Analysing Economic Data with Self-Organizing Maps

A Geometric Neural Network Approach

Lars Edler

Department of Economics
Faculty of Economics, Business, and Social Sciences
Christian-Albrechts-University of Kiel

July 2007

Analysing Economic Data with Self-Organizing Maps

A Geometric Neural Network Approach

Inaugural-Dissertation zur Erlangung des akademischen Grades
eines Doktors der Wirtschafts- und Sozialwissenschaften
der Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Christian-Albrechts-Universität zu Kiel

vorgelegt von
Diplom-Volkswirt Lars Edler
aus Rotenburg/Wümme

Kiel, Juli 2007

Christian-Albrechts-Universität zu Kiel
Institut für Volkswirtschaftslehre
Lehrstuhl für Geld, Währung und Internationale Finanzmärkte
Wilhelm-Seelig-Platz 1
24118 Kiel
Germany

Gedruckt mit Genehmigung der
Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Christian-Albrechts-Universität zu Kiel

Dekan:
Prof. Dr. Helmut Herwartz

Erstberichterstattender:
Prof. Dr. Thomas Lux

Zweitberichterstattender:
PD Dr. Uwe Jensen

Tag der Abgabe der Arbeit:
27. Juli 2007

Tag der mündlichen Prüfung:
02. November 2007

To My Family

And to a person without whom
I had never started this project.

Contents

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ANT	Ant Algorithm
AR	Autoregressive
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
Avg.	Average
BHHH	Berndt-Hall-Hall-Hausman Algorithm
BMU	Best-Matching-Unit
CA	Cluster Analysis
CCA	Curvilinear Component Analysis
CDC	Correct Directional Change Statistic
CEV	Conditional Expectation Value Method
CI	Computational Intelligence
CIS	Community of Independent States
CPI	Consumer Price Index
CPU	Central Processing Unit
DA	Discriminant Analysis
DAX	Deutscher Aktion Index
DBI	Davies-Bouldin-Index
DI	Dunn's Index
DM	Diebold-Mariano Test
DTree	Decision Trees
DW	Durbin-Watson Test
EMH	Efficient Market Hypothesis
ES	Evolutionary Strategies
EP	Evolutionary Programming
EU	European Union
FA	Factor Analysis
FSA	Finite State Automata
FX	Foreign Exchange

GA	Genetic Algorithm
GARCH	Generalised Autoregressive Heteroskedasticity
GP	Genetic Programming
IFS	International Financial Statistic
IMF	International Monetary Fund
JB	Jarque-Bera Test
KNN	k-nearest Neighbours
LPR	Local Polynomial Regression
LB	Ljung-Box Test
LS	Least Square Method
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MDS	Multi-Dimensional Scaling
MLFNN	Multilayer Feedforward Neural Network
MPNN/MLP	Multilayer Perceptron Neural Network
NASDAQ	National Association of Securities Dealers Automated Quotations
NN	Neural Network
NYSE	New York Stock Exchange Index
OLS	Ordinary Least Square Method
PCA	Principal Component Analysis
PT	Pesaran-Timmermann Test
RBF	Radial Basis Function
RBFN	Radial Basis Function Network
RMSE	Root Mean Squared Error
RW	Random Walk
SA	Simulated Annealing
SP	Sammon's Projection
SOM	Self-Organizing Map
SVM	Support Vector Machine
SWARM	Swarm Intelligence
VQ	Vector Quantisation
WLS	Weighted Least Square Method

The country codes used in chapter 7 are summarised in table 7.1.

List of Figures

List of Tables

Introduction

[...] a good billiards player may have no knowledge of physics.
Taylor and Allen (1992)

Is technical trading profitable? This is the question a vast amount of literature has been dealing with in recent years. Basic finance theory such as the Efficient Market Hypothesis (EMH) denies the existence of the possibility to gain excess profits by e.g. analysing charts and exploit their trends. This assumption is linked with the idea of an arbitrage-free market: the standard argument is that once a new technical tool proves to be profitable it is adopted by the market very quickly and the technical advantage – and therefore the arbitrage opportunity – vanishes rapidly. These considerations are convincing and widely regarded as realistic, but inherently there is a small profitable time window during which the developer of a new technical method is able to gain excess return. This window lies exactly between the development of the new technology and its adoption by the market and will most likely exhibit a monotonically decreasing excess return curve. This surplus is the incentive for analysts to pursue the invention of new chartists' techniques.

Economic theory tries to explain the trajectories of prices and/or returns by simplified models that – in the best case – should capture the most influencing features of the data generating process. In order to make these approaches mathematically tractable it is essential to impose (rigourous) restrictions on the assumptions. This is convenient for computation and simulation of the system but does also restrain the model to a less realistic one. This is the central trade-off in economic modelling and so far there is no such a thing as a "one-captures-it-all" model. From an economist's point of view theoretically founded models are the preferable way to predict the future development of prices or returns and to explain the mechanisms of market forces, but chartists appear to perform very well on financial market and this class of agents has not been studied extensively so far. Taylor and Allen get to the point by saying "[...] a good billiards player may have no knowledge of physics."[1]. The reason why successful trading on financial markets does not necessarily require a profound theoretical knowledge is obviously incompleteness of finance literature. Markets

[1] Taylor and Allen (1992)

are far too complex to be explained by "representative agents" and a number of simplifications and restrictions that have to be imposed in a model's environment. Recent models allow for heterogeneous agents and accept the fact that the market is an institution in which a huge number of individuals interact. Of course, this approach increases the degree of complexity and the dynamics are more complicated to model than in representative-agent-world. Well-known examples for this generation of models are Lux & Marchesi[2], Kirman[3], or Brock & Hommes[4] among others.

Fluctuations in financial markets are usually explained either from the Newclassical or the Keynesian viewpoint. The former one attributes fluctuations to the arrival of exogenous, random shocks that occur to fundamentals, e.g. through the arrival of new information (a typical example for new information would be an unanticipated rise of the money market rate). The Keynesian explanation does not explicitly exclude the existence of exogenous shocks, but focuses on the famous Keynesian idea of *animal spirits*. Therefore, the Newclassical approach can be interpreted as the workhorse for the employment of rational expectations concept, whereas the Keynes' animal spirits rely more on psychological factors in markets. Let's assume the Keynesian explanation holds true, then the question arises whether models based on economic fundamentals are able to capture "soft" influences.[5] Does it really suffice to simplify the group of agents to a representative rational prototype who is always acting as a utility-maximising *homo oeconomicus*? Empirical tests of the corresponding class of models prove that there is something beyond (linear) modelling of rational behaviour. Non-linearities in economic time series are not ultimately explained by traditional approaches and economists started to accept this fact during that last two decades by setting up a whole new family of models. These models approach economic problems from a different perspective. Originally, economists built artificial environments in which a stylised model world could be embedded. These frameworks usually consist of several constraints and (ad hoc) premises. This approach is very convenient to formulate mathematically tractable systems and in many cases these are analytically feasible. One major drawback of this methodology is the lack of the goodness of fit, i.e. often theoretical models do not reflect the real trajectories of financial time series. Therefore, a new branch of economic research has been established that can be couched best by Carlos Serrano-Cinca's demand to "let the data speak for themselves"[6]. The idea is to pre-process and analyse the data with appropriate methods and the find a model (or: functional form) that captures the characteristics of the time series best. A theoretical foundation might or might not be found afterwards.

[2]Lux and Marchesi (1999) and Lux and Marchesi (2000)
[3]Kirman (1993)
[4]Brock and Hommes (1997)
[5]By "soft" influences we mean all market driving forces that are not based on the influence of a change in the fundamental data.
[6]Deboeck and Kohonen (1998) p. 3

The method proposed in the thesis at hand pursues an analogous approach. Self-Organizing Maps are from their very nature a special form of neural networks but have two big advantages. First, it is **not** a black-box technique and, second, it avails itself of **unsupervised learning**. These two features are widely perceived as superior to the construction of traditional neural networks. Indeed, it has long been criticised that for the appliance of neural networks a functional form has to be given in advance and that it is unclear what happens within the network layers during the training of the net. Self-Organizing Maps avoid these two issues by a fundamentally different construction of the training algorithm and are, therefore, suitable for a large variety of data-based analyses.

When the Self-Organizing Map algorithm was invented in the early 1980s its application range focused on the auto-organisation of large databases and the improvement of digital imaging by extrapolating the colour information between pixels. In subsequent experiments the algorithm was successfully applied to biological and medical problems. Only recently Self-Organizing Maps have entered economic research (or: finance research). One of the first authors who established the link between these two fields of research was Guido Deboeck who edited a very nice collection of essays on this topic in 1998.[7] Against the background of the algorithm's roots it becomes clear why it is so powerful in the graphical representation and visualisation of complex data sets. This is probably the main advantage and the strongest interceder in favour of the method. Throughout this thesis every now and then we will consult graphical analysis in order to substantiate interpretations and assumptions. It is particularly easy to discover new features and links between observations in a large data set and we will make use of this convenient property.

