range(2, 20000)):
                value = (values[i - 2] + values[i - 1]) % 1
                values.append(value)
        return values
```

Figure 4.4.: Jump Map Calculation

4.3.2. Sparse Grid Construction

The SG++ library provides easy-to-use functionalities in form of the *LearnerBuilder* class to create a sparse grid learner with the proper parameters. In the following code example, the variables were already passed to the implementing class and are now used in constructing the learner. The exact parameters chosen for each test will be given as part of the result documentation in chapter 5.

```
# learners available as classificator/regressor, need regressor
builder = LearnerBuilder().buildRegressor()

# 1. setup grid with chosen level and border type as detailed in chapter 1
builder = builder.withGrid().withLevel(level).withBordeer(Types.BorderTypes.NONE)

# 2. set the training data, see preprocessing chapter for data parsing
builder = builder.withTrainingDataFromNumpyArray(scaled_samples, scaled_values)

# 3. set regression parameters and adaptivity as selected in the GUI
if(with_adaptivity):
       builder = builder.withSpecification().withLambda(regression_parameter)\
AdaptRate(rate).withAdaptThreshold(threshold)
   else:
          builder = builder.withSpecification().withLambda(regression_parameter)

# 4. set 3x cross-validation folding
builder = builder.withRandomFoldingPolicy().withLevel(3)

# 5. set stop policy for adaptivity to use a max. of 5 iterations
builder = builder.withStopPolicy().withAdaptiveItarationLimit(5)

# 6. set conjugate gradient solver
builder = builder.withCGSolver().withAccuracy(accuracy).withImax(400)

# 7. get result
learner = builder.andGetResult()
self.learner = learner.learnData()
```

Figure 4.5.: Sparse Grid Learner Construction

4.3.3. Retraining

Machine Learning applications usually undergo a retraining after a specific amount of time has passed in order to use the newly available data to improve the model. The pipeline provides this functionality. The retraining steps selection in the GUI specifies the number of prediction steps, in this case days, that will be performed before the grid is retrained.

4.3.4. Results Evaluation

The prediction uses the *predict_next_value* function, as given below, to predict a time-series value based on a given delay vector and the alpha values calculated during the learning process as detailed in chapter 1. The operation evaluation class is provided by the SG++ library.

```
def predict_next_value(test_vector):
        opEval = pysgpp.createOperationEval(self._learner.grid)
        ''' numpy vector needs to be transformed into
        special SG++ data structure'''
        vector = DataVector(test_vector)
        return opEval.eval(learner.alphas, vector)
```

Figure 4.6.: Prediction Function

Afterwards the root mean squared error (RMSE), a commonly used evaluation metric, is calculated for each timeseries and written to a file. Additionally, the average of all RMSEs for each individual test is calculated.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Figure 4.7.: Root Mean Squared Error (RMSE)

5. Results

The results are grouped into three sections: Results that are compared to results in a reference paper, results for the S&P 500 index companies, and cryptographic currency results. A more detailed test documentation than the one given in this chapter can be found in Appendix B.

5.1. Sparse Grid Reference Results

The results documented in this section serve as a comparison to the sparse grid implementation in [**1**] and were used as callibration values. Both the Henon Map and the Jump Map tests show adequate results as given below.

Table 5.1.: Non-adaptive grid results for the Henon Map

Training Values	Grid Level	Training Error	Testing Error
50	3	0,003446677	0,006579311
500	6	0,000127079	0,000336158
5000	7	0,000117518	0,000128859

Table 5.2.: Adaptive grid results for the Jump Map

Training Values	Grid Level	Training Error	Testing Error
5000	3	0,1312085935	0,1330364704
5000	4	0,0990780249	0,1035319887
5000	5	0,0801691034	0,0864726977

5.2. Financial Timeseries Forecasting Results

The prediction uses daily adjusted close data in the four time frames and the parameters described in chapter 4. Retraining had little to no effect on the prediction accuracy, most likely because of the limited amount of data available when using daily stock prices, and will therefore not be included in this chapter. However, it would be interesting to see how this mechanic would perform on hourly data. Additionally, only an excerpt of the results are shown to demonstrate the conclusions drawn from each test. A full accounting of the test results can be found in the appendix.

