CHARLES UNIVERSITY IN PRAGUE

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Forecasting stock market returns and volatility in different time horizons using Neural Networks

Bachelor thesis

Author: **Martin Hronec**

Supervisor: **PhDr.Jozef Baruník Ph.D.**

Prague 2015

Bibliographic note

HRONEC, Martin. *Forecasting stock market returns and volatility in different time horizons using Neural Networks.* Prague 2011. 94 pages, Bachelor thesis. Charles University in Prague, Faculty of Social Sciences, Institute of Economical Studies. Supervisor: PhDr. Jozef Baruník Ph.D.

Abstract

This thesis is focused on multiple-step-ahead forecasting of Nasdaq Composite index returns and daily range-based volatility. In order to capture the complex patterns potentially hidden to traditional linear models we use artificial neural networks as nonlinear, nonparametric and robust forecasting tool. We contribute to the ongoing discussion about stock market predictability with following empirical results. In case of Nasdaq Composite returns, all four applied neural networks fail to outperform benchmark model in all time horizons, suggesting high unpredictability in accordance with Efficient market hypothesis. Also in case of Nasdaq Composite daily range-based volatility, 1 day and 1 month ahead predictions are not significantly more accurate than benchmark model. However, we find 1-week and 2-weeks-ahead forecasts to be significantly more accurate than benchmark model and able to capture the predictive patterns.

Keywords

predictability of stock returns, predictability of daily range-based volatility, multiple-step-ahead forecasting, neural networks, RPROP, BFGS learning algorithm

Abstrakt

Tato práce se zaměřuje na predikováni výnosů a denní volatility akciového indexu Nasdaq Composite ve více časových horizontech. Aby bylo možné zachytit komplexní vztahy, které mohou být potenciálně skryty pro tradiční lineární modely, budeme používat neuronové sítě jako nelineární, neparametrické a robustní nástroje pro predikci. Námi dosažené empirické výsledky přispívají k probíhající diskuzi o předvídatelnosti akciovách trhů. V případě výnosů Nasdaq Composite, žádne ze čtyř neuronových sítí nedokázely překonat benchmark model na žádnem časovém horizontu, což naznačuje nepředvídatelnost v souladu s hypotézou efektivních trhů. Stejně tak v případě denní volatility Nasdaq Composite nejsou denní ani měsíční předpovědi výrazně přesnější než benchmark model. Avšak, jednotýdenní a dvoutýdenní předpovědi jsou výrazně přesnější než benchmark model a jsou schopny zachytit přítomné předpovědní vzorce.

Klíčová slova

předvídatelnost akciových výnosů, předvídatelnost denní volatility, predikováni v různych časových horizontech, neuronové sítě,
RPROP, BFGS učící algoritmus

Range of thesis: 94 pages, 82 936 characters

Declaration of Authorship

I hereby proclaim that I wrote my bachelor thesis on my own under the leadership of my supervisor and that the references include all resources and literature I have used.

I grant a permission to reproduce and to distribute copies of this thesis document in whole or in part.

Prague, May 12, 2015

Signature

Acknowledgment

I would like to express my deep gratitude to my supervisor PhDr. Jozef Baruník for his guidance, useful comments and willingness to help at any time.

I am also very grateful to my parents for the continuous support throughout my studies.

Last but not least, I owe thanks to my girlfriend for her support during the work on the thesis.

Bachelor thesis proposal

The Efficient Market Hypothesis is one of the most controversial propositions in financial economics. This concept which asserts that the changes in price of financial instruments are unforecastable is not only of vital theoretical importance because of the role that financial markets play in society but also because of possibility of huge profits. However Hinich and Patterson (1985) were among the first to report nonlinearity in daily stock returns of the NYSE. Since then big part of research of financial economists and econometricians was aimed at this topic. Some of them started to emphasize that stock returns are predictable to some degree in different time horizons. In our research we would like to employ machine learning methods as nonlinear and nonparametric methods to search for those above mentioned nonlinearities.

We will begin with defining the prediction task, enlisting machine learning concepts that will be used and choosing different time horizons. Then we will discuss theoretical framework of stock returns predictability and present Efficient Market Hypothesis.

In the second chapter we will look more deeply at different machine learning methods that will be used for prediction, e.g. Back-Propagation Neural Network, Recurrent Neural Network, Support Vector Machines, Genetic Algorithms, etc. We will also discuss possible problems with our methodology and Black-Box criticism.

In the third chapter we will discuss indicators(attributes) used in predicting, e.g. P/E, P/B, Volume, Beta, ROA, ROE, 52W Range, etc. We will use to our advantage ability of machine learning methods to handle high dimensional data very well.

The fourth chapter will contain the applications of earlier mentioned methods to NYSE and NASDAQ stock returns. We will do so over different time frameworks, since predictability of stock returns might be found only in some of them.

The thesis will conclude with the last chapter summarizing achieved empirical results and suggestions for further research.

Main Sources:

Hinich, Melvin J & Patterson, Douglas M (1985):Evidence of Nonlinearity in Daily Stock Returns. Journal of Business & Economic Statistics, American Statistical Association, vol. 3(1), pages 69-77.

Qingqing Chen, Yongmiao Hong(2010): Predictability of Equity Returns over Different Time Horizons: A Nonparametric Approach.

Paul D. McNelis(2005): Neural Networks in Finance: Gaining Predictive Edge inthe Market. Academic Press Advanced Finance Series. ISBN 01-2485-967-4.

Fama E. F., French K. R.(1982): The Cross-Section of Expected Stock Returns. The Journal of Finance.

Kyoung-jae Kim(2005): Artificial neural networks with evolutionary instance selection for financial forecasting. Expert Systems with Applications: An International Journal.

Contents

List of Figures

List of Tables

Introduction

"Forecasting is like driving a car blindfolded with help from someone looking out the rear window."
Anonymous

Stock market forecasting is a difficult domain. Market participants, as learning agents, create environment that can never be completely predictable and in spite of or maybe because of this fact, question of predictability seems to be at the centre of an academic research. There are two variables that stand out in terms of importance for both academics and traders; *returns* and *volatility*. Returns as a measure of profit and volatility as a measure of risk.

The basic assumption in discussion about predictability of stock markets is the repetitive tendency of historical patterns. Prospect of abnormal returns from laymen's perspective is truly attention grabbing, investing gurus, technical analysts, trading wizards and other nonsense vendors got it all figured out. For the rest of us the fact that for a long time professionals tried to beat the market and most of them failed should serve as warning.

From academics perspective, rationale for using predictive models in task of stock market forecasting is different. Methods and models used may increase the body of knowledge, both empirical

and theoretical, about markets behaviour, it's evolution and role in our society. It is generally accepted that level of predictability depends on the time horizon used in forecast. In case of returns, the extent of predictability seems to grow with the length of the horizon and volatility tends to be highly persistent or with the long memory. However, there is no evidence that these traces of predictability are in form of linear relationships and yet, most of the classical methods used in forecasting are linear in their nature. Therefore, as many before us, notably Zhu et al. (2007) and Roh (2007), we use *artificial neural networks* as nonlinear, nonparametric and robust method for predictive patterns recognition. Neural networks do not require, for other econometric models very necessary, assumptions about the characteristics or statistical distribution of underlying data. This feature puts them among the prime candidates for application in financial time series, since fat-tailness, nonstationarity, nonlinearity and other not easy-to-model characteristics are often present.

The main objective of this thesis is analysis of predictability stock market returns and daily range-based volatility in four time horizons: 1 day, 1 week, 2 weeks and 1 month ahead. We expect superior forecasting performance of examined neural networks compared to benchmark model. The structure of this thesis is following:

In the first chapter we present background framework for both volatility and returns forecasting. In case of returns, martingale model, random walk hypothesis and efficient market hypothesis and in case of volatility, range-based volatility estimation and classical volatility models are discussed.

Neural networks are introduced in the second chapter. All the important components such as neuron, activation function, architecture and learning algorithms are properly described and connected. *Backpropagation* alongside more powerful and in this thesis used *resilient propagation & Broyden-Fletcher-Goldfarb-Shanno learning algorithms* explained. Subsequently in the last two sections, numerous neural networks applications, both in stock returns and volatility forecasting, are reviewed.

The third chapter deals with statistical tests, data preprocessing methods and models evaluation criteria used in the empirical part.

The last chapter starts with description of Nasdaq Composite returns and volatility time series. Then we use four different neural networks for forecasting of returns and volatility in above mentioned four time horizons. Finally, achieved empirical results and their interpretation are presented and followed by brief discussion about possible improvements of this work.

1 Forecasting of stock market

This chapter consists of two main parts: forecasting of stock market returns in section (1.1) and forecasting of stock market daily volatility in section (1.2). These two sections correspond to two forecasting tasks undertaken in this thesis.

1.1 Forecasting of stock market returns

Let P_t be the price of a stock or value of a stock index[1] at time t, where $t \in \{1, \ldots, T\}$. One period simple return is then defined as

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{1.1.1}$$

There are several reasons for using returns instead of prices. The most important one is that returns are comparable, i.e. using returns we are able to analyse relationships between different stocks with unequal prices by using the same metric. Hence, for traders, returns offer scale-free measure of the potential investment opportunity. Furthermore, statistical properties of returns time series fit many assumptions necessary for econometric analysis better than prices.

Here we present few essential stylized facts about stock returns time series. These facts offer us glimpse at difficulty of predic-

[1]Stock price and stock index value are used interchangeably in this text, it should be obvious where distinction is important and where it is not.

tion tasks in stock returns and show us what we can expect from datasets used in this thesis.

- Stock returns time series are noisy, i.e. they often behave random-like, thus displaying lack of predictive patterns.
- Statistical attributes of stock returns time series change with time.
- Stock returns are usually *leptokurtic*[2]
- Volatility, i.e. variation of price, usually forms clusters, discovered and explained by Mandelbrot (1963) as: "Large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes."

Next, in section (1.1.1) we present the probable origins of financial asset pricing.

1.1.1 Martingale pricing model

The roots of financial asset pricing can perhaps be traced to the one of the founders of probability theory and professional gambler[3] Gerolamo Cardano. His ideas led to the *martingale model* based on the idea of a fair game, where the stochastic process

[2]Random variable is leptokurtic when its kurtosis is greater than 3, i.e. it has positive *excess kurtosis*. Distribution of such variable has more narrow peak and fatter tails than normal distribution.

[3]Also mathematician, physician, philosopher and astrologer.

$\{P_t\}_{t=0}^{\infty}$ satisfies

$$E[P_{t+1}|\mathcal{F}_t] = P_t \tag{1.1.2}$$

and P_t is a stock price at time t, $\mathcal{F}_t = \sigma\{P_t, \ldots, P_1\}$ is filtration at time t, i.e. information set that contains all historical but no future information about stochastic process $\{P_t\}_{t=0}^{\infty}$. Thus tomorrow's expected price given the historical prices from information set equals today's price. Equation (1.1.2) implies

$$E[P_{t+1} - P_t|\mathcal{F}_t] = E[r_{t+1}|\mathcal{F}_t] = 0 \tag{1.1.3}$$

where r_{t+1} is stock price change and it is equal to zero. Hence changes in price are unpredictable, given the historical prices, and uncorrelated at all lags. Therefore one cannot effectively find any linear or non-linear relationships and all forecasting attempts must be ineffective.

However, if expected return was equal to zero and no forecasting rules could be applied, why would anyone participate in trading? With this important question in mind we present the *random walk hypothesis* in section (1.1.2) and it's successor the *efficient market hypothesis* in section (1.1.3).

1.1.2 Random walk hypothesis

The first explicit mention of randomness in the behaviour of stock market prices can be found in the work of French economist and broker Jules Regnault (1863), who concluded that: "The de-

viation of prices is directly proportional to the square root of time". Using Regnaults hypotheses Bachelier (1900) was first to use Brownian motion[4] in stock options evaluation. Even though Bacheliers results were ignored in academia for 50 years, Cowles (1944) in his empirical research on performance of professional investors showed that generally not even them can consistently outperform the market. Kendell (1953) discovered that prices behave as if wandering, i.e. they move randomly. Cootner (1964) published Bacheliers work with the empirical studies and Fama (1965) in his Ph.D. thesis both supported random walk hypothesis.

Formally one can say that a time series P_t follows a *random walk* if

$$P_t = P_{t-1} + \epsilon_t; \quad t = 1, 2, \ldots, T \tag{1.1.4}$$

where $\epsilon_t \sim N(0, \sigma^2)$ is an i.i.d.[5] disturbance term. Thus,

$$E[P_{t+1} - P_t] = E[\epsilon_t] = 0 \tag{1.1.5}$$

what is very reminiscent of equation (1.1.3) and martingale model. In fact the martingale and the random walk models are very similar but the martingale is less restrictive and weaker because only independence of the conditional expectation of price changes is required, while for random walk model assumption of indepen-

[4]As limiting case of a random walk. For further details please see Morters & Peres (2010).

[5]Independent and identically distributed random variable.

dence for higher conditional moments of the distribution of price changes is necessary.

Later Campbel, Lo & MacKinlay (1997) add three other models of a random walk RW1, RW2 and RW3.
RW1 is a *random walk with drift* model obtained by adding a constant term to equation (1.1.4)

$$P_t = \mu + P_{t-1} + \epsilon_t; \quad t = 1, 2, \ldots, T \tag{1.1.6}$$

It's obvious that $E[P_t - P_{t-1}] = \mu$, since ϵ_t is white noise with zero mean. Therefore μ can be recognized as the *time trend* of P_t.
In RW2 model, assumption about identical distribution of error term is relaxed, allowing unconditional heteroskedasticity. The most general version is RW3 model of price process with uncorrelated but dependent increments.

1.1.3 Efficient market hypothesis

Martingale model, random walk hypothesis and efficient market hypothesis (EMH) are firmly linked together not only historically but also by Samuelson (1965) who proved that prices on the efficient market, i.e. properly anticipated prices, will fluctuate randomly and follow a random walk. However, the proper formulation of EMH emerged only in Fama (1970), who defined three types of market efficiency:

- **Weak form**

  All past publicly available information is reflected in prices of traded assets. The information set consists only of historical prices or returns.