Within the scope of this thesis Self-Organizing Maps are used for different problems and applications. It will become apparent that it is difficult to integrate the method into theoretical frameworks, but the big advantage lies in the identification and visualisation of previously unknown data features and characteristics. Through the efficient graphical representation of (multivariate) complex data sets regularities are found and can be exploited for further use in terms of qualitative research.

The thesis is organised in three main parts. In the first one, a short chapter is devoted to offer the motivation for this work and to give an overview of important concepts in empirical finance which also underlie the application of Self-Organizing Maps to financial markets. We then turn to a discussion of non-linear models and approaches commonly used in this area of research. This is subsequently extended to neural networks in various forms which builds the bridge to the theory of Self-Organizing Maps. The third chapter deals with the design of Self-Organizing Maps. Since the method is the core of this thesis we spend a longer section on the technical details because the algorithm itself is quite new to economics and so it is worthwhile

[7] Deboeck and Kohonen (1998)

to take a closer look on Self-Organizing Maps' formal background. The second main part of the thesis consists of four chapters and provides different possibilities for the employment of the method in economics. In chapter four a visual exploration of prices is accomplished which is extended in chapter five to forecasting of prices. Chapter six offers an alternative model to predict prices, but allows for the integration of state-probabilities. These probabilities will also be obtained by employment of Self-Organizing Maps. In the last chapter of part two we leave the field of financial time series and turn to the clustering of different countries through the presentation of macroeconomic data to a Self-Organizing Map. Lastly, the third part is reserved to summarise and to evaluate the results obtained in the thesis at hand. Moreover, a conclusion and an outlook for future research is given.

Part one

Theoretical Considerations and Classification of Methods

Chapter 1

Traders, Market Efficiency, and Financial Patterns

1.1 Introduction

High-frequency asset returns are linear unpredictable, conditionally heteroscedastic, and unconditionally leptokurtic.[1] This perception is widely accepted in finance since decades. The effort to find a linear model which is able to beat the random walk in the out-of-sample prediction of returns has been subject to an impressive amount of empirical studies in economic literature.[2] All of these linear models have not led to an improvement compared to the simple random walk. In other words, the best predictor of tomorrow's return is today's realisation.

For convenience, economists tend to linearise their models in order to make them mathematically tractable. In many cases this makes life easier but may also simplify the world too much which, in turn, leads to poor forecasting or explanatory power. This is the reason why most of the rigorous economic models fail when used to forecast e.g. the exchange rate.[3] Nonlinearities in the conditional means of time series seem to be responsible for a large part of these difficulties and economists have come up with considerable progress in the development of models able to capture these effects during the 1990s. It is already a widely accepted fact that economic theory cannot rule out nonlinearities in economic system or data generating processes, respectively. In fact, it is actually this nonlinear behaviour which plays an important role in the determination of asset returns or the rate of change of the exchange rate.[4] Nonlinear approaches are afflicted with the danger of overfitting, i.e. outliers in the data (or the time series) have to be considered carefully. Otherwise, single outliers or other less important oddities in the data would influence the convergence behaviour of the model too much and would consequently bias the results. This is where the preprocessing of data comes into play. A section in chapter 3 is devoted to give some insight into the main techniques and methods of preprocessing.

[1] Diebold and Nason (1990)

[2] See e.g. Cootner (1964), Fama (1965), and Meese and Rogoff (1983).

[3] Meese and Rogoff (1983)

[4] The rate of change of the exchange rate is approximated by $\Delta \ln S_t$, where S_t is the nominal exchange rate at time t.

It is a reasonable assumption to attribute the existence of non-linearities to the diversity of traders that are acting on financial markets. Traditional finance theory acts on the assumption that one rational agent is a good representation of all individual market participants. But this seems to be rather an ad-hoc assumption. Each investor – regardless of his nature – consists of a different "bundle" of socio-cultural factors, such as education, experience of life, dissimilar intelligence quotient et cetera. These factors are unlikely to cancel out in case their number increases. Furthermore, there will be some sort of correlation between the different influences that drives certain opinion formation processes.[5] These are the channels through which non-linearities enter the data generating process on financial markets. In other words, non-linearities and chaos are not constitutional elements, but are generated by market participants. In terms of the thesis at hand non-linearities are an important argument in favour of the employment of sophisticated non-linear methods and complex numerical techniques. Therefore, in this chapter we will give a review of trader types and the *Efficient Market Hypothesis*.

1.2 Types of Traders

What are the market forces that drive the price generating process? Considering the market place as an institution where social individuals exchange goods and services (including financial assets) its complexity becomes obvious. Due to the intention of extremely heterogenous beings to trade with each other, the market structure becomes far too complicated to formalise the events in a linear fashion. Furthermore, these individuals are relatively hard to aggregate. Hence, realistic economic modelling is a difficult business when the researcher tries to capture the entire coherency.

At this point of the thesis it is convenient to introduce the concept of the distinction between trader types. It is important to give a definition of the nomenclature since it forms a significant part of the motivation underlying this work. Of course, due to the heterogenous nature of agents the classification into trader types can only be a rough approximation, but it helps to develop a better idea of the market structure.

As it will become clear later on, a promising way to achieve successful forecasting of financial and/or economic time series by means of technical analysis methods (or: chartist techniques), is to exploit the weaknesses of the information processing on suppositional efficient markets. These weaknesses are mainly due to technical limitations of extracting relevant information from the data at hand. Markets are

[5] For instance, the level of experience and the distribution of wealth among the agents are likely to be positively correlated.

not only driven by the arrival and processing of new information. They are rather a complex system of many heterogenous agents (social individuals) who take decisions not only on the basis of some rationale, but also under consideration of experience, educational background, or socio-cultural motives. It is very difficult to classify these extremely heterogenous groups into a reasonable system and this task will not be pursued here. However, it is possible to – at least – arrange traders by their main objective, which is usually reflected in their activity time horizon.

There is a vast amount of economic data available and this quantity grows every day.[6] In the era of the instantaneous availability of e.g. financial time series quantitative data analysis and highly sophisticated statistical methods have become crucial tools for both, research and practical application, such as the development of investment strategies conducted by banks. Researchers start switching their attitude to a philosophy that might be explained best by Carlos Serrano-Cinca's aforementioned demand "let the data speak from themselves".[7] It becomes more and more important to deduct a theoretical framework from the keen analysis of data rather than to develop a theoretical model in advance and then falsify on the basis of real world data. Financial time series are from their very nature a mirror of the economic environment in which a market or a company stands. To non-institutional investors the development of the stock price or the behaviour of an index is often the only available information and in many cases the private investor is the last person to know the information that arrives on the market. But how do financial analysts of banks and funds decide on the evaluation of a certain new information? How does this information affect the distribution of the time series?

Before we take a closer look at this question we first have to differentiate between the two extremes of acting agents on financial markets. Following Frankel and Froot[8] these two groups can be divided by the characteristics described below:

- **Fundamentalists**: This group of agents relies on an underlying economic model. In the case of the foreign exchange markets this could be the *Dornbusch-Overshooting-Model*[9]. This model is supposed to work completely correct in a world without chartists.

- **Chartists**: The only information that counts for chartists is the time series of the economic variable in question (e.g. the past values of the exchange rate). Past realisations of these variables are extrapolated into the future in order to

[6] The available data sets not only grow in terms of different measures, but also the *frequencies* of the observations have been increased. For instance, due to exponentially increasing computational power it is no longer a problem to process prices that are observed in seconds (or even smaller time units).
[7] Serrano-Cinca (1998)
[8] Frankel and Froot (1986)
[9] Dornbusch (1976)

derive a forecast.

Furthermore, the individuals acting on financial markets need to be differentiated by means of the trading behaviour with respect to the time horizon. Obviously, this factor is a very important one since the time horizon influences in a direct manner the chart patterns we are contemplating. In terms of the time horizon we can split the group of agents roughly into three different types:

- **Day-Traders:** This is a group of agents who are in and out of a trade during a single day.
- **Position-Traders:** This term refers to agents who hold the trade longer than a day but not forever. This group of market participants likes to use chart patterns in order to find entry and exit points in a convenient way.
- **Buy-and-Hold Investors:** This is probably the only group out of these three that deserves the term "investor". They are not looking at the short-term development of stocks and indices. These agents are interested in the long-term development of investments. Many private investors are acting as Buy-and-Hold Traders because they simply do not have the time to observe the markets as closely as it would be necessary if they were e.g. Day-Traders. Nevertheless, even for Buy-and-Hold investors chart patterns are a very convenient and common tool.