5.2.1. S&P 500 Stock Prediction

This chapter provides a detailed documentation of the obtained S&P 500 prediction results. First, the correlation of the companies embedding dimensions will be explained. The results for non-adaptive and adaptive grids respectively, will then illustrate the effect of the various grid parameters on the forecast.

Embedding Dimension

The embedding dimension has to be calculated for each individual timeseries as described in chapter 1. A reasonable assumption for the companies in the S&P 500 index, would be that their embedding dimensions are similar, if not the same. The table below proves that the assumption was correct.

Table 5.3.: Overview of S&P 500 embedding dimensions

Embedding Dimension	# Stocks with the dimension
1	2
2	37
3	11

Regression Parameter Optimization

The regression parameter used in the grid solver plays a substantial role in the accuracy of the prediction. The parameter was tested for all timeframes and the grid levels $3-5$. This selection is based upon the regression parameters that showed the best results in other sparse grid applicatons.

Overall, the choice of 10^{-4} performed best. In the following parts of this chapter, only the results obtained for this regression parameter will be illustrated. The results for timeframe 3 are shown below with the regression parameter given as 10^{-x}.

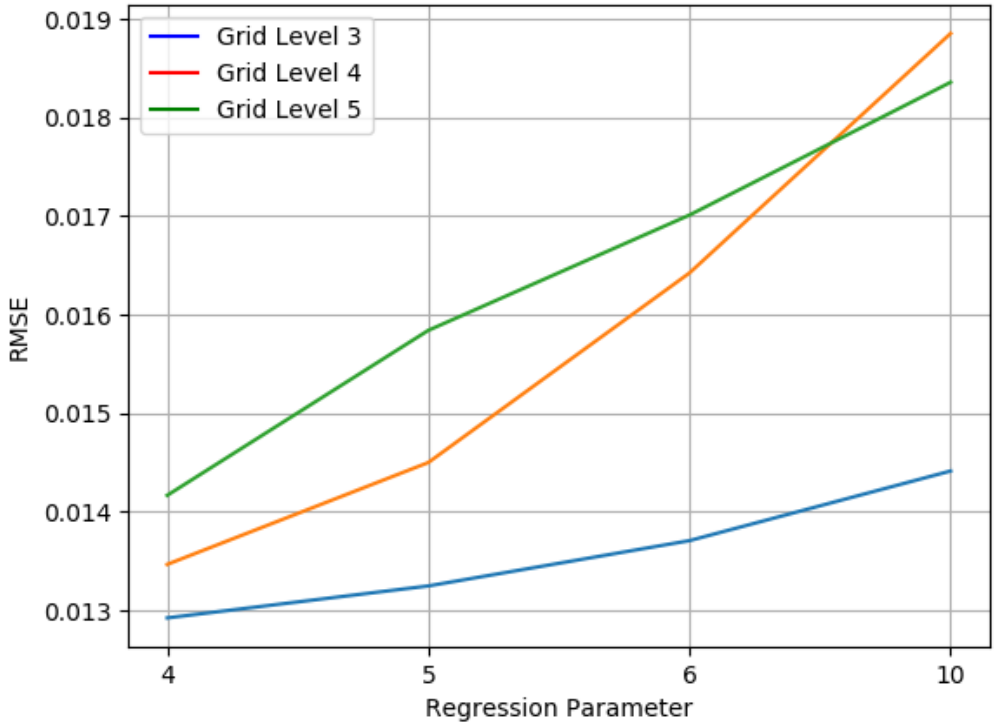


Figure 5.1.: Regression Parameter Optimization for timeframe 3

Using all 500 companies' data for training a grid that is then used to predict individual stocks listed in the index could provide further understanding of this correlation.

Non-Adaptive Grid

The results for non-adaptive grids show RMSEs in the expected range of 10^{-2} to 10^{-3} for the amount of training data used. However, the results for higher grid levels are not as accurate as expected. This topic was adressed in [**1**] and is most likely a result of overfitting, as the grid level 3 exhibits no unusual behavior.