- **Semi-strong form**

  All publicly available information is reflected in prices of traded assets, i.e. prices change instantly to reflect new public information. The information set includes all information known to all market participants.

- **Strong form**

  All public or private information is instantly reflected in prices traded assets. The information set includes all information known to any market participant.

All three forms of EMH describe how information is included in price, nevertheless as Fama explains in the same paper market efficiency can only be tested using expected returns. Fama (1991) also shows that EMH per se is not testable, because one can never be sure whether rejection is caused by market inefficiency or incorrect equilibrium model. This is called *joint hypothesis problem.*

Despite the joint hypothesis problem Fama & French (1989, 1996, 2008), Lo & MacKinlay (1988) and others[6] suggest that markets are neither efficient nor efficient. Main arguments for efficiency are the speed with which prices reflect new available information and general inability of professional investors to outperform the market. On the other hand the strong evidence for inefficiency is so called *momentum effect*, i.e. stocks with bad recent performance tend to perform badly and stocks with good recent performance tend to perform well in the near future, *January effect*, i.e. seasonal anomaly of more frequent positive returns in January, and stock market crashes.

As we will see in chapter (4) we try to exploit the momentum effect by adding lagged returns as explanatory variables to identify good and bad recent performers. For now we present one of the most basic econometric models which will serve as benchmark for comparing performances of studied models.

1.1.4 Autoregressive process

One of the simplest models using predictive power of lagged values is *autoregressive model of order q* or AR(q) defined as

$$R_t = \rho_0 + \rho_1 R_{t-1} + \ldots + \rho_q R_{t-q} + \epsilon_t \tag{1.1.7}$$

where ϵ_t $(0, \sigma_\epsilon^2)$ is white noise and $q \in \mathbf{N}$ is number of included lagged variables.

[6] Also with the help of neural networks as presented in **??**

In this thesis we use AR(1) model

$$R_t = \rho_0 + \rho_1 R_{t-1} + \epsilon_t \tag{1.1.8}$$

as benchmark with which performance of neural networks is compared.

Even with only one included lagged variable as in the case of AR(1) model, a one-time shock in error term has persistent, i.e. far into the future, effect on the random variable. Therefore ϵ_t can be seen as series of shocks that steer the behaviour of corresponding variable.

Mean of a random variable following AR(1) process is

$$\mu = \frac{\rho_0}{1 - \rho_1} \tag{1.1.9}$$

and variance is

$$var(R_t) = \frac{\sigma_\epsilon^2}{1 - \rho_1} \tag{1.1.10}$$

Because we are not sure about ability of neural networks to find hidden patterns in time series of stock market returns we also try to employ neural networks in task of forecasting future volatility. Basic discussion about what volatility is, how can we estimate it and if we can predict it, follows in section (1.2).

1.2 Forecasting of stock market volatility

Volatility, as a measure of price variation is in modern portfolio theory used as a proxy for risk. However, as Gregoriou (2009) notes, reasonable and fundamentally justified volatility doesn't have to be a negative phenomena as it is necessary for efficient price discovery. It's importance can be seen in many areas of financial economics, e.g. derivatives valuation, risk management, etc. Therefore, proper models about volatility behaviour, especially those dealing with predictability are at the utmost importance. The important stylized fact about equity returns is already mentioned volatility clustering. It reflects the leptokurtosis present[7] in returns time series. Bollerslev & Engle (1986) suggest that the main reason behind volatility clustering seems to be changing rate of information arrival and its processing by the market.

Since volatility of any price process is unknown, we can rely only on volatility estimates. There is number of different volatility estimators, where the main differences are in *type of used data*[8] and *efficiency*[9]. Furthermore, *implied volatility* related with option contracts is defined as value of volatility which needs

[7] As explained in section (1.1) and discovered in empirical section (4.1.1).

[8] Intraday data, daily data, etc.

[9] Finite-sample efficiency as a statistical measure of the optimality of an estimator. Estimators with smaller variances are more precise.

to plugged into an theoretical option pricing model in order for it to return the current market price of the option. However, in this thesis we study only only historical volatility calculated from past values of a stock index.

Two of the most popular measures of daily volatility used in forecasting are *squared daily returns*, r_d^2 and *absolute daily returns*, $|r_d|$. Nonetheless, these two estimators use only two consecutive close prices, i.e. information from particular time of the day.

By using daily high H_t and low L_t or open O_t and close C_t stock index values[10] we can extract some additional information about dispersion[11] of returns, i.e. risk in some sense.

1.2.1 Range-based estimators

The main reason for using range-based estimators is their superior ability to capture volatility when compared with squared returns or absolute returns. Intuitively, we can expect that the behaviour (variation) of price is somehow better included in the price range, say over a day, than in the close prices alone. In general we assume the price process P_t evolving over time $t = 1, \ldots, T$ with volatility σ_t, which can be estimated using range-based estima-

[10] Where the high price is the maximum achieved price during the day, the low price is the minimum reached during that day, the open and close are prices at which a security trades first and last during the days trading.

[11] A measure of the deviation from the mean.

tors.

The first range-based estimator of daily volatility σ_t^2 was proposed by Parkinson (1980) and is defined as

$$(\hat{\sigma}_t^{Park})^2 = 0.3607\Big(log(H_t) - log(L_t)\Big)^2 = 0.3607\Big(\frac{log(H_t)}{log(L_t)}\Big) \tag{1.2.1}$$

where $log()$ is natural logarithm. It is claimed that variance of this estimator is five-times lower than the variance of estimator based on daily returns, i.e. Parkinson is five-times more efficient.

Garman & Klass (1980) suggested including daily open O_t and close C_t prices

$$(\hat{\sigma}_t^{GK})^2 = 0.511(log(H_t) - log(L_t))^2 - 0.019\big((C_t - O_t)$$
$$(H_t + L_t - 2O_t) - 2(H_t - O_t)(L_t - O_t)\big) - 0.383(C_t - O_t)^2$$

with claimed efficiency of 7.4 times better then estimators based on daily returns.

Slightly improved estimator is then proposed by Rogers & Satchell (1990) and defined as

$$(\hat{\sigma}_t^{RS})^2 = (L_t - H_t)(L_t - C_t) + (H_t - O_t)(H_t - C_t) \tag{1.2.2}$$

Yet with the unprecedented technological progress the availability of data for the finer sampled high-frequency returns allows approximating volatility by squared intraday returns, called realized volatility. Andersen, Bollerslev, Diebold & Labys (2003) show that realized volatility is an unbiased and highly efficient estimator of return volatility[12], nevertheless because of microstructure errors squared intraday returns are in fact noisy volatility estimates. For more details, please see Bandi & Russel (2006), Hansen & Lunde (2006) and for more on realized daily volatility Pigorsch, Pigorsch & Popov (2011).

Martens & van Dijk (2007) and Christensen & Podolskij (2007) suggested an extreme-value estimation based on intraday price ranges. However, as mentioned in Todorova & Husman (2011), intraday price ranges suffer significantly more from market microstructure effects, e.g. observed high-low range very often overstates the true range by the spread[13].

For the comparison, studies using realized volatility as a benchmark, e.g. Bali & Weinbaum (2005) and Shu & Zhang (2006) provide evidence in favour of the range-based volatility estimators. Therefore, in this thesis we use original σ_d^2 range-based estimator

[12]Under suitable conditions, for more detail see referenced paper.

[13]Due to the fact that the highest price is very likely to be an ask price and the lowest price is very likely to be a bid price.

proposed earlier by Parkinson (1980).

Now it's time to look at time-series models traditionally used in volatility forecasting.

1.2.2 Classical volatility modelling.

Perhaps the simplest historical volatility model is the random walk, i.e.

$$\hat{\sigma_t}^2 = \sigma_{t-1}^2 + \epsilon_t \tag{1.2.3}$$

implying that the optimal forecast of volatility at time t is just realized volatility from previous day. The second, also rather naive model is the historical average, i.e. a long-term average of past standard deviations. Another, similar approach is the moving average, in which case a rolling window of standard deviations of some fixed lenght N[14] is used

$$\hat{\sigma_t}^2 = \frac{1}{N}\sum_{i=1}^{N} \hat{\sigma_{t-i}}^2 \tag{1.2.4}$$

However, the moving average model discards information older than N days and all used historical observations have the same weight[15].

An improved version is an exponentially weighted moving average model called *RiskMetrics*[16] giving higher weight to more

[14]Usually somewhere between 20 to 60 days.

[15]When an influential high observation gets discarded after N days, *ceteris paribus*, volatility drops significantly.

[16]JP Morgan (1996).

recent observations.

$$\hat{\sigma_t}^2 = (1-\lambda)\sum_{i=1}^{N}\lambda^{i-1}\hat{\sigma_{t-i}}^2 \tag{1.2.5}$$

where λ is the decay parameter[17].

As the last classical model we mention here is the famous and often used GARCH model. It enables the modelling of non-constant variances, i.e. conditional heteroskedasticity, and volatility clustering. Formally

$$\begin{aligned} R_t &= \mu + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma_t^2) \\ \sigma_t^2 &= \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta \sigma_{t-1}^2 \end{aligned} \tag{1.2.6}$$

where R_t is a stock return, α_1 is the auto-regressive conditional heteroskedasticity term representing the short-run persistence of previous error terms, i.e. shocks on return variance, and β_t is the GARCH term representing the long-run persistence of older shocks. According to Akgiray (1989), GARCH(1,1) model is able to capture all volatility clustering.

For the comparison of different classical models performance and much more about volatility modelling we suggest Gregoriou (2009).

[17] e.g. $\lambda = 0.94$ in case of daily forecasts and $\lambda = 0.97$ in case of monthly forecasts. From equation 1.2.5 and $N = 75$ is a typical time window.

In this thesis all examined models will be compared with simple AR(1) model, since we are interested only in the predictability and not in the relative comparison of neural networks and standard econometric models. In order to effectively use artificial neural networks later in chapter (4), we devote chapter (2) to brief introduction of the concept.

2 Artificial Neural Networks

Artificial Neural Networks (NNs) are statistical learning algorithms capable of machine learning and pattern recognition. Due to they adaptive nature they serve as a robust method for approximating both discrete and real functions. For deeper understanding of methods concisely described and applied in this thesis as well as greater perspective on the field of NNs we suggest Rojas (1996) and Ke-Lin & Swamy (2013).

In section (2.1) we briefly describe the original motivation in form of biological neural networks and NNs relation to them. The essential building block, i.e. *neuron*, is defined in section (2.2). The concept of architecture of NN as a web of interconnected neurons is presented in section (2.3). Subsequently in section (2.4) we discuss shortly learning in general. Section (2.5) shows, in form of different learning algorithms, adaptation and learning of NN. This chapter ends with sections (2.6) and (2.7) that contain review of NN applications in stock market forecasting, concretely forecasting of stock market returns and volatility.

2.1 Basic biological background

The information processing abilities of biological nervous systems serve as powerful inspiration for artificial neural networks. Nervous systems consist of numerous interconnected neurons of differ-

ent types, e.g. sensory neurons, motor neurons,etc. Basic structure of a typical neuron is depicted in Figure 2.1.1 and consist of a cell body (soma), axon and dendrites.

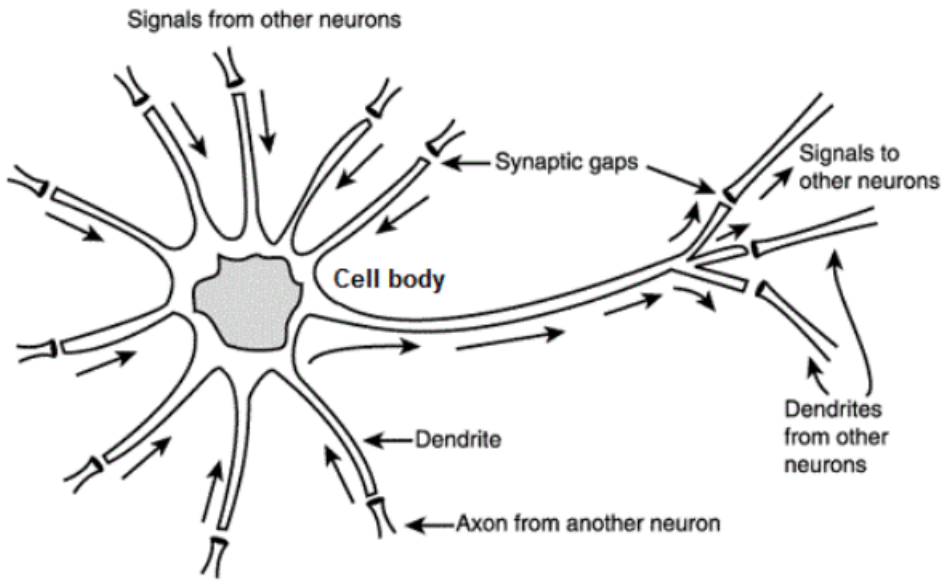

Figure 2.1.1: Structure of a typical biological neuron

Soma is the main part of the neuron and the centre of computational process. Dendrites serve as transmission channels that propagate electrochemical stimulation obtained from other neurons. Axons proceed neural activity in form of electrical impulses away from soma to different neurons.

The most fundamental feature of neurons is their ability to communicate with other neurons what is done via synapses. Synapses are contact points between different neurons and are essential to neural networks. Single neuron might be connected with hundreds of other neurons but these connections are not weighted uniformly. The estimate for number of synapses in adult hu-

man brain is 10^{14}. Architectures of different nervous systems differ substantially in complexity but all are composed of neurons, which act like parallel processors. Figure **??** illustrates the relationships between biological phenomena (pictures A, C) and corresponding theoretical phenomena (pictures B, D).