The above-named categories of investor types are certainly only a rough classification. Between these three groups there exist a whole bunch of nuances, e.g. investors who decide to minimise risk by diversifying their portfolio not only with respect to underlying values but also with respect to the time horizon of a trade.

1.3 Efficient Market Hypothesis

When dealing with the diffusion of information on markets we cannot avoid to take a detailed look at a very famous concept in finance: the Efficient-Market-Hypothesis[10]. The EMH is founded by the idea that on an informationally efficient market the changes of prices fluctuate randomly if they completely incorporate the expectations and information of all market participants. In this case one can speak of "properly anticipated prices".[11] In the words of Fama[12] the significance of this sentence can be interpreted as the "full reflection" of information in the prices of

[10] The Efficient-Market-Hypothesis is hereafter called EMH.
[11] Campbell et al. (1997) p. 20
[12] Fama (1970)

a certain market. In a more formal way it can be said that a market is efficient if the price is unaffected by revealing information to all market participants.[13] The definition by Malkiel (1992) opens the possibility of econometric testing, but this is hard to accomplish because the quantification of information is very difficult. For this reason, the usual approach of empirical research to test for market efficiency is by measuring the profits that can be made by trading on information. To this end an information set has to be determined and it is measured if hypothetical trading on this information would lead to superior profits. The choice of the information set is the crucial factor behind this approach.

There are basically three types of sets which determine the level of efficiency. These types were categorised in 1967 by Roberts[14]:

- **Weak-Form Efficiency:** Only the history of prices is included in the information set.
- **Semistrong-Form Efficiency:** The so-called *publicly available* information is included in the information set, i.e. all the information known to *all* market participants.
- **Strong-Form Efficiency:** All information known to *any* market participants is included, i.e. also the private information.

The categorisation above makes it easier to understand the complexity of the manifold ingredients a market consists of. The EMH in its weak form implies that the development of prices in time follow a random walk, i.e. the best predictor for tomorrow's price is its current value. This implies that every new information arriving on the markets is instantaneously incorporated in the prices. The prices reflect all available information. In the following paragraphs the formal background of the EMH is presented.

Assuming the EMH holds true, the evolution of the price can be formulated as a random walk ($p_t \equiv \ln P_t$)

$$p_{t+1} = p_t + \epsilon_t, \tag{1.1}$$

where p_t is the logarithm of the price and ϵ_t is a Gaussian variable with zero mean and variance σ^2. Therefore, the best predictor for p_{t+1} at time t is

$$\hat{p}_{t+1} = p_t. \tag{1.2}$$

These considerations are true if the time series truly follows a random walk. If there is some predictable component in the process, this leads to a slightly different formulation:

[13] Malkiel (1992)

[14] Roberts (1967) and Campbell et al. (1997) p. 22

$$p_{t+1} = p_t + f(p_t, p_{t-1}, \ldots, p_{t-n+1}) + \epsilon_t, \tag{1.3}$$

where $f(\cdot)$ represents the forecastable part of the process. Again, ϵ_t is a Gaussian random variable with zero mean and variance σ^2, and the function $f(\cdot)$ is nonlinear in its arguments. Given forecastibility of $f(\cdot)$, the best estimator for p_{t+1} is

$$\hat{p}_{t+1} = p_t + f(p_t, p_{t-1}, \ldots, p_{t-n+1}). \tag{1.4}$$

In many cases price time series contain a trend. Once the model is fitted to a series of data it might be able to capture the characteristics of the process in sample, but it could exhibit very poor performance out-of-sample because the trend is not captured by the estimates. For that reason it is extremely difficult to forecast with the model described above. This is shown in figure 1.1 where the model can be fitted easily to the data in section 1 of the price trajectory. On the other hand using the estimates for prediction of the data contained in section 2 will lead to very problematic results.

In order to avoid this sort of problems financial time series are usually modeled on the basis of first order differences.[15] The model collapses to

$$\Delta p_{t+1} = g(\Delta p_t, \Delta p_{t+1}, \ldots, \Delta p_{t-n+1}) + \eta_t, \tag{1.5}$$

with $\Delta p_{t+1} = p_{t+1} - p_t$, η_t a Gaussian variable with $\mu = 0$ and variance σ^2 and $g(\cdot)$ is a function which is nonlinear in its arguments. Hence, the best predictor for this model is

$$\Delta\hat{p}_{t+1} = g(\Delta p_t, \Delta p_{t+1}, \ldots, \Delta p_{t-n+1}). \tag{1.6}$$

The objective of the thesis is the application of a sophisticated technical method that uses the geometric characteristics of (financial) patterns to extract information from the data that is not directly observable. Therefore, the main question posed here is which sort of information is contained in geometrical symbols (patterns). The next section offers some insight into the history and the present state of research in this sector.

1.4 Information Content of Financial Patterns

The first technical analyst known in literature was William Peter Hamilton, the second editor of the *The Wall Street Journal* from 1902 to 1929. It was Hamilton who picked up the work of his predecessor Charles Henry Dow and used the Dow-Theory as the vehicle to predict the market on the basis of stock market movements.

[15] Granger and Newbold (1986)

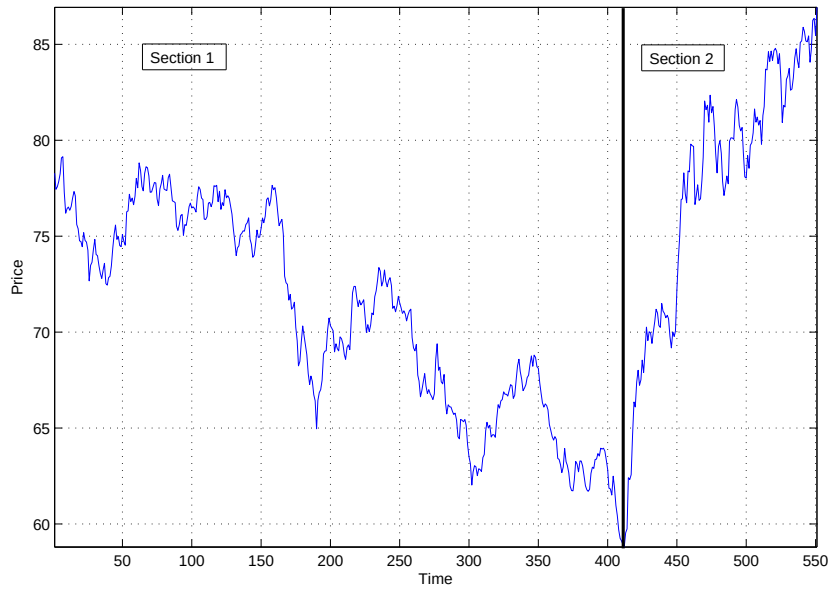


Figure 1.1: Example for a forecasting problem of a time series with trend. A global model might be able to fit section 1 nicely, but it will probably fail to predict section 2.

The basic assumption of the Dow-Theory presupposes that the market moves in persistent bear and bull periods. Following Hamilton the determination of these trends can be used for the analysis of market movements which, in turn, results in the possibiliy of prediction. The identification of trends becomes complicated only through short-term deviations.[16] The notion of Hamilton was the signal for a whole industry to focus on the development of technical analysis tools for financial markets – more or less sophisticated ones. Since that time economics literature basically tried to confute the hypothesis that technical trading is profitable and usually surveys among traders disapprove the existence of profitable technical trading in the long-term horizon, but admit a certain success in the short-term.[17] But then the question arises why e.g. "at least 90 per cent [of chief economists trading on the London FX market] place some weight on this form of non-fundamental analysis [...]"[18].

It is still an economic puzzle why pure chartists seem to perform well on financial markets.[19] In many occasions we can observe the prices drifting off their fundamental values. The observable price cannot be explained on the basis of economic variables but may rather be attributed to a purely technical behaviour of traders. The

[16] Brown et al. (1998)
[17] See e.g. Taylor and Allen (1992), Frankel and Froot (1986) and Frankel and Froot (1990).
[18] Taylor and Allen (1992)
[19] Chen and Tsao (2003)

phenomenon became first of academic interest in 1986 when the paper *The Dollar as an Irrational Speculative Bubble: A Tale of Fundamentalists and Chartists* by Frankel and Froot was published. This paper was followed in 1990 by *Chartists, Fundamentalists, and Trading in the Foreign Exchange Market.*[20] Considering as example the exchange rate market, Frankel and Froot claim that large appreciations of the exchange rate, especially during bubble periods, cannot be traced back to the fundamental variables of the economy. In case of the dollar this means that the currency was appreciating significantly in 1984/85, even though the real interest rate differential had already started to decrease. If there are market participants who act as "technical traders" they can influence the swings of prices or exchange rates, respectively, with their actions. The paper by Taylor and Allen (1992) emphasises the significance of chartist methods in practice and, therefore, for academia, too.