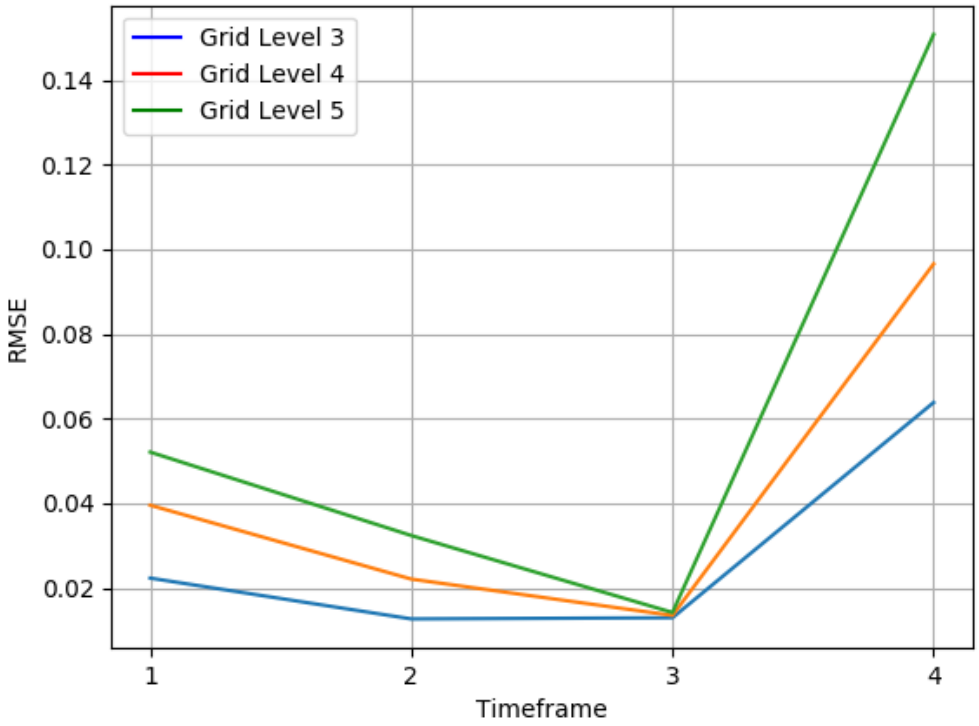


Figure 5.2.: Non-adaptive grid test results for the S&P 500 index companies

As can be seen in the above figure, the results are significantly better for grid level 3 in all timeframes except the third. This provides evidence of the assumed overfitting problem. Furthermore, it is clear that the number of training data used considerably impacts the accuracy of the prediction.

Adaptive Grid

The use of regular sparse grids does not always provide the best results, since the grid points are not distributed according to the underlying process. It is therefore a viable approach to start with a lower grid level, which means fewer initial points, and adapt the grid as explained in chapter 2.

In the adaptive grid tests, only grid levels 2 and 3 were used. The amount of points is specified by the adaptivity rate, which was set to 10% for each iteration, with a maximum of 5 iterations performed. The specific points adapted are selected by the error threshold, for which the three values $(0.1, 0.01, 0.001)$ were tested.