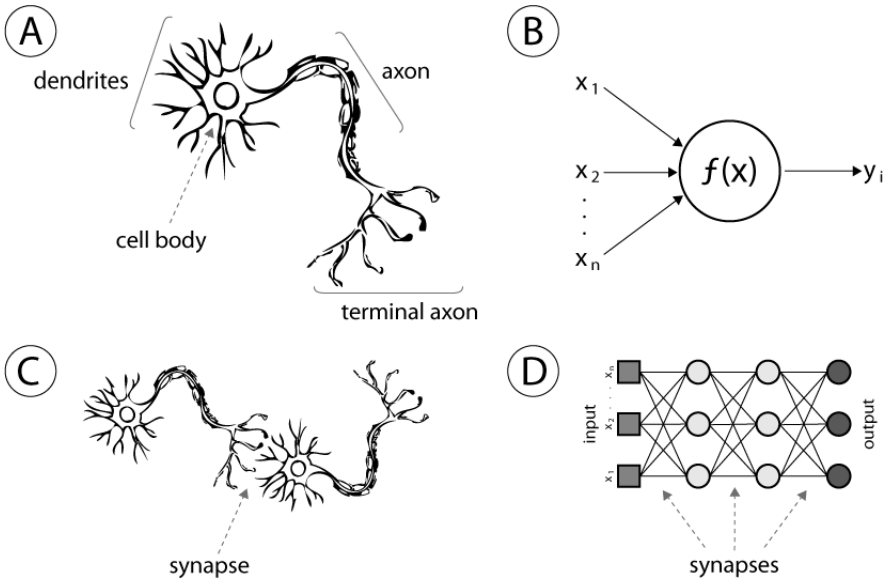

Figure 2.1.2: Biological vs. artificial neurons and neural networks

2.2 Artificial Neuron

Synapses, dendrites, soma and axon all have their theoretical equivalents in the structure of the artificial neuron. As with biological neuron, the artificial one is an information-processing unit and functions as elementary computing element. Figure **??** shows the general model of an artificial neuron.

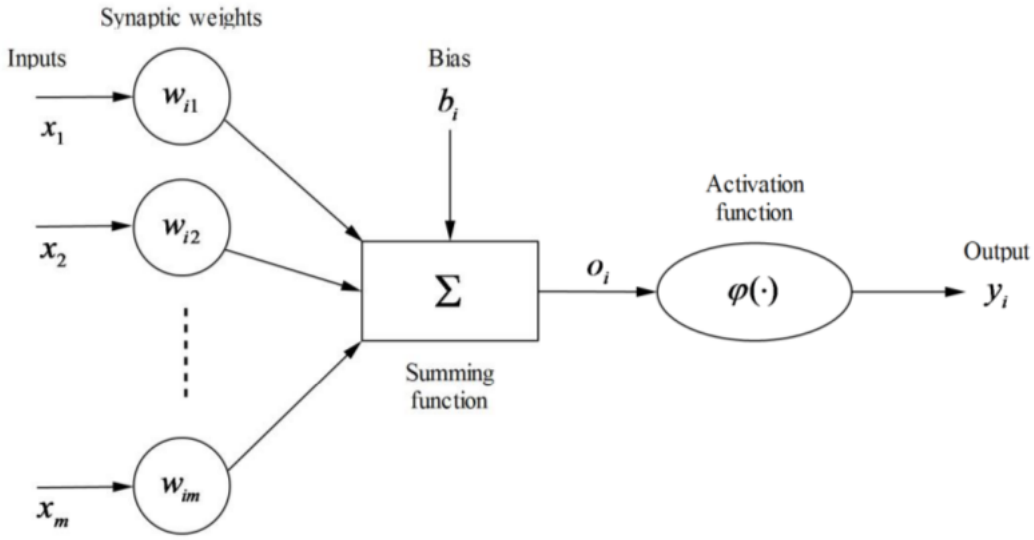

Figure 2.2.1: Artificial neuron

Again as in case of biological neuron the artificial neuron receives the input signal from other neurons or environment represented in form of input variables. Input signal from j-th previous neuron x_j is multiplied by weight w_{ij} representing corresponding synopsis. The neuron sums according to affiliated weights all the inputs, this is basically a linear combination of inputs or the model of linear regression without an error term. Bias term b_i is also added and in that case hyperplane, as geometrical representation of linear combination, does not go through the origin of coordinate system. Since in biological case we have only two possibilities: either no reaction or spike, value of neuron output is restricted by using so called *activation or squashing function*[18], usually between 0 and 1.

The output from i-th neuron can then be defined as:

[18]Various activation functions used in NNs are discussed in the following section (2.2.1).

$$y_i = \varphi(b_i + \sum_{j=1}^{m} w_{ij} x_j) \tag{2.2.1}$$

where $x_1, x_2, \ldots, x_m$ are the neuron inputs, $w_{i1}, \ldots, w_{im}$ are the weights, b_i is the bias term and φ is an activation function.

2.2.1 Activation functions

In this section we present the most important activation functions.

Step function or **threshold function** is defined as

$$\varphi(u) = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases} \tag{2.2.2}$$

where θ is a specified threshold. The neuron spikes, i.e. output of the neuron is equal to one, if the weighted sum of inputs is greater than θ. This also corresponds to the simplest NN model called *binary threshold model* developed by McCulloch & Pitts (1943).

Linear function is just a simple linear function. It is usually used either in the input or output layer of a NN. If all activation functions in the NN were linear it could be reduced to a simple artificial neuron, thus it doesn't make sense to use it in hidden layers.

In order for a NN to capture the *non-linear* patterns in data, non-linear activation functions need to be used at least on some neurons. Most frequently used activation functions are *sigmoid* functions with well known "S" shape. General *sigmoid function* is defined as

$$S(z) = \frac{1}{1 + e^{-z}} \tag{2.2.3}$$

Two of the most used activation functions along with their derivatives that will be later used in learning of artificial neural network are presented next.

Log-sigmoid function is defined as

$$\Lambda(x) = \frac{L}{1 + e^{-x}} \tag{2.2.4}$$

where L is the maximum desired function value, usually $L = 1$. In this case, it's derivative is

$$\Lambda'(x) = \frac{1}{1 + e^{-x}}\Big(1 - \frac{1}{1 + e^{-x}}\Big) \tag{2.2.5}$$

Hyperbolic tangent is defined as

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x + e^{-x}}{e^x - e^{-x}} = \frac{1 + e^{-2x}}{1 - e{-}2x} \tag{2.2.6}$$

where, as can be seen, $sinh(x)$ and $cosh(x)$ are hyperbolic sine and cosine. The derivative is then

$$tanh'(x) = 1 - tanh^2(x) = 1 - \Big(\frac{e^x + e^{-x}}{e^x - e^{-x}}\Big)^2 \tag{2.2.7}$$

The main difference between these two alternatives is that log-sigmoid squashes incoming linear combinations into the interval of $(0, 1)$ and hyperbolic tangent into the interval of $(-1, 1)$. Answer to the question which activation function to choose is not simple. It is generally suggested to try few candidates and choose the one which performs best, however this could be time consuming since NNs don't belong to the fastest of procedures.

2.2.2 Duality of input space and weight space

Now when we have all the pieces of a simple neuron put together we can look at the relationship of input space and weight space, i.e. *duality*.

Let's have a simple neuron with threshold activation function, also called *perceptron*, the (n+1)-dimensional input vector $(x_1, \ldots, x_n, 1)$ and the (n+1)-dimensional weight vector $(w_1, \ldots, w_{n+1})$. Every linear combination of inputs and weights has geometrical interpretation as hyperplane in (n+1)-dimensional space. Since inputs are fixed, in a sense that they are not parameters, the only way how to move our hyperplane is by modifying weights. These n+1 parameters represent a point in (n+1)-dimensional weight space and therefore every vector from weight space defines a hyperplane in the input space.

The same kind of relation holds also in the inverse direction. Let's

say input vector $(x1, x2, x3)$ is supposed to be in the positive half-space of the hyperplane. In order to do so, corresponding weight vector needs to be chosen from the weight space. Figure 2.2.2 illustrates this relationship.

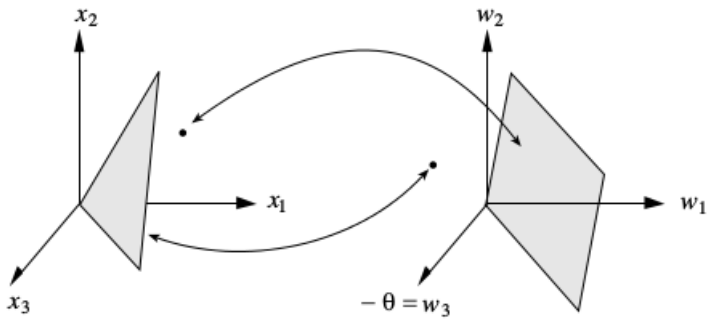

Figure 2.2.2: Duality of input space and weight space

Until now we've presented only the model of a single computing unit, neuron. However, one neuron alone has only limited abilities that correspond to the abilities of e.g. logistic regression in case of log-sigmoid activation function, thus we proceed with interconnecting number of neurons and describing of what's known as *architecture* or *topology* of a NN.

2.3 Architecture of Neural Networks

The building blocks of NN architecture are the computing units neurons and their interconnections affiliated with weights. For our purposes we deal only with *layered* architectures in which all NN's neurons are subdivided into l *layers*. Neurons in a same layer are usually not connected to each other which is also case for NNs used in this thesis. Order of the layers is strictly set

and connections go from input layer to the first hidden layer, from there to the second hidden layer and so on until the last hidden layer and finally to the output layer, where hidden layers are layers with no direct connection from or to the outside, and finally to the output layer. NNs with these types of architectures are called *feed-forward networks* because they don't contain any cycles, i.e. signal travels one way only.

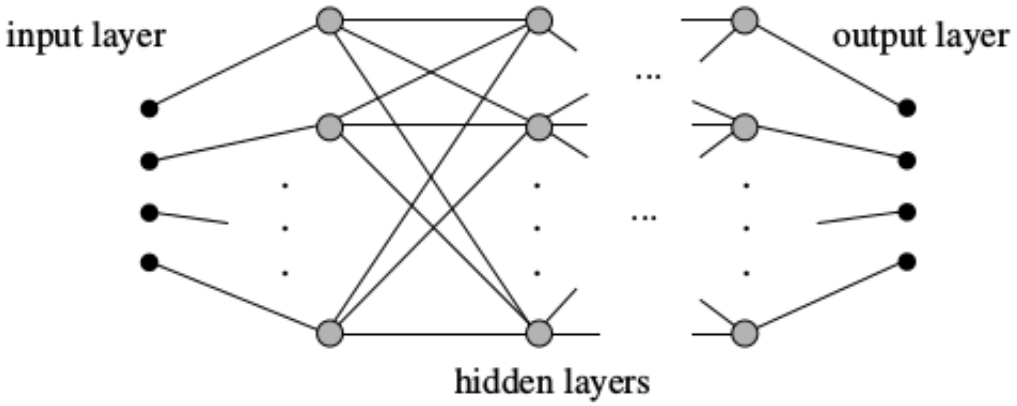

Figure 2.3.1: A generic feed-forward neural network

The most used type of *connectivity* between two layers is *full connectivity*, i.e. all m neurons from one layer are connected to all n neurons in the other layer which gives us mn weights. For NNs with more layers, i.e *deep neural networks*, it can present problem in form of high computational complexity, therefore it is often necessary to prune the network in order to have *sparse connectivity* between the layers. However, great number of layers or neurons does not necessarily mean better performance of NN, since unnecessary complexity likely leads to overfitting and NN

looses ability to generalize well. The evidence in favour of simplicity can be found in Dayhoff & DeLeo (2001). The problem of overfitting will be discussed more in the following section.

Now when brief morphology of NNs is behind us, true "magic" lies ahead. Process which brings NNs "alive"; *learning*.

2.4 Learning

Learning of NNs belongs to the very wide area of machine learning. Machine learning in general deals with algorithms designated to build models that recognize patterns and allow making predictions on data. There are three main learning paradigms: *supervised, unsupervised and reinforcement learning*.

Supervised learning is used on *labeled* data, i.e. each observation consists of an input vector and an output vector. The main task of a supervised learning algorithm is to infer a function from training data that will generalize well on an unseen data. Formally:

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of N training examples, where (x_i, y_i) is the i-th training example. The main task for supervised learning algorithm is to find a function

$$g : X \to Y \tag{2.4.1}$$

where X and Y are input and output space. A *loss function* is

defined as

$$L : Y \times Y \rightarrow \Re^{\geq 0} \tag{2.4.2}$$

and can be calculated for any training example, where $\hat{y} = g(x_i)$. The expected loss of function g can be estimated from collection of training examples in a following way

$$\frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, g(x_i)) \tag{2.4.3}$$

The optimal function g with respect to performance on training dataset can be found by number of different minimization algorithms. However, what is truly important is a performance on unseen data, i.e. testing dataset. Now we come to the main issue of supervised learning; *bias vs. variance dilema.*

The perfect model would be one that captures important patterns in training data perfectly and generalizes well to an unseen data. Usually, as we will also see in the empirical part of this thesis, this is not possible. *Low-bias models* that perform very well on training data are usually those with greater complexity. Unfortunately with greater complexity these models tend to represent a lot of noise what makes them behave poorly on testing data, i.e. their predictions have high variance. On the other hand *high-bias models* are usually much more simple, but usually behave better on testing data, i.e. with lower variance predictions. Low bias and high variance models are also called overfitted and models with inverse problem called underfitted. In Figure 2.4.1

examples of such models are illustrated, where $h(x)$ is an optimal model.