Chartists usually use techniques based on past realisations of variables in question and extrapolate them into the immediate future. They might also use the correlations between historical values in order to extract information from the past structure of the data. Another possible approach to technical trading is the assumption that a price time series is simply a limited number of recurrent symbolic patterns that are stringed together in a complex order. This idea is very much in the spirit of the long memory idea.[21] Recurrent patterns are representative geometrical structures for a certain market constellation that not only embraces fundamental data but that also receives impulses from the technical analyst's side of the market. Hence, this sort of model can be perceived as being more complete than a pure fundamentalist's model in terms of the information that is being processed. This, of course, hypothesises a certain information content of the patterns and the feasibility of extracting relevant information from these "symbols". The latter task will be one of the subject-matters of the dissertation at hand.

Authors such as Bulkowski (2000) lend support to the hypothesis that sophisticated application of chart pattern methods are able to outperform the market. In his book *Encyclopedia of Chart Patterns* Bulkowski argues that the trajectory of a financial market consists of a variety of different patterns which are observable on a recurrent basis. This point of view implicitly assumes a certain information content of patterns - but does not prove its existence. To the best of the author's knowledge there has not been accomplished any research on this challenge. The only exception might be found in the work by Chen and Tsao (2003) who try to verify information content by means of an event-study analysis. The results are promising and stimulate a new field of research.

The departure point of this thesis is the presumption that financial patterns contain an unknown quantity of information that is not – or at least not completely

[20] Frankel and Froot (1986) and Frankel and Froot (1990)
[21] See e.g. Ding et al. (1993).

– processed by the market. This assumption does not necessarily conflict with the EMH that economists usually impose. Indeed, the information is hidden in the data but nobody possesses the technology to extract it. Hence, the information is not available, it is worth nothing and is consequently not reflected in the prices. But then, this opens an opportunity for successful trading. If a single agent manages to achieve technical advances, she might be able to benefit from the information advantage. Once this new technology has proven to be capable of e.g. forecasting better than other methods, it is likely to be mimicked by the competitor agents, and the marginal revenue generated by using the new technique successively vanishes as more and more traders make use of it. So, there is a very small profitable time window for trading, and the profits that can be generated by searching and using innovative methods for data analysis might be labelled best by the term *winning margin of financial markets*.

The aim of this work is to offer a method that is capable to identify patterns. Once this is successfully done it should be possible to use this supplementary information accrued from the analysis and to convert it into profitable trading schemes.

1.5 Data Mining in Finance

The term *data mining* refers mainly to a new generation of computing techniques.[22] The aim of the development of data mining methods is the discovery of hidden patterns or trends in data, such as financial time series or other economic databases. It is a dangerous undertaking to follow trends on financial markets because it is still an ongoing academic controversy whether there exist informative patterns among the vast amount of data that is available nowadays. Hence, the focus of research is the question of how to discriminate between profitable real trends and useless, randomly distributed illusions.

Data mining is a combination of two essential technologies: database management and machine learning techniques. Since researchers and practitioners have comprehensive access to a huge amounts of data and different frequencies of these data the necessity of database management is obvious. Databases tend to grow in the future because cost of storage space will decrease and therefore it will become more and more important to organise this surge of information in an efficient way. On the other hand, machine learning represents the branch of computing sciences that develops software tools which allow for learning from experience. This group of computer programmes (e.g. Neural Networks or Genetic Algorithms) are already applied successfully to many practical problems. Software tools will become more sophisticated and advanced in future and leads to a very efficient division of labour

[22] Sullivan et al. (1998)

between man and computer. Data mining is an extensive term that covers an application range from classical statistics to extremely complex programming methods.

In their book Kovalerchuk and Vityaev (2000) look at ten different analyses which can be subsumed under the concept of data mining: Fundamental Analysis, Technical Analysis, Autoregression, Neural Networks, Genetic Algorithms, k-Nearest Neighbours, Markov Chains, Decision Trees, Hybrid Methods and Relational Data Mining.[23] As one can see from this listing, data mining serves as an umbrella to unify a whole diversity of aspects of data-based research. Some of these methods will be described later on, others will simply be related to the current literature, and some of them are beyond the scope of this thesis and will therefore not be treated here.

Kovalerchuk and Vityaev (2000) characterise data mining methodologies by the *data type*, the *data set*, and the *mathematical algorithm* used. Data types can be either attributes or relations, whereas the data set consists of either the time series itself or a set of variables that may affect the series' dynamics in some way. *Fundamental* and *technical analysis* are closely connected with the respective data set choice. Fundamental analysis tries to identify every variable that influences the dynamics of a certain economic variable. Technical analysis assumes that every information necessary for forecasting the trajectory of a certain economic variable is contained in the time series itself. Finally, the mathematical algorithm gives the technical method that is used to analyse the data, e.g. a Neural Network model.

The *learning* of an algorithm is either *supervised* or *unsupervised*. The unsupervised version of learning is more flexible because it does not stipulate any known classes for training examples, i.e. no design of the network specified.[24] In contrast, for supervised learning the number of classes for training examples is known. The desired aim in machine learning is to find regularities or recurrent patterns in a set of training (or: historical) data which can be exploited and used for forecasting and/or develop profitable trading strategies. This goal is – in parts – pursued in this thesis, too. This subsection served to give a short overview of data mining but is not an exhaustive review of all the concepts summarised by the term. There are too many different fields of computer sciences and economics/econometrics involved to give a detailed description of every facet of data mining. For further reading we refer to the book by Kovalerchuk and Vityaev (2000) who give an excellent survey of the state-of-the-art data mining methods.

[23] Kovalerchuk and Vityaev (2000) p. 2

[24] The Self-Organizing Map algorithm belongs to the class of *unsupervised learning* algorithms.

1.6 Chapter Summary

This chapter was supposed to offer a short overview of the most important theoretical ideas that usually underlie models in finance. To this end concepts such as the EMH and the categorisation of traders into different classes were given. The Efficient Market Hypothesis is well known from many models in empirical finance and provides the foundation for the processing of information among traders. The heterogeneity of agents was used to (partly) explain non-linearities which can be found in financial time series and furthermore the EMH was only accepted in its weak form. From a practitioner's viewpoint this makes sense because it is challenging to agree with the assumption that all information is known to any market participant. Moreover, the difference between chartists and fundamentalists was elaborated and types of traders were differentiated by their investment horizon.

Subsequently, a short review of the concept *data mining* was given. Self-Organizing Maps are from their very nature part of typical data mining methods. Data mining summarises a broad range of techniques and methods applied to economic time series analysis and is therefore an important part in the framework of the thesis.

In the last section we turned to the question whether financial patterns contain information. Technical analysis is a common tool for traders and for many of them trends are an essential barometer of market opinion. For economists it is hard to accept the lack of confidence in fundamental data and therefore it is still subject to ongoing research why chartists seem to perform good on markets.

In the following chapter we will turn to a classification of Self-Organizing Maps into the family of computational intelligence and in particular into the group of Neural Networks. This will help to highlight the link between Kohonen's algorithm and conventional artificial learning algorithms.

Chapter 2

Towards Neural Networks

2.1 Introduction

Advances in computer technology are a blessing for economics research. Many of the answers economists are looking for are hidden in the data and might give solutions to some of the most fundamental questions a researcher is permanently up against. In recent years an impressive number of new sources of economic data in many different frequencies, accuracies, and quantities emerged. This gives researchers the opportunity to test theoretical ideas or to develop new models based on the availability of formerly unaccessible data. But the opening of these immense data streams entails new problems which have to be solved in order to efficiently make use out of them. New techniques for the handling of high-dimensional matrices have to be found and alternative approaches for the processing of complex data sets have to be developed. In this respect *Self-Organizing Maps* is one out of a large variety of promising new tools. The purpose of the following chapter is to give a profound insight into the technical details of the method and, moreover, an intuitive explanation how to interpret the generated results.

This chapter is organized as follows. First, we will turn to general Artificial Neural Network (ANN) theory in order to classify the *Self-Organizing Maps* into the context of literature on computational intelligence. Then we will have a closer look on the main ideas and technical considerations behind the SOM algorithm. In a first step an intuitive motivation is given for the use and the advantages of *Self-Organizing Maps* and later on a more technical description of the theoretical foundation and the necessary mathematics will be shown. This part of the chapter is also intended to give some important and useful background for a better understanding of this thesis. We will close the chapter with a short conclusion and an outlook of the future of SOM.

2.2 A General Overview of Computational Intelligence

Before we go on with the application of the Self-Organizing Map algorithm it is important to gain a better insight into the theory and the history of ANNs and their importance to modern research activities. Kohonen's Maps are a special form of the neural network family and therefore it is worthwhile to extend a bit on this issue.