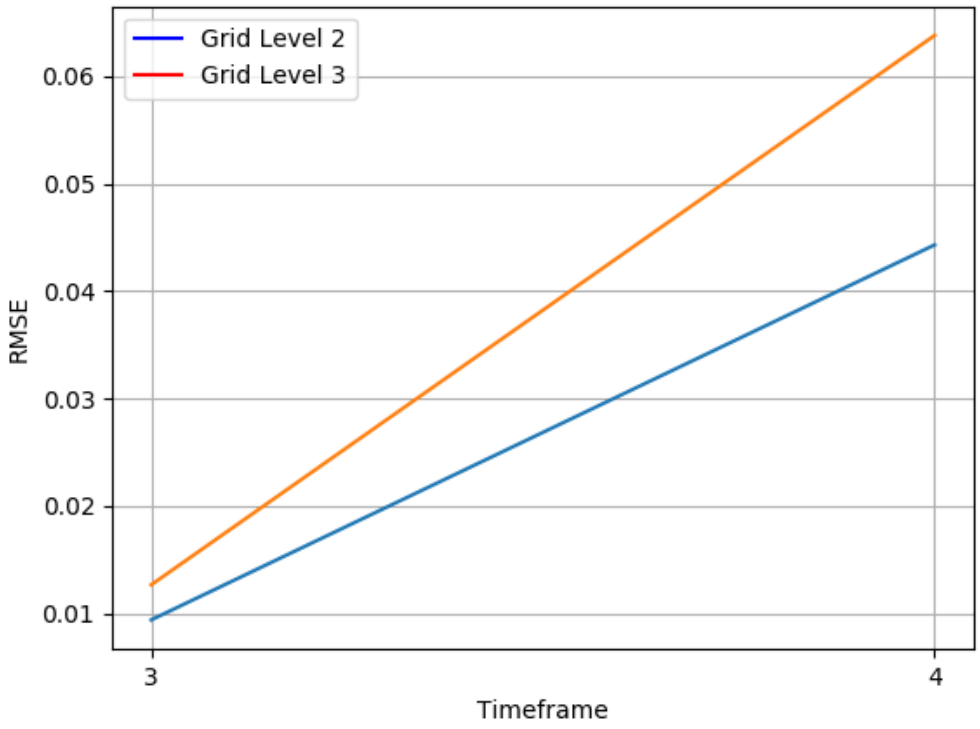


Figure 5.3.: Adaptive grid test results for the S&P 500 index companies and an adaptivity rate of 0.1

A side by side comparison demonstrates the benefit of using a spatially adaptive grid for timeframes with fewer training points. Even for the timeframes with a larger dataset used for training, the adaptive grid is at least as accurate as the non-adaptive version with a lower number of points used.

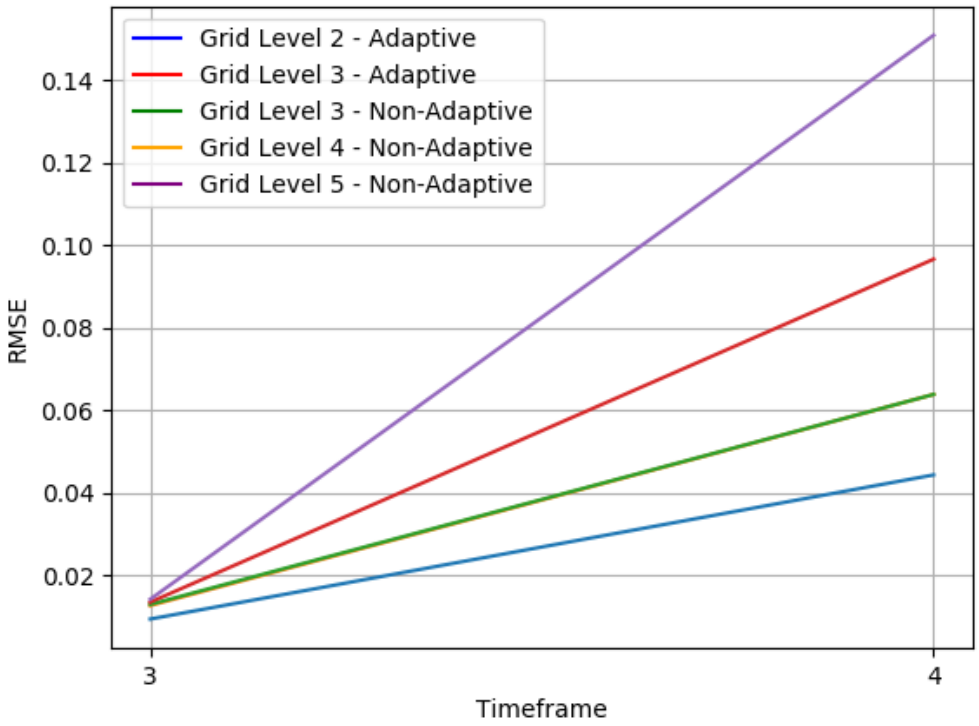


Figure 5.4.: Comparing non-adaptive and adaptive grid test results for timeframes 3 and 4

The highest accuracy was achieved for an adaptivity rate of 0.1. The following figures illustrate that the remaining error for most grid points is too small to be reasonably adapted with the selected thresholds. Choosing even lower thresholds may again lead to overfitting (see adaptivity rate 0.001 in the figures below) and will most likely not give a significant increase in accuracy.