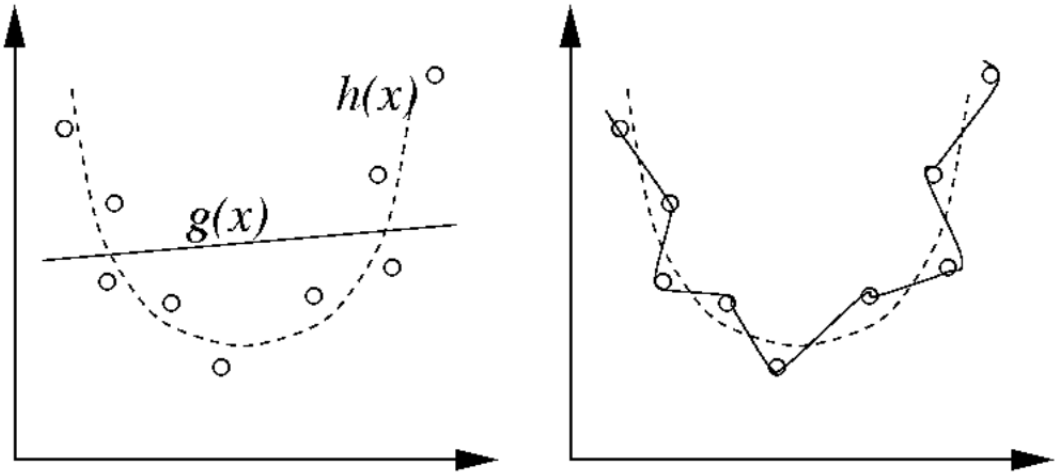

Figure 2.4.1: Underfitted vs overfitted model

In order to prevent overfitting, *early stopping* is employed in this thesis. It consists of dividing data into a *training set* and a *cross-validation set*. NN is trained using only training data but is regularly stopped and tested, i.e. cross-validated, on validation data. As long as NN is learning patterns that enable generalization, error function is decreasing on both training and validation set. Since NNs are universal approximators as stated in the *universal approximation theorem*[19] by Hornik (1991), they learn training data perfectly, unless stopped. This is however undesirable, therefore learning is stopped when error on cross-validation data set starts to grow, as shown in Figure 2.4.2. Weights of NN in this moment are optimal with respect to bias vs. variance

[19] A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\Re^n$, under mild assumptions on the activation function.

dilemma.

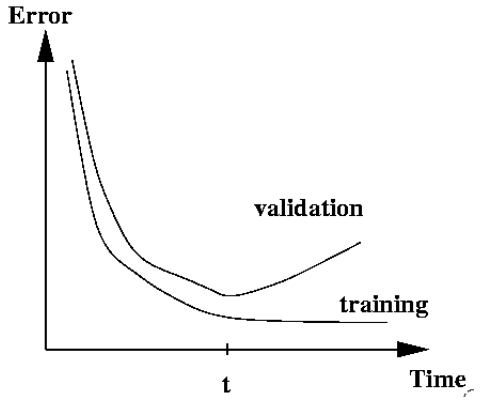

Figure 2.4.2: Early-stop of learning at time t

The other two paradigms are rather distanced from our topic, thus we only briefly mention them. For some deeper insight we suggest excellent work of Bishop (2006).

Unsupervised learning is used on *unlabeled data*. The only way how to compare potential models and then choose the best is with minimization of a *cost function*. The form of cost function is task dependent. The most frequently used applications of unsupervised learning are *clustering, filtering, compression*, etc.

Reinforcement learning occurs when machine learning algorithm is trying to perform some desired behaviour, e.g. drive car, walk, play a game, etc. In this case data are usually obtain by models (agents) interaction with some environment.

With this short introduction to the topic of learning in general,

we are ready to look at the individual learning algorithms in the next section.

2.5 Learning algorithms

Since all the knowledge of a NN is stored in weights[20], *learning algorithm* is strictly speaking a method defining how NN updates its weights. Each neuron serves as computing unit, i.e. mathematical function defined in (2.2.1). A feed-forward NN consists of interconnected layers of neurons and therefore is nothing more than a chain of functions, i.e. function composition, that transforms (maps) vector of inputs to vector of outputs. Inputs and outputs together form training examples and they give us implicitly discrete function f to be approximated by NN. Learning algorithms are therefore methods for finding optimal combination of weights.In the next paragraph we set the framework in which all learning will take place.

We are given T training examples consisting of n-dimensional input vectors $\mathbf{x_t}$ and outputs y_t, i.e. $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_T}, y_T)\}$. Accordingly we choose feed-forward NN with n neurons in input layer and 1 neuron in output layer, number of hidden units and hidden layers is irrelevant for now, nevertheless they are present. All activation functions in neurons are continuous and differentiable, e.g. tan-sigmoid or log-sigmoid, the weights of the con-

[20] As in brains where all knowledge seems to be stored in synapses.

nections between neurons are real numbers. When we feed NN with inputs it outputs o_i which in our case is a forecast. In order to learn all patterns in training data perfectly[21] we would like to have $o_i = y_i$, i.e. estimated output equal to true value.

In the next section (2.5.1) we explain the most common and very intuitive way how to train a NN; *backpropagation* or *backward propagation of errors*. In section (2.5.2) we talk about *resilient backpropagation* or *RPROP* and in section (2.5.3) we deal with *Broyden-Fletcher-Goldfarb-Shanno algorithm*.

2.5.1 Backpropagation

Backpropagation is usually a supervised learning algorithm[22] which used to be the most frequently used learning algorithm employed in training of NNs but seems to be slowly pushed backwards by more powerful learning techniques such as Newton methods, quasi-Newton methods, conjugate gradient, etc.

It consists of two phases, *forward propagation* and *backward propagation*.

Forward propagation transforms inputs into outputs by transmitting output signal of each neuron to the neurons in the next layer until the output layer is reached and the outcome of

[21] Again, this is not what we end up doing, see section (2.4

[22] Although it can be used as unsupervised learning method, e.g. in *autoencoders*, for more details see Bengio (2009).

the NN calculated. The output of the NN is then compared with true value, i.e. expected target value. This comparison is done according to an arbitrary error function E[23] that is function dependent only on weights w_{ij}. It is important to note that during the forward propagation no weights are changed.

Backward propagation Now when the NN output o_i is obtained by forward propagation and achieved error is calculated, in order to decrease it we need to modify the weights according to some specific rule. It's possible by propagating the error signal from output neuron back through the NN. Neuron by neuron the chain rule is used to compute the impact of each weight on chosen error function E

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_i}\frac{\partial o_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}} \tag{2.5.2}$$

where w_{ij} is the weight from neuron j to neuron i, o_i is the NN output and net_i is the weighted sum of the inputs to neuron i. Now when the partial derivatives for each weight are at our disposal we can proceed with minimization of error function E by performing a *gradient descent*. However, gradient, as a vector in weight space, provides us with a direction of steepest increase in E, therefore in order to decrease the magnitude of E we need

[23] Usually *squared error function*, defined as

$$E = \frac{1}{2}(\hat{y} - y)^2 \tag{2.5.1}$$

is used. This error function is also used in all NNs examined in this thesis.

to take the negative value and thus weight update is defined as

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \delta \frac{\partial E}{\partial w_{ij}} \tag{2.5.3}$$

where $\delta \in \Re$ is the *learning rate*. Training speed increases with the learning rate, however if δ is too big, the minimum might not be found at all, i.e. oscillation around minimum happens. The lower the learning rate, the more accurate the training is, but if δ is too small convergence to optimal weights might take long. Most commonly $0 < \delta < 0.5$.

Forward and backward propagation are repeated until the achieved errors on training and cross-validation samples are satisfactory[24]. The Figure 2.5.1 illustrates the desired behaviour of gradient descent algorithm.

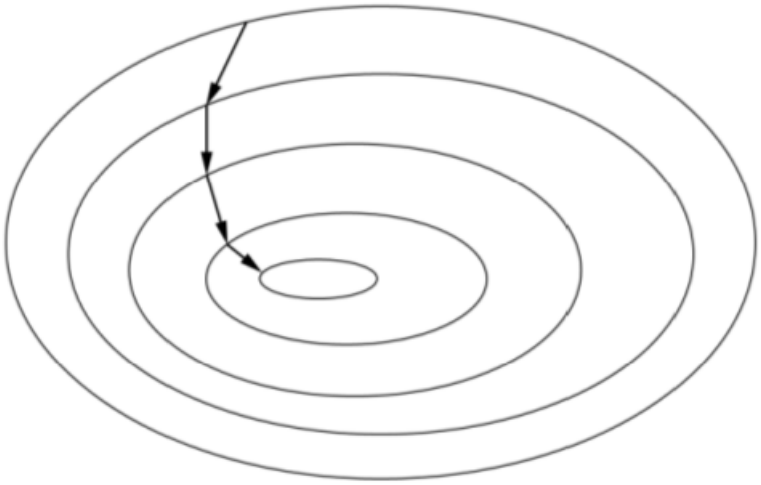

Figure 2.5.1: Steps of gradient descent toward local minimum

[24]See discussion on *bias-variance dillema* in section 2.4.

In order to try to soften the problems with setting the right learning rate and achieving optimal training process, *gradient descent with a momentum term* might be used for weight update. It's defined as

$$\triangle w_{ij}^{(t)} = -\delta \frac{\partial E}{\partial w_{ij}} + \mu \triangle w_{ij}^{(t-1)} \tag{2.5.4}$$

where $\triangle w_{ij}^{(t-1)}$ is the weight change from previous iteration and the momentum parameter $\mu \in (0, 1)$ changes influence of the previous weight change[25]. However, Riedmiller and Braun (1992) note that choosing the optimal value of the momentum parameter μ is as hard as in case with learning rate δ, i.e. no general improvement is accomplished as both parameters are highly problem dependent.

Another complication is that convergence in backpropagation is not guaranteed and the results may converge to a local minimum. This problem cannot occur when there is only one minimum, however usually error surface is substantially more complex with number of local minima.

2.5.2 Resilient propagation or RPROP

Resilient propagation is an efficient learning algorithm proposed by Riedmiller and Braun (1992). For each weight in NN its in-

[25] If $\mu = 0$ method is equal to standard gradient descent and if $\mu = 1$, the previous weight change has significant effect on current weight change.

dividual *update-value* $\triangle_{ij}$ adapts during the process of learning according to following algorithm

$$\triangle_{ij}^{(t)} = \begin{cases} \eta^+ * \triangle_{ij}^{(t-1)} & , \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \eta^- * \triangle_{ij}^{(t-1)} & , \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \triangle_{ij}^{(t-1)} & , else \end{cases} \tag{2.5.5}$$

where $0 < \eta- < 1 < \eta+$. When the last update-value $\triangle_{ij}^{(t-1)}$ was too big in a sense that the algorithm has jumped over a local minimum we can detect it because the partial derivative of w_{ij} must change its sign. And since we know that the last change was too big we need to move back, thus we decrease the update-value $\triangle_{ij}$ by the factor η^-. On the other hand when the last update-value $\triangle_{ij}^{(t-1)}$ was too smal and algorithm didn't cross a local minimum, i.e. the sign of corresponding partial derivatives remained the same, we need to increase $\triangle_{ij}$ by the factor η^+.

When all the update-values are computed we can finally move to individual weight-updates following simple rule

$$\triangle w_{ij}^{(t)} = \begin{cases} -\triangle_{ij}^{(t)} & , \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\triangle_{ij}^{(t)} & , \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0 & else \end{cases} \tag{2.5.6}$$

with one exception

$$\triangle w_{ij}^{(t)} = - \triangle w_{ij}^{(t-1)} \quad , if \ \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \tag{2.5.7}$$

if the previous step was too large, the previous weight-update needs to be undone by setting current weight update to negative value of the previous one. And finally new weights are computed as

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \triangle w_{ij}^{(t)} \tag{2.5.8}$$

The update-values and the weights are changed every time a training example is presented to the NN, i.e. *learning by epoch.*

2.5.3 Broyden-Fletcher-Goldfarb-Shanno algorithm

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, named after its creators who proposed it independently, method is the most popular so called *quasi-Newton method* and requires only the gradient[26] of the error function to be computed. Both Newton and Quasi-Newton methods are much faster then than steepest descent and this convergence improvement is achieved by utilizing second-order information about the error function. Without going into too much details error function can be approximated by the second-order Taylor series

$$E(\mathbf{w} + \triangle\mathbf{w}) = E(\mathbf{w}) + \mathbf{G}^T \triangle \mathbf{w} + \frac{1}{2}(\triangle\mathbf{w})^T \mathbf{H} \triangle \mathbf{w} \tag{2.5.9}$$

where

$$\mathbf{G} = \frac{\partial E}{\partial \mathbf{w}}$$

is gradient vector,

$$\mathbf{H} = \frac{\partial E^2}{\partial \mathbf{w}^2}$$

[26] Just like the steepest descent.

is Hessian matrix and $\mathbf{w}$ is the weight vector.

According to Newton method, which we do not use in this thesis, optimal update of weights is

$$\triangle\tilde{\mathbf{w}} = \mathbf{G}\mathbf{H}^{-1} \tag{2.5.10}$$

where $\mathbf{H}^{-1}$ is the inverse of the Hessian matrix. However, when evaluation of Hessian matrix is either slow or expensive (or both), quasi-Newton methods can be more efficient because the Hessian matrix is not required directly.

Since authors knowledge of numerical optimization is significantly limited we present only brief description of BFGS process and then the final formula for the weight-change. Difference with the Newton method is that instead of computing a new Hessian Matrix $\mathbf{H}_k$ in each iteration we only update a positive definite approximation matrix $\mathbf{B}_k$ such that

$$\lim_{x\to\infty} \mathbf{B}_k = \mathbf{H}^{-1} \tag{2.5.11}$$

where we take advantage of information about the curvature at k-th step. $\mathbf{B}_0 = I$ can be used as the first approximate Hessian matrix and in that case the first step will be the same as in case of a gradient descent. The final formula for the weight-change is then

$$\triangle\mathbf{w} = \mathbf{G}\mathbf{B}_k \tag{2.5.12}$$

For further details we suggest Broyden (1970), Fletcher (1970), Goldfarb (1970), Shanno (1970) and Nocedal and Wright (2006).

All illustrative pictures used in this chapter were taken either from Rojas (1996) or from open source wiki-book on neural networks[27].

With both[28] learning algorithms used in empirical part of this thesis presented, we have everything necessary for complex procedure of estimation with NNs. Also now, familiar with terminology used in NNs, we review how are they applied in forecasting of stock market returns, section (2.6), and volatility, section (2.7).

2.6 Neural Networks and forecasting of stock market returns

In order to make sense of a numerous attempts and researches directed at forecasting stock or stock market returns using NNs we can look at Atsalakis & Valavanis (2008), who surveyed over 100 related scientific articles.

Papers surveyed by Atsalakis & Valavanis (2008) focus on forecasting returns of a single stock market index or collection of them[29]. Stock market indexes from well developed markets, e.g.