The first question we have to deal with is the description of the embedding of SOMs into the family of Computational Intelligence (CI). In most textbooks CI is presented as a mixture of *Fuzzy Sets*, *ANN* and *Evolutionary Computation* or *Machine Learning*, respectively. As Cheng and Wang (2002) point out these are the three main pillars of CI. Figure 2.1 shows a possible family tree for the CI structure.[1] The chronological order of the development of the above mentioned three main parts of CI (Fuzzy Sets, ANN and Evolutionary Computation) is shown going from left to right.

The left branch can be interpreted as the first appearance of CI in literature. The *Fuzzy Logic* branch accounts for two tasks of the study of intelligent behaviour. On one hand *Fuzzy Logic* is an attempt to deal with that kind of uncertainty which is beyond the mathematical framework of probabilistic theory. On the other hand fuzzy sets are constructed to capture the significance of *natural language* when one is confronted with uncertainty. The roots of fuzzy sets can be traced back to psychology. Some psychologists find it reasonable to understand the ability of processing the amount of information we are permanently confronted with by means of fuzzy logic as a part of our thought process. This makes it possible to reduce the complexity of information and give way to communicate about it. The task is to classify verbal statements, such as "If the inflation and the unemployment rate for Japan stay moderate, the Bank of Japan will raise the interest rate above zero", into certain groups. Here the word *moderate* is not very exact. Fuzzy Logic offers the tool to deal with these inexact formulations by formalising them in terms of *membership functions*. One of the most important features of Fuzzy Logic is that classification is rather a *more or less* than an *all or nothing* decision. For instance, a prominent example for an If-Then-Rule of Fuzzy Logic is the *Mamdani style* rule.[2] The *Mamdani style* rule could be formulated as

`If` x `is high and` y `is low, then` z `is also low`

and the *Sugeno style* rule is

`If` x `is high and` y `is low, then` $z = f(x, y)$.

[1] Figure c.f. Cheng and Wang (2002) p. 4.

[2] There are basically two styles of rules. For the *Mamdani style* rule the antecedents and consequents of rules are fuzzy, whereas for the *Sugeno style* rule fuzziness applies only for the antecedents of a rule (see Cheng and Wang (2002) p. 11).

The history and the delopment of *Fuzzy Logic* is far beyond the scope of the work at hand. Fuzzy Logic is a wide and extensively studied field of research and application and has become a natural tool for economic model building. The short description given here does not suffice to give a complete explanation. For a deeper understanding of *Fuzzy Sets* or one of its sub-classes *Rough Sets* and *Grey Forecasting* we refer to one of the standard textbooks of CI.[3]

The second part of the family tree is more interesting for the purpose of this thesis. Starting from the ANN part we first have to divide this class into *supervised* and *unsupervised learning*. The latter one is characterized by the popular *Multilayer Perceptron Neural Network* (MPNN). Further down the *Support Vector Machine* (SVM) can be interpreted as a generalisation of the MPNN. The last subgroup of this branch of the CI family is given by the *Fourier Analysis* which represents a shift from *time-domain* to *frequency-domain* approaches. According to Cheng and Wang (2002) there is no specific reason to link the *Fourier Analysis* with the *Wavelets* at this specific point in the family tree. It only indicates that nowadays ANN are often used in connection with wavelets.

In contrary to the *frequency-domain* models such as *Fourier Analysis* the left branch of the CI family in figure 2.1 represents the shift to the so-called *feature-domain* models. SOM, k-Nearest Neighbours (KNN), *Decision Trees* (DTree), *Finite State Automata* (FSA), and *Local Polynomial Regression* belong to this class of CI. Economic models such as ARCH(m), GARCH(p,q), and ARIMA belong to the group of *time-domain* models. This sort of models tries to make use of the correlations between the lagged observations and error terms in order to extrapolate past values into the immediate future. The *feature-domain* approach is based on the idea that one can discover certain patterns or similar events in time series and make profit out of these features in order to improve forecasting performance. This is based on the belief that some sort of long memory is inherent in financial markets. Once a certain pattern appeared in a time series it is believed to be either representative for a whole class of patterns that are driven by the same market forces and, therefore, lead to the same outcome, or the pattern belongs to a family – even remotely – and, hence, is interpretable. We will seize this idea later in this thesis for model building.

Feature-based models try to find geometric or symbolic characteristics of the process in question and then act accordingly by taking advantage of these features. This can also be seen from another viewpoint: the modeling strategy shifts from a *global modeling* to *local modeling*. The world, as many economists have already emphasised, is far too complex to be modelled in one (linear) mathematical statement. It seems to be a promising path to fractionalise the complex relationships and find appropriate models for certain situations. When applied to (non-linear) time series this should lead to improved results.

[3] See e.g. Pedrycz and Gomide (1999).

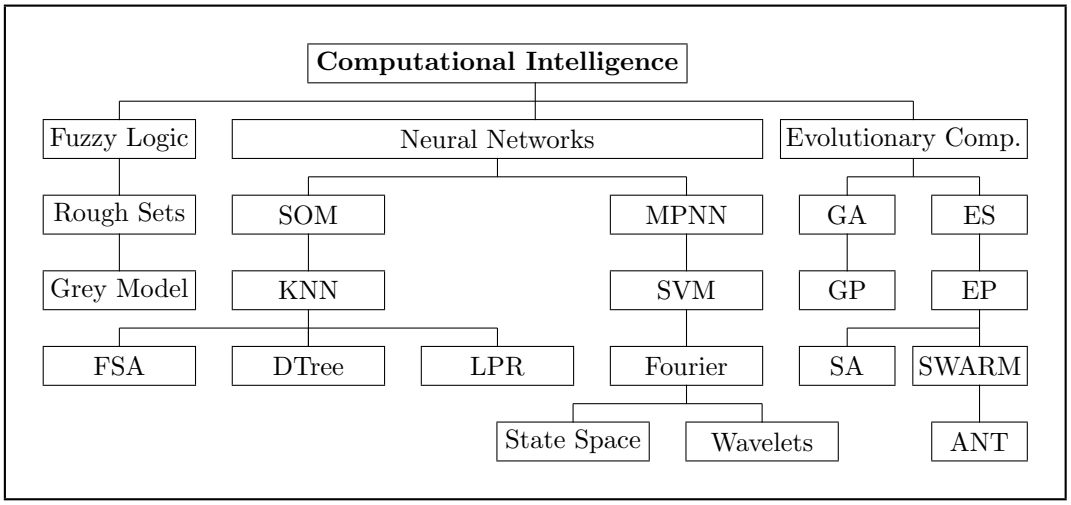


Figure 2.1: Family tree for computational intelligence

Figure 2.2 gives a better insight into the different strategies of creating models.[4] On the left branch we see the "traditional" approach which adopts the idea of correlations among observations and error terms. When used for forecasting this technique underlies the assumption that past realisations have influence on the future. This holds only for some processes, such as return series on financial markets where the volatility clustering phenomenon is observable. The GARCH(p,q) process provides a powerful tool to simulate the statistical properties of a return series.[5] A problem arises if e.g. prices are the data in question. In such a case the time series is usually not a stationary process and the past values are not correlated with the future realisations. That is the area of application where feature-domain models come into play. Feature-domain models make use of past events and the historical aftermath of these events. The system is able to group similar events and to filter information content from these categorisations. If one can tell what happened (*always* or *with a certain probability*) in the aftermath of a certain group of patterns this knowledge could be used to increase forecasting power of economic models. The right branch in figure 2.2 represents the group of feature-domain modelling. SOM, *Decision Trees* (DTree) and *k-Nearest-Neighbours* (KNN) are parts of this subgroup.