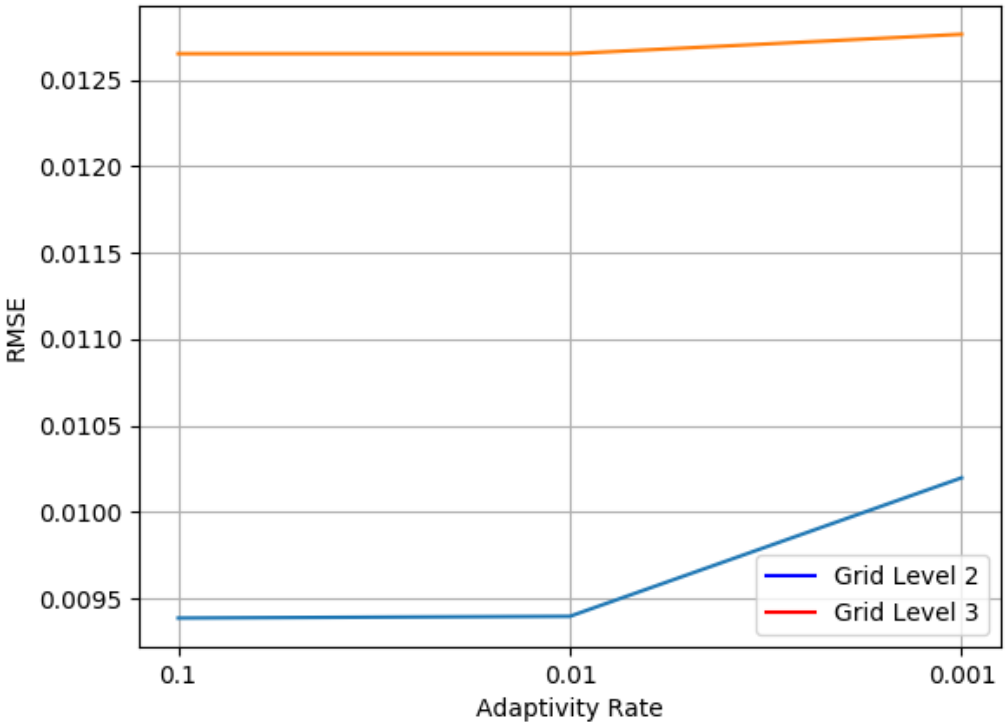


Figure 5.5.: Impact of different adaptivity rates on timeframe 3 test errors

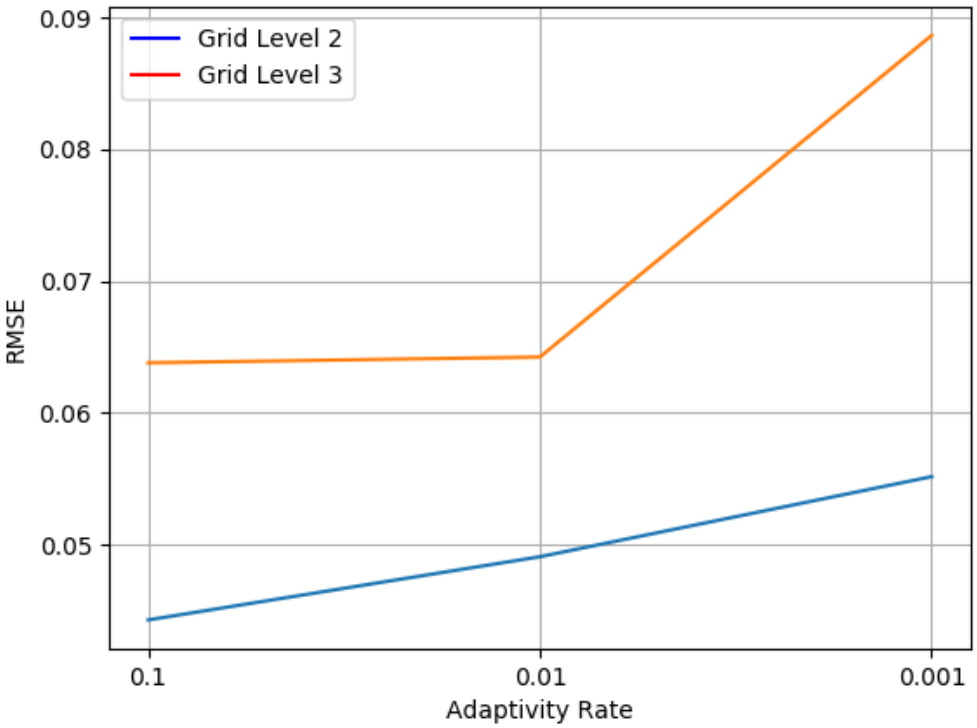


Figure 5.6.: Impact of different adaptivity rates on timeframe 4 test errors

5.2.2. Crypto Currency Value Prediction

Unlike expected, volatility seems to have no negative impact on the forecast. In fact, the errors are slightly better than those of the respective S&P 500 tests. Grid level 5 shows a substantial increase in accuracy compared to the S&P 500 error. However, this could be a cause of the small sample set of two currencies.

Non-Adaptive Grid

Table 5.4.: Non-adaptive grid for timeframe 4

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,036881068	0,073282997
4	4	0,036325440	0,081726369
5	4	0,035026696	0,085934471

Adaptive Grid

Table 5.5.: Aadaptive grid for timeframe 4

Grid Level	Adaptivity Threshold	Training Error	Testing Error
2	0,1	0,039488687	0,053472128
2	0,01	0,039445190	0,058724438
2	0,001	0,039445190	0,058724438
3	0,1	0,036881068	0,073282997
3	0,01	0,036608683	0,068067743
3	0,001	0,035099903	0,068488273

6. Conclusions

In the process of this paper, we reviewed the theoretical concepts of sparse grids and the delay embedding as introduced by Taken's Theorem. We then took an in-depth look at the stock market, its history and the datasets that were analyzed in the forecast. Finally, the pipeline implementation with the SG++ library was outlined and the results explained. After reading this thesis the reader should have a good understanding of a timeseries prediction with sparse grids. Moreover, the complex dynamics of financial markets and their influence on the prediction should be clear.