[27] http://en.wikibooks.org/wiki/ArtificialNeuralNetworks/PrintVersion

[28] RPROP and BFGS algorithm

[29] Few of them focus also on forecasting of returns of individual stocks.

Western Europe, North America and other developed countries, as well as indexes from emerging markets are used in surveyed articles. The wide range of examined stock market indexes is appealing since there is no guarantee that possible predictability discovered at one stock market should be present in another as well.

The average number of input variables used in surveyed models is between four and ten, maximum of 61 inputs is used by Zorin & Borisov (2002) for the Latvian Riga stock exchange index and minimum of 2 inputs in case of Constantinou et al. (2009). The most frequently used input variables are historical index values with different number of lags. Volume, Open, Low, High, Close, technical indicators, fundamental indicators as well as economic indicators are all used in different articles[30].

Time horizons and sample sizes differ significantly ranging from just 40 observations to 24 years of daily returns used by Thawornwong & Enke (2004).

With respect to data preprocessing there are 13 studies where input data are scaled either to (0,1) or (-1,1), additional prepro-

[30]For specific percentages of articles using particular techniques, please see Atsalakis & Valavanis (2008).

cessing techniques are Z-score normalization[31], Principal Component Analysis (PCA)[32] or logarithmic data preprocessing. Almost all surveyed articles, even if they don't use it, find data preprocessing useful.

The average number of hidden layers in used NNs is one or two and the maximum number of input neurons is used by already mentioned Zorin & Borisov (2002).

The main conclusion of the Atsalakis & Valavanis (2008) survey is that NNs outperform conventional models in most cases, i.e. they offer better results as trading systems and higher accuracy.

Furthermore, according to Kanas & Yannopoulos (2001) trading volume is considered to be fundamental factor in long-term forecasting, however Phua et al. (2003) in case of stock index direction forecasting, find little to no effect of including volume as input variable. Also in direction forecasting, Zhu et al. (2007) show modest improvement using trading volume, nevertheless only in case of medium to long-term forecasting.

Because of these disputes about usefulness of trading volume

[31] See section (3.2.1).

[32] See section (3.2.2).

as explanatory variable, in section (4.2.1) we use 20 lagged values of trading volume as inputs for examined NNs under different time horizons.

2.7 Neural Networks and forecasting of stock market volatility

As presented in section (2.6) neural networks have strong reputation in financial forecasting, therefore it is no surprise that they found their way into task of volatility prediction as well.

NNs were first used in the forecasting of DAX German index volatility by Ormoneit & Neuneier (1996) providing evidence that forecasting capabilities of this method can be applied also in case of volatility prediction. Donaldson & Kamstra (1997) use NNs and demonstrate through comparison with more standard models such as GARCH, EGARCH, etc., that neural networks are able to capture patterns in volatility that compared models cannot. Miranda & Burgess (1997) examining predictability of implied volatility on Spanish stock market reject the hypothesis that volatility changes are unpredictable on an hourly basis. Schittenkopf et al. (1998) using Austrian stock market data found that NNs outperform both ARCH and GARCH models with respect to higher correlation with implied volatility. Hamid & Iqbal (2004) use NNs for forecasting volatility of S&P500 Index future prices and find that forecasts from NNs outperform

implied volatility forecasts. Monfared & Enke (2014) study abilities of NNs in volatility forecasting during four different economic cycles from 1997-2011 and they conclude that NNs perform especially well compared to other econometric models during extreme turmoil conditions, i.e. 2008 financial crash. However, in low volatility periods, they do not recommend to use NNs for forecasting purposes because of the unnecessary complexity of the model. It is also shown that there is no best architecture for forecasting volatility, since each crisis has its own characteristics. Kristjanpoller et al. (2014) also show superior performance of NNs in forecasting of three Latin-American stock exchange indexes and that results are robust and consistent with respect to different specifications of NN and different volatility measures.

Furthermore alongside with standard NNs often different hybrid models of NNs are used achieving mixed results, e.g. Roh (2007), Tseng et al. (2008) and Hajizadeh et al. (2012).

This thesis contributes to the existing body of evidence about predictability of stock index volatility in different time horizons. Corresponding empirical findings can be seen in section 4.2.2.

3 Statistical tests, data preprocessing & models evaluation criteria

In section (3.1) we briefly present two statistical test that are later, in section (4.1), used to obtain some insights about characteristics of analysed data.

Section (3.2) explains concept of data preprocessing and its importance in relation with NNs.

This chapter is concluded with presentation of criteria that are used in performance evaluation and help us with the interpretation of achieved results.

3.1 Statistical tests

Traditional financial time series models require strict assumptions about distributions of relevant time series. Comparative advantage of NNs is that they do not depend on these types of assumptions, because of their superior ability to adapt, McNelis (2005). Nevertheless, in order to verify presence of expected stylized facts in our dataset we present in section (3.1.1) Jarque-Bera normality test and in section (3.1.2) augmented Dickey-Fuller test for unit root.

3.1.1 Normality test

Jarque and Bera (1980) presented now common goodness-of-fit test of normality based on comparing empirical skewness and kur-

tosis with skewness and kurtosis of normal distribution. The null hypothesis is that data are distributed according to the normal distribution with zero skewness and kurtosis of 3. Formula for the Jarque-Bera statistics is presented in equation and under the null hypothesis follows $\chi^2(2)$:

$$JB(x) = \frac{N}{6}(S^2 + 1/4(K-3)^2), \tag{3.1.1}$$

N is the number of observations, S is the sample skewness and K is the sample kurtosis. In case of using JB test in multiple regression analysis, formula for JB statistic needs to be slightly modified:

$$JB(x) = \frac{N-k}{6}(S^2 + 1/4(K-3)^2), \tag{3.1.2}$$

where k is the number of regressors.

3.1.2 Unit root test

Stationary random variables cannot contain an unit root. The presence of the unit root can be tested in numerous ways and one of the most popular methods was proposed by Dickey and Fuller (1979). The null hypothesis is that a stochastic process contains an unit root against the alternative hypothesis that process is stationary. Formula for test statistic is following:

$$DF = \frac{\sum_{t=1}^{T} x_{t-1}\epsilon_t}{\hat{\sigma}_\epsilon \sum_{t=1}^{T} x_{t-1}^2} \tag{3.1.3}$$

Under the null hypothesis of non-stationarity, DF statistic follows non-standard distribution, therefore it's not possible to use standard t-test. Critical values were obtained by Dickey and Fuller

using Monte Carlo simulations. However, when the errors are autocorrelated the normal test significance is not precise and the distortion grows with the order of autocorrelation.

Augmented Dickey-Fuller (ADF) test is, as name has it, augmented version of Dickey-Fuller test described above. It includes an unknown amount of the lagged first differences as dependent variables, this way autocorrelated omitted variables are accounted for and the negative property of previous test is eliminated.

3.2 Data preprocessing

Data preprocessing consists of transforming the input and output variables in order to assist a neural network in learning the relevant patterns[33]. In this thesis input variables are scaled to range $(-1, 1)$, then principal component analysis is performed in order to obtain the principal components which serve as inputs for all examined NNs. Both data scaling and principal component analysis are discussed below in sections (3.2.1) and (3.2.2).

3.2.1 Data scaling

When including multiple independent or dependent variables into models based on machine learning algorithms it is often required to standardize corresponding variables. For NNs it is rarely neces-

[33]In broader sense data preprocessing of course also includes detecting trends, minimizing noise, etc.

sary because any rescaling of an input variables can be undone by adapting NN's weights leading to the same outputs as we would have without scaling the input vector. However, as Sarle (1999) reports even though both RPROP and quasi-Newton training algorithms such as BFGS are capable to adapt to ill-conditioning rather quickly, proper conditioning can lead to performance improvement. Here we present two most common data scaling methods from which one is used in this thesis.

Z-score normalization is a well known statistical procedure in which the mean (μ) and standard deviation (σ) is calculated for each (usually) input variable as shown in formula (3.2.1)

$$z_i = \frac{x_i - \mu_i}{\sigma_i} \tag{3.2.1}$$

Normalized data have zero mean and unit variance. One of the effects of Z-score normalization is that it reduces the weight of outliers in the data.

Min-Max normalization is linear transformation of data from one range of values to a new range of values. Two most often used ranges for input variables are $(0, 1)$ and $(-1, 1)$, however the first one is very common misconception from earlier days of machine learning [34], it is better to have data centered near zero

[34] As mentioned earlier when using more advanced training algorithms the negative effect is rather small.

and in interval $(-1, 1)$. It can be achieved by using formula 2:

$$x_i' = 2 * \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} - 1, \tag{3.2.2}$$

where of course $max(x_i) - min(x_i) \neq 0$, since in that case we would have a constant feature that should be removed because it doesn't provide any information.

It should also be noted that any statistics used in data scaling, e.g. min, max, mean, etc., need to be computed from the the training dataset and both cross-validation and test dataset need to be scaled with statistics obtained from training dataset.

3.2.2 Principal component analysis

The core of *principal component analysis* (PCA) is reducing the dimensionality of input vector with respect to keeping as much information contained in the data as possible. It's based on the idea that the most relevant information in a given set of variables lies in the linear combinations of data that have the largest variance. For n given vectors, n linearly independent combinations explain 100% variation of the original data, but if we want to reduce the dimensionality of the original data we need to use less of them. Principal components are nothing more than those linear combinations, they are the underlying structure in the data. PCA is also referred to as Karhunen-Loeve transform, singular value decomposition, Hotelling transform and many others. Next we present the basic mathematical background of PCA.

Let's have d-dimensional vectors x_i that we want to map to m-dimensional vectors z_i, where $m < d$, then a set of d orthonormal vectors u_i, i.e. satisfying

$$u_i^T u_j = \delta_{ij} \tag{3.2.3}$$

where δ_{ij} is Kronecker delta[35]. Then $x = \sum_{i=1}^{d} z_i u_i$ and z_i can be expressed as

$$z_i = u_i^T x \tag{3.2.5}$$

In order to approximate x using orthonormal vectors u_i and coefficients we can do the following

$$x = \sum_{i=1}^{m} z_i u_i + \sum_{i=m+1}^{d} c + iu + i \tag{3.2.6}$$

where c_i is constant. Sum of squared errors $\Phi(u)$ of dataset of N samples needs to be minimized.

$$\Phi(u) = \frac{1}{2} \sum_{n=1}^{N} \|x_n = \tilde{x_n}\|^2 = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=m+1}^{d} (z_{n,j} - b_i)^2 \tag{3.2.7}$$

For more technical details we direct reader to Jolliffe (2002) who also shows that minimum can be found when following condition is satisfied:

$$\sum_{i=1}^{n} (x_n - \bar{x})(x_n - \bar{x})^T u_i = \lambda_i u_i \tag{3.2.8}$$

[35]Defined as function of two variables:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{3.2.4}$$

Since

$$\sum_{i=1}^{n}(x_n - \bar{x})(x_n - \bar{x})^T \tag{3.2.9}$$

is covariance matrix, λ_i are eigenvectors of covariance matrix. Value of error in the minimum is then

$$\Phi(u_{min}) = \frac{1}{N}\sum_{i=m+1}^{d}\lambda_i \tag{3.2.10}$$

and by eliminating $d - m$ eigenvectors with the smallest eigenvalues we end up with m desired *principal components*.

3.3 Models evaluation

3.3.1 Mean squared error

One of the most common statistics used for evaluating out-of-sample performance was already suggested in (2.5.1) and used as an example of a cost function in learning of the NN. Here we offer the complete definition of *mean squared error*

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{3.3.1}$$

where T is the number of observations, $\hat{y}_t$ is the prediction and y_t is the true value.

3.3.2 Root mean squared error

Another often used statistics is root mean squared error statistic (RMSE) defined simply as square root of MSE

$$RMSE = \frac{1}{T}\sqrt{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{3.3.2}$$

3.3.3 Mean absolute error

Mean absolute error (MAE) is defined as

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t| \tag{3.3.3}$$

This chapter is concluded with the presentation of Diebold-Mariano test in the following section.

3.3.4 Diebold-Mariano test

As mentioned earlier one of the most important element of this thesis is our ability to compare out-of-sample performance of one model with another, i.e. to choose model which out-of-sample fit is significantly better. Diebold-Mariano test, created by Diebold & Mariano (1995), offers an answer.

The null hypothesis is that of equal forecast accuracy and is tested against the alternative of non-equal forecast accuracy, i.e. different predictive ability of two *non-nested* models.

The forecast errors are defined as

$$e_{it} = \hat{y_{it}} - y_t; \qquad i = 1, 2 \tag{3.3.4}$$

where $\{\hat{y_{1t}}; t = 1, ..., T\}, \{\hat{y_{2t}}; t = 1, ..., T\}$ are forecast values of models we are comparing and $\{y_t; t = 1, ..., T\}$ are true values. In order to choose more accurate model, the expected losses associated with both forecasts need to be compared. For our purposes

we restrict ourselves on two loss functions based on the forecast errors:

- *squared-error* loss function

$$L_{sqr}(e_{it}) = e_{it}^2 \tag{3.3.5}$$

- *absolute error* loss function

$$L_{abs}(e_{it}) = |e_{it}| \tag{3.3.6}$$

The loss differential between the two forecasts is defined as

$$d_t = L_(e_{1t}) - L_(e_{2t}) \tag{3.3.7}$$

thus $H_0 : E[d_t] = 0$ and $H_1 : E[d_t] \neq 0$. The corresponding test statistics is

$$DM_\tau = \frac{\frac{1}{T}\sum_{t=1}^{T} dt}{\sqrt{\frac{1}{T}\sum_{\tau=-(T-1)}^{T-1} 1\left(\frac{\tau}{S(T)}\right)\hat{\gamma}(\tau)}} \sim N(0,1) \tag{3.3.8}$$

where

$$\hat{\gamma}(\tau) = \frac{1}{T}\sum_{t=|\tau|+1}^{T} (d_t - \hat{d}_t)(d_{t-|\tau|} - \overline{d}) \tag{3.3.9}$$

$S(T)$ is the truncation lag and $1\left(\frac{\tau}{S(T)}\right)$ is the lag window.
The DM_τ statistics is approximately normally distributed under the null hypothesis of no difference in models accuracies. Therefore at the 5% significance level in order to, let's say forecast errors of a first model to be significantly lower than errors of a second model, the following should hold: $|DM_\tau| > 1.96$. For

deeper understanding of DM test please see Diebold & Mariano (1995).