Features can either be symbolic or geometric. The SOM has the ability to deal with geometric features, whereas the *Finite State Automata* (FSA) provides a powerful tool to deal with symbolic features. SOMs as well as KNN and DTree classify individual objects into groups with similar characteristics by means of some distance metric (usually the Euclidean distance is chosen which has proven to perform very

[4]Picture in the style of Cheng and Wang (2002) p. 6.
[5]Bollerslev (1986)

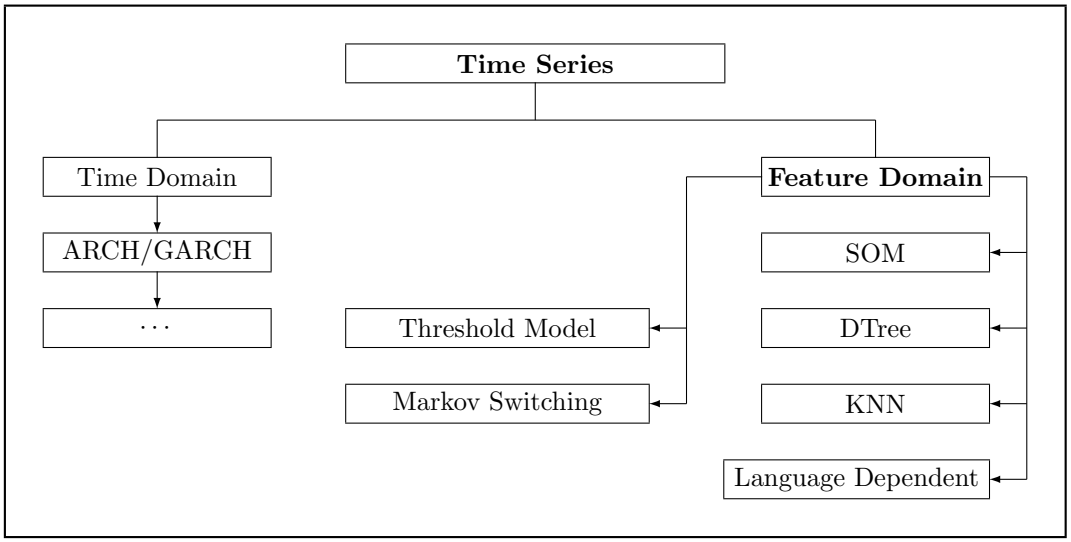


Figure 2.2: Time-domain and feature-domain approach

well in most of the applications; there are also non-euclidean approaches suggested in the literature[6]). In that sense SOM is only a tool for grouping events and, contrarily, KNN, LPR, and DTree are building simple (linear) models which involve simultaneous grouping.

SOM is not only a method for dimension reduction of high-dimensional data but represents also a very sophisticated technique for clustering of similar features, characteristics, and events. This makes the technique a family member of methods such as k-nearest neighbours, decision trees, local polynomial regression, and finite state automata.[7] The SOM represents the group of methods which do not deal with the time-domain approach, but show a shift to the feature-domain consideration.

Branching down the right side of figure 2.1 we find the section *Evolutionary Computation*, where *Genetic Algorithms* (GA), *Genetic Programming* (GP), *Evolutionary Strategies* (ES) and *Evolutionary Programming* (EP) are located. These subgroups of CI have experienced a lot of academic interest in recent years. The term *genetic* refers to *crossover* operations and the words *evolutionary* is strongly connected with *mutation* strategies. From this point of view it becomes clear that mixed forms of EP and GP are a natural development in the Evolutionary Computation theory. Both groups have proven to be powerful tools to model complex and non-linear structures in data. There is a large amount of literature available which will not be treated here in detail, but for a very nice review of this topic the

[6] See section 3.4.
[7] Cheng and Wang (2002) pp. 4-5.

Ph.D. thesis by Zschischang is worth reading.[8] A further difference between these four algorithms is the scheme of coding. Historically, GA uses binary coding, ES and EP commonly used real-valued coding. Genetic Programming – even if it is a generalisation of GA – adopts *parse-tree* coding, which formulates an algorithm that is different from the other three concepts in a dramatic manner. In parse-tree programming each node of the system is either a root, branch or leaf node. A root node is the source or starting point, respectively, of a system. Branch nodes refer to different alternatives/paths among which the programme can choose, and leaf node denotes the outcome of the programme at the very end of the branches. The last difference among the class of CI we are contemplating here is their fields of application.

Historically, ES was designed to solve complex numerical optimisation problems. EP and GA are mainly found when the subject of research is the simulation of intelligent behaviour. GP is designed to accomplish a more general purpose. When computer programmes are written by humans it is not necessarily clear in the beginning what the shape of the solution or its size will be. GP gives the computer programme the flexibility to grow in order to solve a specific problem. This is why GP has become a convenient tool for nonparametric modeling and at this point the connection between ANN and Evolutionary Computation may also be found. Both techniques offer a wide range of applications to solve nonparametric and nonlinear problems. Of course, these are two completely different approaches to deal with this sort of problems but they are definitely close relatives in the big tree of the CI family.

2.3 A Closer Look at Neural Networks

SOMs are a special class of Neural Networks. Before we turn to a detailed description of the algorithm we first have to clarify what Neural Networks really are, how they work, and what their econometric foundation is.

2.3.1 Linear Regression

In economics one is often confronted with forecasting or predicting certain target variables. Forecasting theory has become a large field of research and is equally important for each area of specialisation in economics. Usually, forecasting is accomplished by linking one output variable y to a set of observed variables x. The set of x can also include lagged observations of the target variable in question. In the most simple case this leads to the linear regression model

[8] Zschischang (2005) pp. 4-17

$$y_t = \sum_{k=1}^{k^\star} \beta_k x_{k,t} + \epsilon_t \tag{2.1}$$

$$\epsilon_t \sim N(0, \sigma^2). \tag{2.2}$$

It is an usual assumption that the random error term ϵ_t follows a normal distribution with mean zero and variance σ^2 (the variance is assumed to be constant). The set of parameters $\{\beta\}_{k=1}^{k^\star}$ is the part to be estimated and the target variable calculated by means of the estimated parameters $\{\hat{\beta}\}_{k=1}^{k^\star}$ is denoted by $\hat{y}$. The linear regression model tries to find the vector of $\{\hat{\beta}\}_{k=1}^{k^\star}$ that minimises the squared differences between the actual y and the values predicted by the model $\hat{y}$, i.e. the estimation process should lead to a parameter set $\{\hat{\beta}\}_{k=1}^{k^\star}$ that minimises the function Φ, where Φ is defined as the difference between the observed values y and the estimated values $\hat{y}$. Formally, the process can be written as

$$\min_{\hat{\beta}} \Phi = \sum_{t=1}^{T} \hat{\epsilon}_t^2 = \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 \tag{2.3}$$

$$\text{s.t.} \quad y_t = \sum \beta_k x_{k,t} + \epsilon_t \tag{2.4}$$

$$\hat{y}_t = \sum \hat{\beta}_k x_{k,t} \tag{2.5}$$

$$\epsilon_t \sim N(0, \sigma^2) \tag{2.6}$$

Prediction is usually accomplished by "freezing" the previously estimated parameters and use these estimates in order to extrapolate into the immediate future. The most common model for forecasting used in literature is the autoregressive model of the form

$$y_t = \sum_{k=1}^{k^\star} \beta_k y_{t-k} + \sum_{j=1}^{j^\star} \gamma_j x_{j,t} + \epsilon_t. \tag{2.7}$$

In this model $k^\star$ lags of the dependent variables y are included and $j^\star$ independent variables x (weighted with the parameters γ). This leads, of course, to a sum of $k^\star + j^\star$ parameters to be estimated. In this sense the degrees of freedom of the overall regression are reduced by augmenting the model with additional lagged dependent variables. By means of linear regression the linear model has a closed form solution which minimises the sum of squared errors (or: differences) between y and $\hat{y}$ and it is computationally very fast. On the other hand the linear model struggles when attempting to capture nonlinearities in time series. For instance, on financial markets we can observe long-term (up-) trends but in the short-run there are often huge swings in time series caused by e.g. bursting bubbles. This is where the linear model encounters its limits.

2.3.2 Nonlinear Models - GARCH

There are different nonlinear models which try to capture the true underlying process with a parametric assumption and a specific functional form. The most prominent example is the GARCH model, developed by Bollerslev[9] and Engle[10]. In this model the variance of the disturbance term ϵ directly affects the mean value of the target variable. The variance itself is modeled as a function of its own past value(s) and the past squared prediction error(s) ϵ_{t-1}^2. In the simplest form the GARCH model can be written down as

$$\begin{aligned} \sigma_t^2 &= \delta_0 + \delta_1 \sigma_{t-1}^2 + \delta_2 \epsilon_{t-1}^2 && (2.8) \\ \epsilon_t &\sim N(0, \sigma_t^2) && (2.9) \\ y_t &= \alpha + \beta \sigma_t + \epsilon_t. && (2.10) \end{aligned}$$

In the framework of the GARCH model y represents the rate of return of some asset, α is the expected rate of appreciation, and ϵ_t is the error term which is assumed to be normally distributed with zero mean and conditional (heteroskedastic) variance σ_t^2. The parameter β indicates the risk premium effect on the return, i.e. the investor demands a higher return on an asset when the risk increases ($\beta > 0$). The parameter vector δ defines the evolution of the conditional variance. The GARCH model is a *recursive stochastic* model. If initial conditions for σ_0^2 and ϵ_0^2 are given and the estimates for α, β, δ_0, δ_1 and δ_2 are known the asset return is completely determined as a function of a random shock, its own mean, and the risk premium effect (via the parameter β).