The results obtained are in the expected range for lower grid levels. The adaptive grid results for predicting the 50 S&P 500 index company stock values are surprisingly good for the fourth time frame, considering the small amount of training data. The error values for higher grid levels are a bit larger than expected. This is a known issue with regular sparse grids as the points are not distributed according to the underlying process and can therefore lead to overfitting. This can mostly be remedied by using an adaptive grid.

While this paper demonstrated the viability of using sparse grids to predict financial timeseries, it should be stated that a number of possible improvements come to mind. For one, using daily stock data only provided limited training points - it would be better to use hourly data. However, this is not freely available. Additionally, the errors have only been calculated in a theoretical setting. It would be interesting to see how they forecast fares in an investment scenario such as the one in the Master's Thesis using long short-term memory recurrent neural networks [**9**]. The tests on cryptographic currencies showed no significant influence of volatility on the prediction. Should this become an issue with a larger testing set, a number of methods from the field of technical analysis exist to analyse trends and calculate volatility. These could be included in the preprocessing.

The goal of an algorithm is to provide an accurate approximation of future values by reducing the error. Unfortunately, this does not allow for an analysis of the underlying process. Nevertheless, by transforming various information streams into timeseries and testing their combined accuracy, it should be possible to build a good model of

financial markets and significantly increase the accuracy as well as open up new fields of forecasting applications. In light of the recent reawakening of machine learning, thanks to increased computational capacities, it will be of particular interest to follow new developments in the area of financial predictive analytics.

Appendix

A. List of the used S&P 500 companies

The 50 used companies are given as stock tickers, under which they can be easily be found.