In the following chapter, when comparing performances of examined models we report DM_τ statistics for both squared-error loss function as well as absolute error loss function.

4 Empirical results

In this chapter we use previously described methods to try to predict the future returns and volatility of NASDAQ Composite stock market index. NASDAQ Composite index, listed on Nasdaq stock market, is a market-capitalization weighted index including all listed stocks on the Nasdaq stock exchange and is considered to be one of the most important indicator of US stock markets performance. Predictability of returns and volatility is analysed in four different time horizons: 1 day, 1 week, 2 weeks and 1 month ahead (i.e. 1, 5, 10 and 20 trading days). According to articles reviewed in sections (2.6) and (2.7), we can expect reasonably good performance of examined NNs in case of both prediction tasks, i.e. stock index returns and volatility.

In section (4.1) we describe examined data for both returns and daily range-based volatility. Data are partitioned into 3 complementary subsets:

Training sample contains first 60% of observations and are used in model fitting, i.e. learning of our NN's.

Cross-validation sample contains following 20% of observations. It enables us to asses how will our models generalize to an independent data set and avoid problem of overfitting.

Testing sample contains the last 20% of observations and is used to compare performances of individual models.

It should be noted here that we only use 2-fold non-exhaustive cross-validation, i.e. we cross-validate only on one selected subset not on all possible subsets.

In sections (4.2.1) and (4.2.2) we obtain empirical results from both returns and volatility forecasting and using MSE, RMSE, MAE and Diebold-Mariano test we compare them. In the last section (4.3) achieved results are discussed alongside possibilities for further improvements of performed analysis.

4.1 Data description

All the data were obtained from Yahoo Finance, they comprise of daily Open, High, Low, Close, Volume , Adjusted Close and range from January 1990 to March 2015 offering us 6339 observations.

4.1.1 Returns

We analyze returns defined in 1.1.1 as

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

where P_t is Adjusted Close from original dataset. As mentioned, in our case there are four different time periods and corresponding returns are defined as

$$R_t^h = \frac{P_t - P_{t-h}}{P_{t-h}}; \quad h = \{1, 5, 10, 20\} \tag{4.1.1}$$

Time series of returns in all different time horizons are illustrated in Figure 4.1.1.

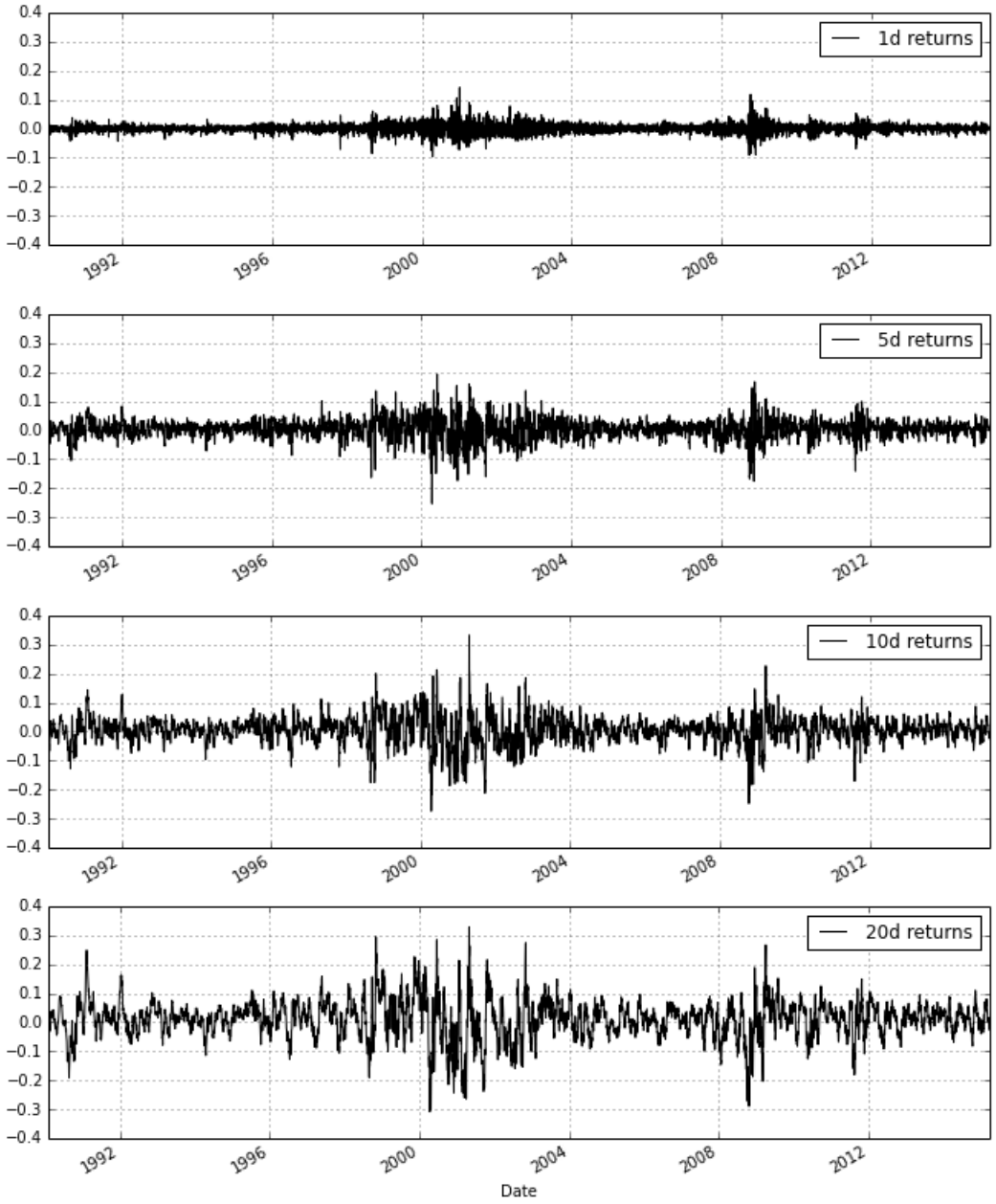

Figure 4.1.1: Returns in different time horizons

As we can see, dot-com bubble at the end of the last and beginning of the new millennium and the global financial crisis of 2007-2008 brought significant amount of volatility. These two

extremely volatile periods in the last 25 years create challenging complication for any possible forecasting method. Summary statistics for returns in all time horizons as well as Jarque-Bera test statistics are presented in Tables 4.1 and 4.2.

Table 4.1: Summary statistics for returns: Part 1

	Mean	**Median**	**Minimum**	**Maximum**
1d returns	0.000487	0.001144	-0.096685	0.141732
5d returns	0.002410	0.004421	-0.253047	0.192396
10d returns	0.004786	0.007812	-0.273690	0.331547
20d returns	0.009771	0.014693	-0.307795	0.327316

Table 4.2: Summary statistics for returns: Part 2

	Std. Dev.	**Skewness**	**Kurtosis**	**JB**
1d returns	0.014930	0.098672	6.541341	11290.8791*
5d returns	0.032330	-0.436229	3.797371	4001.7399*
10d returns	0.044672	-0.377281	3.319753	3054.9237*
20d returns	0.064642	-0.367248	2.217055	1437.5325*

*Significant at the 1% level.

Mean values and standard deviations of returns are growing with widening of the time frame, which is nothing unexpected. JB test results allow us to reject the null hypothesis under which returns follow normal distribution in all time horizons. Distributions are strongly *leptokurtic*. Although none of this is new it should again be stressed that markets are more volatile with more tail movements than normal distribution would imply. This characteristic only underpins the importance of volatility predic-

tion. In order to obtain another insight into data characteristics we perform Augmented Dickey-Fuller test on all time series and reject the null hypothesis of unit roots presence, results are reported in Table **??**.

	1d returns	**5d returns**	**10d returns**	**20d returns**
ADF	-13.363834*	-13.054419*	-11.883663*	-11.182196*

*Significant at the 1% level.

4.1.2 Volatility

As mentioned in section 1.2.1 we define our range-based volatility estimator in day t according to Parkinson (1980) as in(1.2.1)

$$v_t = 0.361 * (ln(High) - ln(Low))^2$$

Corresponding historical daily range-based volatility is plotted below in Figure 4.1.2.

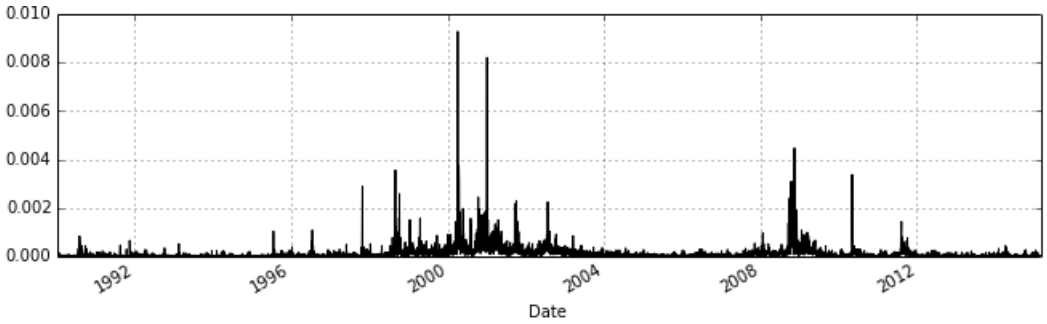

Figure 4.1.2: Daily range-based volatility

Since daily volatility is a measure of price variation and returns are simply measure of change, we can observe clear relationship between their charts and once again see the turmoil

brought by two crisis into otherwise rather peaceful behaviour of NASDAQ Composite index. Summary statistics for time series of daily volatility are presented below in Tables 4.3 and 4.4.

Table 4.3: Summary statistics for daily volatility: Part 1

	Mean	**Median**	**Minimum**	**Maximum**
1d volatility	0.000131	5.03299e-05	0.0	0.009273

Table 4.4: Summary statistics for daily volatility: Part 2

	Std. Dev.	**Variance**	**Kurtosis**	**Skewness**
1d volatility	0.000301	9.05e-08	11.55064	248.050897

We perform ADF test and get statistic of -6.288196 which allows us to reject the null hypothesis of unit roots presence on 1% significance level.

Finally after knowing basic characteristics of data that we are working with we can move to the most essential parts of this thesis, i.e. forecasting returns and volatility in different time horizons.

4.2 Empirical results

In order to test if our NNs hold any predictive power we compare them with simple AR(1) process on both returns and volatility data. In both cases we are interested if there are some non-linear patterns present in the data as well as time window in which they

might exist.

For both returns and volatility we examine 4 neural networks, 2 training algorithms and 2 network architectures.
Since estimation with NNs is still considered to be a "state of the art", architectures, in our case basically only number of hidden layers and hidden neurons, are chosen according to empirically-derived rule of thumb: "The number of hidden neurons is the mean of the neurons in the input and output layers."

In case of training algorithms, previously described Broyden-Fletcher-Goldfarb-Shanno and resilient backpropagation were chosen. We picked BFGS as personal preference of the author because he never worked with it before, rprop because it is considered to be one of the fastest weight update algorithms and omitted others either because of technical difficulties or because they've already been used previously by members of IES, e.g. Baruník (2006), Baruník & Vácha (2008), Baruníková & Baruník (2011), Verner (2011).
All examined networks are fully connected.

Our analysis is performed using Python programming language with exception of DM test which up to this date did not find the way into the Python libraries of scientific computing.

Thus for DM test we use R programming languge. For interested reader, the source code used in our research can be directly provided if requested.

4.2.1 Returns

In case of examining predictability of returns in different time horizons we choose as inputs 10 principal components from PCA performed on 20 lags of scaled returns and 20 lagged scaled daily volumes. Historical values of volume were chosen according to Kanas & Yannopoulos (2001) and Zhu et al. (2007) who find them to be important factor in the prediction task of stock index returns. Examined NNs are therefore:

- 11-6-1 (BFGS)
- 11-3-3-1 (BFGS)
- 11-6-1 (rprop)
- 11-3-3-1 (rprop)

Where we add 1 neuron corresponding to the bias term. In input and hidden neurons previously mentioned hyperbolic tangent sigmoid transfer function is used and for output neuron we use simple linear function.

In Tables 4.5, 4.6 and 4.7 we compare the results achieved by all the examined models. We do so in 4 time horizons that are in our interest and with respect to meansquared error, root mean-square error and mean absolute error.

Method	1d returns	5d returns	10d returns	20d returns
AR(1)	0.000125	0.000575	0.000993	0.001906
NN1-BFGS	0.000131	0.000593	0.001046	0.002028
NN2-BFGS	0.000126	0.000589	0.001161	0.002105
NN1-RBP	0.000145	0.000833	0.001262	0.002567
NN2-RBP	0.000144	0.000719	0.001294	0.001938

Table 4.5: Mean squared errors for returns

According to the results in Table 4.5 we can see that the lowest MSE among the compared models for all time horizons is obtained by AR(1) model. These results are very surprising, since according to Atsalakis & Valavanis (2008) NNs should very probably outperform classical econometric models, especially such simple one as AR(1).

Method	1d returns	5d returns	10d returns	20d returns
AR(1)	0.011194	0.023971	0.031504	0.043656
NN1-BFGS	0.011464	0.024360	0.032347	0.045033
NN2-BFGS	0.011219	0.024269	0.034080	0.045881
NN1-RBP	0.012036	0.028865	0.035528	0.050661
NN2-RBP	0.012019	0.026818	0.035972	0.044027

Table 4.6: Root-mean-squared errors for returns

Method	1d returns	5d returns	10d returns	20d returns
AR(1)	0.007998	0.018041	0.024444	0.034490
NN1-BFGS	0.008110	0.018383	0.024972	0.034943
NN2-RBP	0.007993	0.018295	0.025556	0.035372
NN1-BFGS	0.008496	0.020292	0.027424	0.039037
NN2-RBP	0.008371	0.019943	0.025995	0.034253

Table 4.7: Mean absolute errors for returns

Interpretation of results in Table 4.7 exactly the same as in case with MSE. The lowest MAE at 1, 5 and 10 days was against achieved by AR(1), however in case of 20 days forecast the lowest MAE was achieved by neural network with 2 hidden layers and resilient back-propagation as learning algorithm.