The distribution of the random shocks is assumed to be normal. Therefore, it is very easy to set up the likelihood function for this problem.

$$\begin{aligned} L_t &= \prod_{t=1}^{T} \sqrt{\frac{1}{2\pi\hat{\sigma}_t^2}} \exp\left(-\frac{(y_t - \hat{y}_t)^2}{2\hat{\sigma}_t^2}\right) && (2.11) \\ \hat{y}_t &= \hat{\alpha} + \hat{\beta}\hat{\sigma}_t && (2.12) \\ \hat{\epsilon}_t &= y_t - \hat{y}_t && (2.13) \\ \hat{\sigma}_t^2 &= \hat{\delta}_0 + \hat{\delta}_1 \hat{\sigma}_{t-1}^2 + \hat{\delta}_2 \hat{\epsilon}_{t-1}^2 && (2.14) \end{aligned}$$

The hats denote that the respective variables are estimates of the underlying parameters. Usually, the logarithm of the likelihood function is taken for numerical minimisation by means of some standard iterative procedure such as the *Berndt-Hall-Hall-Hausman* (BHHH) algorithm. In a formal way the function that is to be maximised is

[9] Bollerslev (1986) and Bollerslev (1987)
[10] Engle (1982)

$$\max_{\hat{\alpha},\hat{\beta},\hat{\delta}_0,\hat{\delta}_1,\hat{\delta}_2} \sum_{t=1}^{T} \ln(L_t) = \sum_{t=1}^{T} \left(-0.5 \ln(2\pi) - 0.5 \ln(\hat{\sigma}_t^2) - 0.5 \left(\frac{(y_t - \hat{y}_t)^2}{\hat{\sigma}_t^2} \right) \right)$$

$$\begin{aligned} \text{s.t.} \quad \hat{\sigma}_t^2 &> 0 \\ t &= 1, \cdots, T. \end{aligned} \tag{2.15}$$

It is often very difficult to achieve a minimisation of the likelihood function. This is one of the major drawbacks of the GARCH model. The GARCH model underlies the same difficulties as known from every other likelihood parameter estimation. When one is interested in the statistical significance of the parameters $\hat{\alpha}$, $\hat{\beta}$, $\hat{\delta}_0$, $\hat{\delta}_1$ and $\hat{\delta}_2$ it may be very difficult to obtain estimates for confidence intervals. Furthermore, the GARCH approach is very restrictive since it is limited to a well-defined distribution and a well-defined set of parameters. The estimation method does not always converge to interpretable parameters.

GARCH processes take the nonlinearities explicitly into account. This can be seen from the process that drives the conditional variance. It is determined by a nonlinear transformation of its own past values and additionally a nonlinear transformation of the past prediction errors are a driving force for the development of the variance. It is straightforward to use the variance as a determinant of the asset return since the risk plays an important role for the pricing on financial markets and, of course, for the forecasting of dynamics.

2.3.3 Polynomial Models

With polynomial approximation methods we try to face the drawbacks from the GARCH approach as described in subsection 2.3.2. These methods (which also include Neural Networks) are able to approximate an unknown nonlinear process with less-restrictive semi-parametric assumptions.[11] Here, the functional forms are given but the degree of the polynomial or the number of neurons are not. The number of parameters is not limited and for this reason it is not possible to give any straightforward interpretation of the estimates as it is usually the case, e.g. for GARCH models. This feature is the reason why this kind of models is called *semi-parametric*.

Let $g(\cdot)$ be some unknown but continuous function. The Taylor expansion around $x_{t-1}, x_{t-2}, ... = 0$ yields a discrete time *Volterra Series*[12]:

[11] McNelis (2005) p. 17
[12] Volterra (1959)

$$g(x_{t-1}, x_{t-2}, ...) = \sum_{i=1}^{\infty} a_i x_{t-i} + \sum_{i=1}^{\infty}\sum_{j=i}^{\infty} b_{ij} x_{t-i} x_{t-j} + \sum_{i=1}^{\infty}\sum_{j=i}^{\infty}\sum_{k=j}^{\infty} c_{ijk} x_{t-i} x_{t-j} x_{t-k} + \cdots . \quad (2.16)$$

The first part in equation (2.16) (the single summing) represents the standard linear moving average. The double-summing accounts for the lagged cross-products of two variables, and so on. In order to avoid counting a cross-product more than once the summation indexed by j start at i, the summations indexed by k start at j and so on. The function shown in equation 2.16 gives a weighted sum of polynomial functions of the past variables. The number of parameters grows exponentially with the degree of polynomial expansion. This is known as the *curse of dimensionality* in nonlinear approximation. An increasing number of parameters (and, thus, an improvement in the degree of accuracy) is accompanied by a decreasing degree of freedom of the statistical estimates. That is the trade-off a researcher has to face.[13]

2.3.4 Neural Networks

Neural Networks (NN) are a very efficient way to model nonlinear statistical processes. Generally speaking, NN relate a set of input variables to a set of one or more output variables. The difference between this method and the approaches described in the previous subsections is the one or more hidden layer which a NN uses in order to *squash* the variables. This transformation is accomplished by means of a special function (called *logistic* or *logsigmoid* functions).

Many different architectures of NN are known in the literature. In order to start with an understandable example we will first present the design of the *Feedforward Network*.

Feedforward Network

The simplest specification of a NN is the *Feedforward Network* which consists in our example of one hidden layer of two neurons, one output variable y and three input variables x_1, x_2 and x_3. Figure 2.3 shows schematically the architecture of such a network.

In this graphic the input variables, also known as *input-neurons*, and the output variable (*output-neurons*) are connected with the hidden layer through *synapses* (respresented by the arrows in between). The hidden layer attaches a certain weight to the input variables and produces an output. The hidden layer processes this operation in a parallel manner to improve prediction (*parallel processing*). This

[13] See e.g. Campbell et al. (1997) p. 471 or McNelis (2005) pp. 17-18.

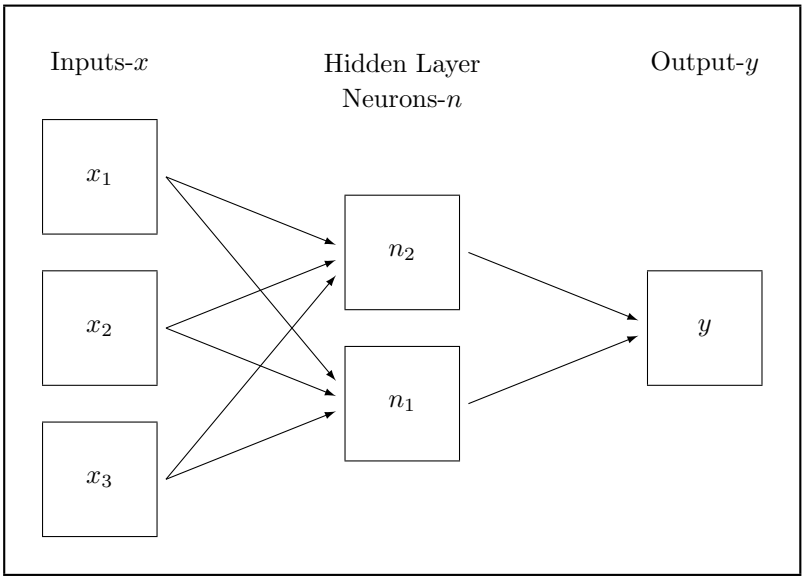


Figure 2.3: Feedforward neural network

technique is copied from the structure of the human brain, which engages a huge number of neurons in parallel processing. While the human brain develops, the number of synapses increase considerably. If a signal (or: *input neuron*) is presented to the NN the brain processes the data in a parallel fashion. For economic models and forecasting purposes it fortunately suffices to deal with a very limited number of neurons. In econometrics it is often important to include latent variables in the models. This class of variables represents usually the driving force in an economics process.

Decisions of investors on financial markets are not only influenced by lagged and current fundamental variables but also by a diversity of (subjective) factors, such as personal experiences, education, intelligence, and culture. All these factors are interconnected among themselves and the relationships are usually very hard to capture with standard modeling strategies. NN is a natural tool to forecast decisions of investors because it tries to clone the processing of signals in the human brain. Because of the complexity and the above mentioned latent variables influencing this decision process NN seem to perform well in this field.

Rustichini et al. (2002) point out that NN learning processes are based on two principles[14]:

- *Principle of Functional Segregation*: Basically, this principle formulates the idea that not the whole brain is needed for taking certain decisions or process-

[14]See Rustichini et al. (2002) p. 3.

ing all functions of the brain, respectively.

- *Principle of Functional Integration*: This principle states that different networks of regions (of the brain) are used for different functions. However, there are overlaps between the regions used in different networks.

In the same paper the authors argue that humans take decisions based on approximations. If this argument holds true, NN are probably the most natural way to approach forecasting problems.