- AAPL
- ABT
- BCR
- CVC
- COG
- AMZN
- CAM
- CPB
- C
- CAH
- HSIC
- CELG
- CTL
- CERN
- STZ
- GLW
- INTC
- COST
- CCI
- CSX
- DVA
- DE
- GOOG
- DLPH
- DG
- D
- LEN
- LMT
- LOW
- LYB
- MA
- MAT
- R
- SNDK
- LUV
- SBUX
- UPS
- URI
- VIAB
- V
- VNO
- ZION
- PCG
- SWN
- AVGO
- EBAY
- NDAQ
- EQIX
- MCHP
- PPL

B. Detailed Results

B.1. Timeframes

Table B.1.: Prediction Timeframes

#	Learning	Prediction
1	01.01.2000 - 31.12.2015	01.01.2016 - 31.12.2016
2	01.01.2000 - 31.12.2015	01.01.2016 - 31.01.2016
3	01.01.2010 - 31.12.2015	01.01.2016 - 31.12.2016
4	01.01.2015 - 31.12.2015	01.01.2016 - 31.01.2016

B.2. S&P 500 Results

The S&P 500 results are average RMSE values for 50 randomly selected companies. For the adaptive tests a regression parameter of 10^{-4} and an adaptivity rate of 10% was used.

Table B.2.: Non-adaptive grid for timeframe 1

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,00858712	0,02229880
3	5	0,00843852	0,02131000
3	6	0,00837369	0,02181785
3	10	0,00829644	0,02342339
4	4	0,00831583	0,03952849
4	5	0,00811849	0,06000509
4	6	0,00793946	0,10439847
4	10	0,00782095	0,27183828
5	4	0,00792747	0,05208400
5	5	0,00759326	0,07521914
5	6	0,00733413	0,13057265
5	10	0,00728406	0,19629566

Table B.3.: Non-adaptive grid for timeframe 2

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,00611713	0,01265011
3	5	0,00594373	0,01103323
3	6	0,00587932	0,01135053
3	10	0,00581576	0,01574689
4	4	0,00591812	0,02206343
4	5	0,00574139	0,02404011
4	6	0,00561684	0,04040327
4	10	0,00553722	0,10071518
5	4	0,00566598	0,03235783
5	5	0,00543406	0,05009903
5	6	0,00527897	0,09921640
5	10	0,00524652	0,15173385

Table B.4.: Non-adaptive grid for timeframe 3

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,006493711	0,012925385
3	6	0,006258973	0,013709262
3	10	0,006187836	0,014415688
4	4	0,006280444	0,013468127
4	5	0,006081342	0,014501443
4	6	0,005946111	0,016424088
4	10	0,005871322	0,018853949
5	4	0,006001652	0,014168259
5	5	0,005758367	0,015841998
5	6	0,005593788	0,017011666
5	10	0,005561506	0,018358696

Table B.5.: Non-adaptive grid for timeframe 4

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,02631322	0,06380292
3	5	0,02519292	0,07846546
3	6	0,01914764	0,26829837
3	10	0,02386670	0,52747927
4	4	0,02156441	0,09656290
4	5	0,01870198	0,18334964
4	6	0,01626391	0,37691858
4	10	0,01544332	0,63879420
5	4	0,01324588	0,15080361
5	5	0,01018476	0,14819475
5	6	0,00791513	0,23789700
5	10	0,00766509	0,29050998

Table B.6.: Adaptive grid for timeframe 3

Grid Level	Adaptivity Threshold	Training Error	Testing Error
2	0,1	0,006228528	0,009385548
2	0,01	0,006224242	0,009395658
2	0,001	0,006086274	0,010196915
3	0,1	0,006117134	0,012650110
3	0,01	0,006117134	0,012650110
3	0,001	0,005995181	0,012762614

Table B.7.: Aadaptive grid for timeframe 4

Grid Level	Adaptivity Threshold	Training Error	Testing Error
2	0,1	0,029827988	0,044302965
2	0,01	0,028149638	0,049094915
2	0,001	0,025296664	0,055167822
3	0,1	0,026313226	0,063802929
3	0,01	0,025828605	0,064240769
3	0,001	0,018480490	0,088633317

B.3. Cryptographic Currency Results

The crytopgraphic currency results are average RMSE values for Bitcoin and Ethereum. For the adaptive tests a regression parameter of 10^{-4} and an adaptivity rate of 10% was used. Only timeframe 4 was evaluated.