In Figure 4.2.1 we plot the predicted values of the only examined model with lower MAE than that of AR(1) model; NN2-RBP.

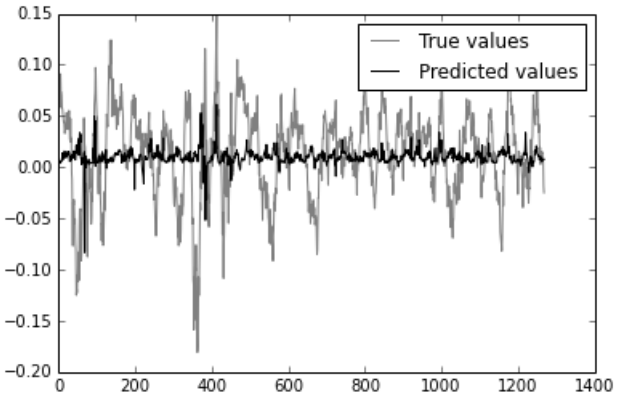

Figure 4.2.1: True vs. predicted 20d returns (NN2-RBP)

In order to compare accuracy of forecasts, we present the values of Diebold-Mariano test statistics with both squared-error and absolute error loss functions in Tables (4.8) and (4.10) and corresponding p-values in Tables (4.9) and (4.11). AR(1) model is considered as benchmark model against which all other models are competing. As alternative hypotheses we choose better performance of compared NN, i.e. significantly lower errors of NN.

Method	**1d returns**	**5d returns**	**10d returns**	**20d returns**
NN1-BFGS	-1.111136	-1.845942	-1.511426	-4.919630
NN2-BFGS	0.055630	-1.560099	-1.818941	-1.401076
NN1-RBP	-4.707618	-3.297899	-4.358871	-4.674702
NN2-RBP	-3.095161	-4.766005	-2.069901	0.438610

Table 4.8: DM test statistics for returns (absolute error loss function)

Method	**1d returns**	**5d returns**	**10d returns**	**20d returns**
NN1-BFGS	0.866640	0.967433	0.934535	0.789772
NN2-BFGS	0.477823	0.940507	0.965422	0.909017
NN1-RBP	0.999999	0.999499	0.999993	0.999998
NN2-RBP	0.998995	0.999999	0.980668	0.330509

Table 4.9: DM p-values for returns (absolute error loss function)

DM test with absolute error loss function shows that we cannot reject the null hypothesis of equal forecasts even on 10% significance level for any NN and on any time horizon.

Method	1d returns	5d returns	10d returns	20d returns
NN1-BFGS	-0.000166	-0.018050	-0.003203	-0.026499
NN2-BFGS	-0.000009	-0.016226	-0.037780	-0.035071
NN1-RBP	-0.001065	-0.071115	-0.025229	-0.126390
NN2-RBP	-0.000990	-0.029612	-0.174061	-0.016334

Table 4.10: DM test statistics for returns(squared error loss function)

Method	1d returns	5d returns	10d returns	20d returns
NN1-BFGS	0.500066	0.507199	0.501277	-0.026499
NN2-BFGS	0.500003	-0.016226	0.515066	0.513986
NN1-RBP	0.500425	0.528341	0.510062	0.550278
NN2-RBP	0.500395	0.511809	0.569077	0.506515

Table 4.11: DM p-values for returns (squared error loss function)

DM test with squared error loss function confirms our observation, i.e. again we cannot reject the null hypothesis even on 10% significance level.

Achieved results that show failure of examined NNs to compete even with simple AR(1) model, indicate lack of predictability of Nasdaq Composite returns in all four time horizons. Furthermore, trading volume doesn't seem to be an important factor in forecasting of Nasdaq Composite returns as suggested by Zhou et al. (2007).

4.2.2 Volatility

For analysis of volatility predictability in different time horizons we choose 20 lagged values of range-based volatility as inputs. Performances of following NNs are examined:

- 21-11-1 (BFGS)
- 21-6-5-1 (BFGS)
- 21-11-1 (RBP)
- 21-6-5-1 (RBP)

where we again add 1 bias neuron, however this time in output layer log-sigmoid activation function is chosen, since volatility cannot be negative. In Tables (4.12), (4.13) and (4.14) we compare achieved results according to MSE, RMSE, MAE.

Method	**1d ahead**	**5d ahead**	**10d ahead**	**20d ahead**
AR(1)	1.664840e-08	2.016840e-08	2.012067e-08	2.085648e-08
NN1-BFGS	1.499944e-08	1.761323e-08	1.793961e-08	1.865430e-08
NN2-BFGS	1.538466e-08	1.721411e-08	1.767091e-08	1.871359e-08
NN1-RBP	1.529729e-08	1.754336e-08	1.773455e-08	1.868167e-08
NN2-RBP	1.602894e-08	1.757511e-08	1.812715e-08	1.995388e-08

Table 4.12: Mean squared error for volatility

Method	1d ahead	5d ahead	10d ahead	20d ahead
AR(1)	0.000129	0.000142	0.000142	0.000144
NN1-BFGS	0.000122	0.000133	0.000134	0.000137
NN2-BFGS	0.000124	0.000131	0.000133	0.000137
NN1-RBP	0.000124	0.000132	0.000133	0.000137
NN2-RBP	0.000127	7 0.000133	0.000135	0.000141

Table 4.13: Root mean squared error for volatility

According to MSE the performances of all NNs on all the time horizons are superior to simple AR(1) process. First of all it is important to note the difference in input variables, i.e. for NNs we take 20 lagged values, but in case of AR(1) only one. This might be a strong source of advantage for all NNs[36]. Another possible reason for good performances of NNs are potential non-linear patterns hidden in the data. Forecasting 1 day ahead, the best result was achieved by NN1-BFGS, i.e. NN with 1 hidden layer, 6 hidden neurons and BFGS learning algorithm. Predicting 1 week ahead, NN2-BFGS fared the best. In 10 day time frame again NN2-BFGS had the best performance. On 1 month time horizon the lowest MSE was achieved by NN1-BFGS.

[36]Since we are concerned with question of predictability and are not necessarily interested in comparison of classical econometric models with machine learning models we do not include other classical models such as ARIMA, GARCH,etc. for comparison.

Method	1d ahead	5d ahead	10d ahead	20d ahead
AR(1)	0.000048	0.000054	0.000053	0.000056
NN1-BFGS	0.000052	0.000059	0.000063	0.000069
NN2-BFGS	0.000050	0.000058	0.000063	0.000071
NN1-RBP	0.000050	0.000059	0.000060	0.000068
NN2-RBP	0.000054	0.000059	0.000062	0.000066

Table 4.14: Mean absolute error for volatility

Results from comparison of models with respect to MAE is unexpected. No NN was able to outperform the AR(1) process. What's more, this inability is observable in all time horizons.

MSE as a quadratic loss function gives considerably more weight to large errors than to small one[37]. On the other hand, MAE is a linear error measure, not quadratic, therefore big errors have smaller influence than in the case of quadratic error measures. If necessary and depending on whether we consider large errors to be increasingly more harmful we could choose one or the other to be more relevant.

As in the case of returns, in order to compare accuracy of forecasts we present the values of Diebold-Mariano test statistics with both squared-error and absolute error loss functions in Tables (4.17) and (4.15) and corresponding p-values in tables (4.18) and (4.16). AR(1) model is again considered as benchmark model

[37]e.g. square error of 10 is 100 but square error of 5 and 5 is only 25 + 25, i.e. half

against which all other models are competing. As alternative hypotheses we choose better performance of compared NN.

Method	**1d ahead**	**5d ahead**	**10d ahead**	**20d ahead**
NN1-BFGS	-2.744249	-2.387130	-3.476500	-3.098224
NN2-BFGS	-1.225082	-1.792200	-3.224676	0.000071
NN1-RBP	-1.463990	-2.541751	-2.435477	-3.071865
NN2-RBP	-4.802668	-2.529713	-3.311132	-2.747985

Table 4.15: DM test statistics for volatility (absolute error loss function)

Method	**1d ahead**	**5d ahead**	**10d ahead**	**20d ahead**
NN1-BFGS	0.996925	0.991437	0.999737	0.999005
NN2-BFGS	0.889614	0.963330	0.999353	0.999692
NN1-RBP	0.928278	0.994426	0.992495	0.998914
NN2-RBP	0.999999	0.994232	0.999522	0.996959

Table 4.16: DM p-values for volatility (absolute error loss function)

The results of DM test with absolute error loss function are not surprising. Since MAE achieved by NNs was higher than MAE of AR(1) we cannot expect that DM would be close to rejecting null hypothesis of equal accuracy in favour of better accuracy of NNs.

Method	1d ahead	5d ahead	10d ahead	20d ahead
NN1-BFGS	1.295077	1.357627	1.972178	1.286340
NN2-BFGS	0.987012	1.686755	0.999353	1.277674
NN1-RBP	1.318519	1.327419	1.695625	1.297662
NN2-RBP	1.253251	1.291062	1.663498	1.388661

Table 4.17: DM test statistics for volatility(squared-error loss function)

Method	1d ahead	5d ahead	10d ahead	20d ahead
NN1-BFGS	0.097765	0.087412	0.042677	0.099280
NN2-BFGS	0.161913	0.082880	0.044855	0.100799
NN1-RBP	0.093784	0.092305	0.045101	0.097320
NN2-RBP	0.105173	0.098459	0.048230	0.082590

Table 4.18: DM p-values for volatility(squared error loss function)

Results of DM test with squared error loss function are more interesting and they differ according to chosen time horizon. No NN is significantly more accurate either at 1 day or 1 month time frame. 1 week ahead NN2-BFGS is more accurate than AR(1) process but only at 10% significance level. Time horizon of 2 weeks seems to be where most of the predictability lies. Both NN1-RBP and NN2-RBP are more accurate than AR(1), i.e. we can reject the null hypothesis of equal forecast accuracy at 10% significance level. The best performer is NN1-BFGS, its 2 weeks ahead forecasts are significantly more accurate than those of AR(1) model, i.e. we can reject the null hypothesis of equal forecast accuracy at 5% significance level.

As an illustration of out-of-sample predictions we show in Fig-

ure (4.2.2) true daily range-based volatility and values predicted by NN1-BFGS.

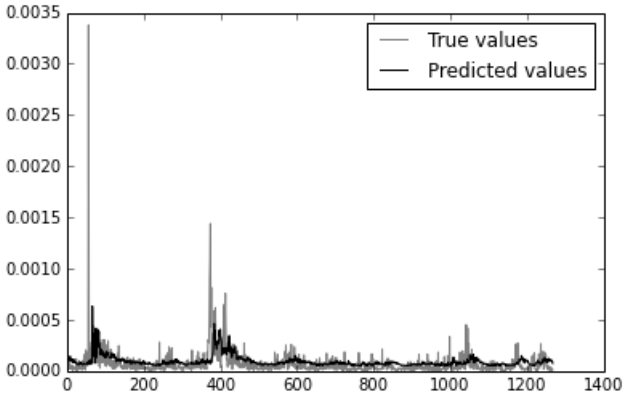

Figure 4.2.2: True vs. predicted volatility in 10 day (NN1-BFGS)

Results obtained in case of forecasting daily range-based volatility are more optimistic than results from forecasting stock market returns. Especially in case of forecasting two weeks ahead, NN with 1 hidden layer, 11 hidden neurons and BFGS learning algorithm is capable of recognizing predictive patterns in daily range-based volatility. Also in case of forecasting one week ahead, even though only on 10% significance level, NN with 2 hidden layers and BFGS learning algorithm seems to be able to capture patterns. In general, examined NNs were able to detect predictability in medium-term time window, i.e. 1-2 weeks ahead.

Following section offers short discussion of possible improvements of our methodology, then in the final chapter (4.3) we summarize achieved results.

4.3 Discussion

There is number of possible heuristics that could be used to enhance our methodology.

First of all, set of input variables considered in this thesis is very restricted. With huge number of possible financial indicators and many other random variables there are many possibilities of including superior inputs in terms of relevance. However, if we include all possible indicators as inputs, we will very soon suffer the *curse of dimensionality*[38], i.e. required amount of data for proper estimation of a model grows exponentially with the number of dimensions in the feature space. That being said, there certainly are variables that would benefit our analysis.

Second of all, as shown by Monfared & Enke (2014) in case of volatility prediction, each time period has its own characteristics suggesting that methods trained on the particular periods and applied in time periods with similar characteristics should perform better. As an simple example can serve models behaving differently with respect to the level of volatility, i.e. low risk vs. high risk periods.

Related to the second point, ability to identify individual peri-

[38]First noticed by Belman (1961) in dynamic optimization.

ods seems to be crucial. Unsupervised learning methods focused on clustering might offer a solution.

As the last point, the length of the time period used in this thesis, approximately 25 years, has both advantages and disadvantages. Advantage is obvious, large number of observations. Disadvantage is that stock market is different environment in a sense that relationships between individual variables of the system are different compared to 25 years ago. Therefore, shorter time period might be useful.

Conclusion

In this thesis we apply neural networks as nonlinear, nonparametric and robust method in stock market modelling. We are interested in stock market forecasting, concretely forecasting of Nasdaq Composite index returns and its daily range-based volatility. Predictability of returns and volatility is examined in four different time horizons, i.e. 1 day, 1 week, 2 weeks and 1 month ahead.

In the beginning we present the backgrounds of individual forecasting tasks, for returns it is martingale model, random walk hypothesis and efficient market hypothesis and in case of volatility it is range-based volatility estimation and classical volatility models.