Table B.8.: Non-adaptive grid for timeframe 4

Grid Level	Regression Parameter as 10^{-x}	Training Error	Testing Error
3	4	0,036881068	0,073282997
4	4	0,036325440	0,081726369
5	4	0,035026696	0,085934471

Table B.9.: Aadaptive grid for timeframe 4

Grid Level	Adaptivity Threshold	Training Error	Testing Error
2	0,1	0,039488687	0,053472128
2	0,01	0,039445190	0,058724438
2	0,001	0,039445190	0,058724438
3	0,1	0,036881068	0,073282997
3	0,01	0,036608683	0,068067743
3	0,001	0,035099903	0,068488273

Bibliography

[1] B. Bohn, M. Griebel. *An adaptive sparse grid approach for time series prediction*, 2012. Pages 3-6; 9 - 13.

[2] F. Zipperle. *Density-based clustering with periodic adaptive sparse grids*, 2014. Pages 6-10.

[3] T. Gerstner, M. Griebel. *Sparse Grids*, In Encyclopedia of Quantitative Finance, R. Cont (ed.), Wiley, 2008. Pages 1- 3.

[4] G. Kember and A.C. Fowler. *Ramdom sampling and the Grassberger—Procaccia algorithm*, In Physics Letters A 161, 1992. Pages 429—432.

[5] E. F. Fama. *Efficient Capital Markets: A Review of Theory and Empirical Work*, In Journal of Finance, Volume 25, Issue 2, Papers and Proceedings of the Twenty-Eighth Annual Meeting of the American Finance Association New York, N.Y. December, 28-30, 1969 (May, 1970). Pages 383-417.

[6] J. Clarke, T. Jandik, G. Mandelker. *The Efficient Markets Hypothesis*, 2001. Pages 1—7.

[7] C. Thomas. *Chaos Theory versus the Efficient Market Hypothesis in Financial Markets*, 2002. Pages 4—16.

[8] S. Nordin, N. Nordin. *The Impact of Capital Market on Economic Growth: A Malaysian Outlook* In International Journal of Economics and Financial Issues, Vol. 6, 2016. Pages 259—261.

[9] Q. Gao. *Stock Market Forecasting Using Recurrent Neural Network*, 2016. Pages 6—25; 26; 30-35; 40.

[10] S. Lebert. *Application of Machine Learning Techniques on the Estimation of the Phillips Curve*, 2017. Pages 3—7; 15-21; 37-50.

[11] S. Nakamoto *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

[12] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*, 2010. Pages 12 - 16

[13] G. Mattarocci. *Market characteristics and chaos dynamics in stock markets: an international comparison*, 2006. Pages 2 - 5

Websites

[14] *http://www.scholarpedia.org/article/Grassberger-Procaccia_algorithm*, visited on 01.10.2017.

[15] *http://www.investopedia.com/articles/07/stock-exchange-history.asp*, visited on 15.09.2017.

[16] *https://www.library.hbs.edu/hc/ssb/history.html*, visited on 15.09.2017.

[17] *http://financial-dictionary.thefreedictionary.com/Volatility*, visited on 16.09.2017.

[18] *http://www.multpl.com/inflation-adjusted-s-p-500*, visited on 02.09.2017.

[19] *https://www.buybitcoinworldwide.com/price/*, visited on 02.09.2017.

[20] *http://www.visualcapitalist.com/all-of-the-worlds-stock-exchanges-by-size/*, visited on 02.09.2017.

[21] *http://www.businessdictionary.com/definition/forecasting.html*, visited on 26.09.2017.

[22] *D. Pflüger, Spatially Adaptive Sparse Grids for Higher-Dimensional Problems. Verlag Dr. Hut, München, 2010. ISBN 9-783-868-53555-6*, visited on 27.09.2017.

[23] *https://www.quandl.com/*, visited on 01.07.2017.

[24] *https://etherscan.io/chart/etherprice*, visited on 01.07.2017.