Then in the second chapter we introduce neural networks with all the important elements such as neuron, activation function, architecture and learning algorithms. Learning and its possible pitfalls are briefly discussed, followed by Backpropagation, resilient propagation and Broyden-Fletcher-Goldfarb-Shanno learning algorithms explanations. The rest of the chapter focuses on reviewing applications of neural networks both in stock returns and volatility forecasting. It confirms that our expectation of high capability of neural networks are reasonable.

Subsequently in the third chapter we present statistical tests that are later used in describing statistical properties of examined time series, explain importance and techniques used in data preprocessing and introduce evaluation criteria used in comparing of models.

The last chapter starts with the data description, where we reject both the normality and unit root presence in Nasdaq Composite returns in all time horizons. In case of volatility presence of unit root is rejected as well. Input variables used in returns prediction are 10 principal components of between (-1,1) scaled 20 lagged returns and trading volumes. For volatility 20 lags are taken as input variables, without scaling.

In case of returns results don't match our expectations and we fail to find significant predictive patterns in any time horizon. Moreover, it seems that Nasdaq Composite returns time series are so noisy, that even simple AR(1) model performs better than more complex neural networks.

On the other hand results from forecasting of daily range-based volatility are positive. Even though, using Diebold-Mariano test, we find that 1 day and 20 day ahead forecasts by neural networks are not significantly better than AR(1), in case of 5 and 10 day ahead forecasts, neural networks are more accurate than AR(1)

model at 10% and 5% significance level. Therefore, we are able to detect predictive patterns in medium-term time horizon and coclude that neural networks are useful alternative for volatility forecasting.

References

[1] Alizadeh1, S., Brandtl, M., Diebold F. (2002): Range-Based Estimation of Stochastic Volatility Models, *The Journal of Finance*, 57 (3): 1047-1091.

[2] Akgiray, V. (1989): Conditional heteroscedasticity in time series of stock returns: Evidence and forecasts, *Journal of business*, 55-80.

[3] Andersen, T. G., Bollerslev, T., Diebold, F. X., Labys, P. (2003): Modeling and forecasting realized volatility, *Econometrica*, 71, 579-625.

[4] Atsalakis, G. S., Valavanis, K. P. (2009): Surveying stock market forecasting techniques.Part II: Soft computing methods. *Expert Systems with Applications*, 36 (3), 5932-5941.

[5] Baruník, J. (2008): How do neural networks enhance the predictability of central European stock returns? *Czech Journal of Economics and Finance (Finance a uver)*, 58 (07-08), 358-376.

[6] Baruník, J., Vacha, L. (2009) Neural networks with wavelet based denoising layer: Application to central European stock market forecasting, *IES Working Paper, No. 23/2009*

[7] Baruníková, M., Baruník, J. (2011): Neural networks as a semiparametric option pricing tool.

[8] Bachelier, L.(1900): Théorie de la spéculation, *Annales Scientifiques de l'École Normale Supérieure*, 3 (17): 21-86.

[9] Bali, T. G., Weinbaum, D. (2005): A comparative study of alternative extreme-value volatility estimators, *The Journal of Futures Markets*, 25 (9), 873-892.

[10] Bandi, F., Russell, J. (2006): Separating microstructure noise from volatility, *Journal of Financial Economics*, 79, 655-692.

[11] Bera, A., Jarque, C. (1980): Efficient tests for normality, homoscedasticity and serial independence of regression residuals, *Economics Letters*, 6 (3): 255-259.

[12] Bellman, R. E. (1961): Adaptive Control Processes, Princeton, *NJ: Princeton University Press.*

[13] Bishop, Ch. (2006): Pattern Recognition and Machine Learning, *Springer Information Science and Statistics Series*, ISBN 0-387-31073-8.

[14] Braun, H., Riedmiller M. (1994): Rprop - A Fast Adaptive Learning Algorithm, *Proceedings of the International Symposium on Computer and Information Science VII.*

[15] Broyden, C. G. (1970): The convergence of a class of double-rank minimization algorithms 2. The new algorithm, *IMA Journal of Applied Mathematics*, 6 (3), 222-231.

[16] Campbell, J. Y., Lo, A. W., MacKinlay, A. C. (1997): The econometrics of financial markets, *Princeton University Press*, Princeton, New Jersey. ISBN 0-691-04301-9.

[17] Christensen, K., Podolskij, M. (2007): Realized range-based estimation of integrated variance, *Journal of Econometrics*, 141 (2), 323-349.

[18] Constantinou, E., Georgiades, R., Kazandjian, A., Kouretas, G. P. (2006): Regime switching and artificial neural network forecasting of the Cyprus Stock Exchange daily returns, *International Journal of Finance & Economics*, 11 (4), 371-383.

[19] Cootner, P. (1964): The random character of stock market prices, *MIT Press*, ISBN 978-0-262-03009-0.

[20] Cowles, A. III (1944): Stock Market Forecasting, *Econometrica*, 12 (3-4): 206-214.

[21] Dayhoff, J. E., J. M. DeLeo (2001): Artificial Neural Networks: Opening the Black Box. *Cancer* 91 : 1615-1635

[22] Dickey A., Fuller, A. (1979): Distribution of the Estimators for Autoregressive Time Series with a Unit Root, *Journal of the American Statistical Association*, 74 (366): 427-431.

[23] Diebold, F.X. and R.S. Mariano (1995): Comparing Predictive Accuracy, *Journal of Business and Economic Statistics*, 13: 253-63.

[24] Donaldson, R. G., Kamstra, M. (1997): An artificial neural network-GARCH model for international stock return volatility, *Journal of Empirical Finance*, 4 (1), 17-46.

[25] Du, K. L., Swamy, M. N. S. (2013): Neural networks and statistical learning, *Springer Science & Business Media*, ISBN 978-1447155706.

[26] Engle, R. F., Bollerslev, T. (1986): Modeling the persistence of conditional variances, *Econometric Reviews*, 5 (1): 1-50.

[27] Fama, E. F. (1965): Random Walks In Stock Market Prices, *Financial Analysts Journal*, 21 (5): 55-59.

[28] Fama, E. F. (1970): Efficient Capital Markets: A Review of Theory and Empirical Work, *Journal of Finance*, 25 (2): 383-417.

[29] Fama, E. F.(1991): Efficient Capital Markets: II, *Journal of Finance*, 46 (5): 1575-1617.

[30] Fama, E. F., French, K. (1989): Business conditions and expected returns on stocks and bonds, *Journal of financial economics*, 25 (1): 23-49.

[31] Fama, E. F., French, K. (1996): Multifactor explanations of asset pricing anomalies, *The journal of finance*, 51 (1), 55-84.

[32] Fama, E. F., French, K. (2008): Dissecting anomalies, *The Journal of Finance*, 63 (4), 1653-1678.

[33] Fletcher, R. (1970): A new approach to variable metric algorithms, *The computer journal*, 13 (3), 317-322.

[34] Garman, M.B., Klass, M.J., (1980): On the estimation of security price volatilities from historical data, *The Journal of Finance*, 53 (1), 67-78.

[35] Gregoriou, G. N. (2009): Stock market volatility. *CRC Press.* ISBN 978-1-4200-9954-6

[36] Goldfarb, D. (1970): A family of variable-metric methods derived by variational means, *Mathematics of computation*, 24 (109), 23-26.

[37] Gonzalez Miranda, F., Burgess, N. (1997): Modelling market volatilities: the neural network perspective, *The European Journal of Finance*, 3 (2), 137-157.

[38] Hajizadeh, Ehsan, et al. (2012): A hybrid modeling approach for forecasting the volatility of S&P 500 index return, *Expert Systems with Applications*, (39) 1: 431-436.

[39] Hamid, S. A., Iqbal, Z. (2004): Using neural networks for forecasting volatility of S&P 500 Index futures prices, *Journal of Business Research*, 57 (10), 1116-1125.

[40] Hansen, P. R., Lunde, A. (2006): Realized variance and market microstructure noise, *Journal of Business and Economic Statistics*, 24, 127-161.

[41] Hornik, K. (1991): Approximation capabilities of multilayer feedforward networks, *Neural networks*, 4 (2), 251-257.

[42] Jarque, C. M., Bera, A. K. (1980): Efficient tests for normality, homoscedasticity and serial independence of regression residuals, *Economics letters*, 6 (3), 255-259.

[43] Jolliffe, I. (2002): Principal Component Analysis, *Springer Series in Statistics*, 2nd ed. p. 28. ISBN 978-0-387-95442-4.

[44] Kanas, A., Yannopoulos, A. (2001): Comparing linear and nonlinear forecasts for stock returns, *International Review of Economics & Finance*, 10 (4), 383-398.

[45] Kandel, E.R., Schwartz, J. H., Jessel, T. M. (2000): Principles of Neural Science," Ch. 2: Nerve cells and behavior", *McGraw-Hill Professional.* ISBN 978-0-8385-7701-1.

[46] Kendall, M. (1953): The Analysis of Economic Time-Series-Part I: Prices, *Journal of the Royal Statistical Society. A (General) (Blackwell Publishing)*, 116 (1): 11-34.

[47] Kristjanpoller, W., Fadic, A., Minutolo, M. C. (2014): Volatility forecast using hybrid Neural Network models, *Expert Systems with Applications*, 41 (5), 2437-2442.

[48] Lo, A. W., MacKinlay, A. C. (1988): Stock market prices do not follow random walks: Evidence from a simple specification test, *Review of financial studies*, 1 (1), 41-66.

[49] McNelis, P. D. (2005): Neural networks in finance: gaining predictive edge in the market, *Academic Press*, ISBN 978-0124859678

[50] Mandelbrot, B. B. (1963): The Variation of Certain Speculative Prices, *The Journal of Business*, 36 (4): 394-419.

[51] Martens, M., van Dijk, D. (2007): Measuring volatility with the realized range, *Journal of Econometrics*, 138, 18-207.

[52] McCulloch, W. S., Pitts, W. (1943): A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5 (4), 115-133.

[53] Monfared, S. A., Enke, D. (2014): Volatility Forecasting Using a Hybrid GJR-GARCH Neural Network Model, *Procedia Computer Science*, 36, 246-253.

[54] Mörters, P., Peres, Y. (2010). Brownian motion (Vol. 30). Cambridge University Press.

[55] Nocedal, J., Wright, S. (2006): Numerical optimization, *series in operations research and financial engineering*, Springer, New York, USA, ISBN 978-0-387-40065-5.

[56] Nocedal, J., Wright, S. (2006): Numerical optimization, *Series in operations research and financial engineering. Springer*, New York, USA. p. 192-201.

[57] Ormoneit, D., Neuneier, R. (1996): Experiments in predicting the German stock index DAX with density estimating neural networks, In *Computational Intelligence for Financial Engineering, Proceedings of the IEEE/IAFE 1996 Conference* on (pp. 66-71). IEEE.

[58] Parkinson, M., (1980): The extreme value method for estimating the variance of the rate of return, *Journal of Business*, 53 (1), 61-65.

[59] Pigorsch, U., Pigorsch, C., Popov, I. (2011): Volatility estimation based on high-fre- quency data, in J.-C. Duan, J. E. Gentle, W. Hardle (Eds.), Handbook of computational finance. *Springer Science & Business Media.*

[60] Regnault, J. (1863): Calcul des Chances et Philosophie de la Bourse, *Paris: Mallet-Bachelier and Castel.*

[61] Riedmiller, M., Braun, H. (1993): A direct adaptive method for faster backpropagation learning: The RPROP algorithm,

In Neural Networks, 1993, IEEE International Conference on (pp. 586-591) IEEE.

[62] Rogers, L., Satchell, S. (1991). Estimating variance from high, low and closing prices, *Annals of Applied Probability*, 1, 504-512.

[63] Roh, T. H. (2007): Forecasting the volatility of stock price index. *Expert Systems with Applications*, 33 (4), 916-922.

[64] Rojas, R. (1996): Neural networks: a systematic introduction, *Springer Science & Business Media*, ISBN 978-3-642-61068-4.

[65] Samuelson, P. (1965): Proof That Properly Anticipated Prices Fluctuate Randomly, *Industrial Management Review*, 6: 41-49.

[66] Sarle, W. S. (1999): Ill-Conditioning in Neural Networks , *SAS Institute Inc.*, ftp://ftp.sas.com/pub/neural/illcond/illcond.htm

[67] Schittenkopf, C., Dorffner, G., Dockner, E. J. (1998): Volatility prediction with mixture density networks, *Springer London*: 929-934.

[68] Shanno, D. F. (1970): Conditioning of quasi-Newton methods for function minimization, *Mathematics of computation*, 24 (111), 647-656.

[69] Shu, J., Zhang, J. E. (2006): Testing range estimators of historical volatility, *The Journal of Futures Markets*, 26 (3), 297-313.

[70] Thawornwong, S., Enke, D. (2004): The adaptive selection of financial and economic variables for use with artificial neural networks, *Neurocomputing*, 56, 205-232.

[71] Todorova, N., Husmann, S. (2012): A comparative study of range-based stock return volatility estimators for the German market, *Journal of Futures Markets*, 32 (6), 560-586.

[72] Tseng, C. H., Cheng, S. T., Wang, Y. H., Peng, J. T. (2008): Artificial neural network model of the hybrid EGARCH volatility of the Taiwan stock index option prices, *Physica A: Statistical Mechanics and its Applications*, 387 (13), 3192-3200.

[73] Verner, R. (2011): Stock Markets Analysis Using New Genetic Annealed Neural Network, Prague, 134 s. Master thesis (Mgr.). Charles University in Prague, Faculty of Social Sciences, Institute of Economical Studies. Department of Macroeconomics and Econometrics. Supervisor: PhDr. Jozef Baruník.

[74] Wang, X., Phua, P. K. H., Lin, W. (2003): Stock market prediction using neural networks: does trading volume help in short-term prediction? In *Neural Networks, Proceedings*

of the International Joint Conference on (Vol. 4, pp. 2438-2442). IEEE.

[75] Zhu, X., Wang, H., Xu, L., Li, H. (2008): Predicting stock index increments by neural networks: The role of trading volume under different horizons, *Expert Systems with Applications*, 34 (4), 3043-3054.

[76] Zorin, A., Borisov, A. (2002): Traditional and index tracking methods for portfolio construction by means of neural networks, *network*, 1, 1.