Logsigmoid Activation The question now is how one can formalise such a NN and how exactly the above mentioned *Squasher Function* can be formulated. The input data are processed in two different ways when presented to the network. First, linear combinations of the data are formed and second, these linear transformations are squashed by means of a logsigmoid or logistic function. In many applications such a function appears in terms of an activation function in which small changes in the data do not activate the neuron if the system is in periods of extreme values. But when "threshold areas" are present small changes can make a difference and therefore can have an activating impact on the system. A system with a threshold feature (and the most basic NN) can be written down in the following way:

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i^\star} \omega_{k,i} x_{i,t} \tag{2.17}$$

$$N_{k,t} = \frac{1}{1 + e^{-n_{k,t}}} \tag{2.18}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^\star} \gamma_k N_{k,t}. \tag{2.19}$$

In this system $N_{k,t}$ is the logsigmoid activation function (see figure 2.4 (a) for a graphical representation). Each neuron $n_{k,t}$ enters that function as a linear transformation of $i^\star$ input variables x_t at time t (with $\omega_{k,i}$ as weights and $\omega_{k,0}$ as the constant term). The result of this first data processing are $k^\star$ neurons which are getting squashed by the logsigmoid function. This transformation in turn leads to $N_{k,t}$ neurons at time or observation t. Again, linear combination is used in order to forecast the output variable y_t. The vector $\{\gamma\}$ represents all weights and γ_0 is a constant term.

This special form of a feedforward neural network is also known as the *Multi-Layer-Perceptron* (MLP) network. In economics it is of special interest because threshold decisions are often made on financial markets. A trader's decision to sell or to hold an asset might not be affected by a company's announcement of less profit in one period if the company is still operating on a very high level of profit.

This scenario becomes a different story when the firm is already struggling to be profitable. In such a situation a much higher weight would be attached to the announcement simply because it had become more important. The trader definitely takes into account this sort of information and would act accordingly. Therefore, this very simple NN represents an extremely good alternative to traditional linear time series forecasting models. There is a whole bunch of modifications and more complicated forms available in the literature but the simple MLP we are presenting here is sufficient to understand the basic ideas behind the NN approach.

Other Activation Functions A natural adjustment of the algorithm is the reformulation of the activation function. The *tansig* or *tanh* function (also known as the *hyperbolic tangent function*) "scales" the linear combinations of the input into the interval $[-1;1]$. This stands in contrast to the logsigmoid function defined above (the logsigmoid function squashes the input combinations in a $[0;1]$ interval). In figure 2.4 (b) the *tansig* function is plotted. Mathematically, the system adopts the same form as in system (2.17) - (2.19) but the equation in (2.18) is substituted by

$$N_{k,t} = \frac{e^{n_{k,t}} - e^{-n_{k,t}}}{e^{n_{k,t}} + e^{-n_{k,t}}}. \tag{2.20}$$

In the case of a Gaussian activation function (2.18) assumes the following form (see figure 2.4 (a)):

$$N_{k,t} = \int_{-\infty}^{n_{k,t}} \sqrt{\frac{1}{2\pi}} e^{-\frac{1}{2}n_{k,t}^2}. \tag{2.21}$$

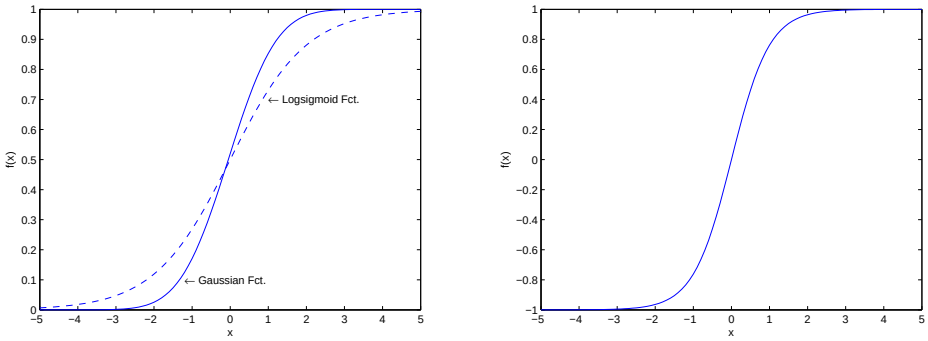
The Gaussian function has a more narrow distribution than the logsigmoid function and shows only little or no response when the input takes extreme values. (i.e. $(-\infty;-2]$ and $[2;\infty)$). Within the interval of critical values, such as $[-2;2]$ the Gaussian activation function has a steeper appearance than the logsigmoid function.

There have been used many other activation functions in the literature and it is impossible to give a rule of thumb which one is the best to employ. The researcher has to decide on that question under consideration of the subject of research. The design of the network and the choice of the respective activation function has to fit the subject matter. The three functions presented here give a nice showcase of the basic structure of neural networks.

Radial Basis Functions

Radial Basis Functions (RBF) can be distinguished from the MLP described above through several different characteristics. The two main differences are

- the existence of maximal **one** hidden layer and



(a) Logsigmoid and Gaussian Activation Function

(b) Tansig Activation Function

Figure 2.4: Graphical representation of different activation functions of neural networks

- the RBF computes the Euclidean Distance between the signal from the input vector and the center of a unit.[15]

As in the case described in the previous section the RBF also uses the Gaussian density function as the activation function but the structure of the NN is different to the one presented in the system (2.17), (2.21) and (2.19). It is possible that the input neuron is a linear combination of regressors, but it exists only one input signal or one set of coefficients of the input variables, respectively. Every neuron is a Gaussian transformation around $k^\star$ different means and the input signal is the same to every neuron. Therefore, the input signals have different centers for the normal distributions. Taking this set of differing Gaussian transformations and combining them in a linear way results in the output or the forecast, respectively. In a more formal notation the system can be written down as

$$\min_{\omega,\mu,\gamma} \sum_{t=0}^{T} (y_t - \hat{y}_t)^2 \tag{2.22}$$

$$n_t = \omega_0 + \sum_{i=1}^{i^\star} \omega_i x_{i,t} \tag{2.23}$$

$$\begin{aligned} R_{k,t} &= \phi(n_t; \mu_k) \\ &= \frac{1}{\sqrt{2\pi\sigma_{n-\mu_k}}} \exp\left(\frac{-[n_t - \mu_k]}{\sigma_{n-\mu_k}}\right)^2 \end{aligned} \tag{2.24}$$

$$\hat{y}_t = \gamma_0 + \sum_{k=1}^{k^\star} \gamma_k R_{k,t}. \tag{2.25}$$

[15] See Haykin (1994) for further differences between RBF and MLP.

As above in this system the input variables are represented by x, and n stands for their linear transformation under consideration of the weights $\{\omega\}_{i=1}^{i^\star}$. There are $k^\star$ different centers $\{\mu\}_{k=1}^{k^\star}$ giving $k^\star$ standard errors. Using this information the $k^\star$ different radial basis functions R_k are obtained. Once the radial basis functions are identified they are combined linearly in order to forecast y. To this end the R_k are weighted by a vector of parameters $\{\gamma\}_{k=1}^{k^\star}$. The optimisation procedure embraces the adjustment of the parameter sets $\{\omega\}_{i=1}^{i^\star}$ and $\{\gamma\}_{k=1}^{k^\star}$. Moreover, the centres of the transformation $\{\mu\}_{k=1}^{k^\star}$ are an argument for fitting the net.

There is quite a number of analogies between RBF and SOM. Both methods have their foundations in measuring the distance between a certain set of data in vectorial space and the center of a neural unit. In a recent paper Blayo et al. (2003) tried to combine these two methods in order to forecast the DAX30 Index. RBFs were used to build local models on the basis of data sets identified by SOM before.

Extensions

Multilayered Feedforward Neural Networks There exist a vast number of modifications of the basic networks in the literature and one can find networks which are exclusively designed for very special and unique problems. We will limit the attention to some of the most common extensions of the feedforward neural network. One of those is the *Multilayered Feedforward Network* (MLFNN). In this case not only one hidden layer is used but a number of layers are part of the network design. Figure 2.5 respresents the architecture of such a network.

The mathematical representation can be written down in the following way:

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i^\star} \omega_{k,i} x_{i,t} \tag{2.26}$$

$$N_{k,t} = \frac{1}{1 + e^{n_{k,t}}} \tag{2.27}$$

$$p_{l,t} = \rho_{l,0} + \sum_{k=1}^{k^\star} \rho_{l,k} N_{k,t} \tag{2.28}$$

$$P_{l,t} = \frac{1}{1 + e^{p_{l,t}}} \tag{2.29}$$

$$y_t = \gamma_0 + \sum_{l=1}^{l^\star} \gamma_l P_{l,t}. \tag{2.30}$$

The system includes $i^\star$ input variables, $k^\star$ neurons in the first hidden layer and $l^\star$ neurons in the second hidden layer. It is easy to see that this extension increases complexity and the number of parameters to be estimated is increased by the factor $(k^\star + 1)(l^\star - 1) + (l^\star + 1)$ when a second hidden layer is added. With an additional

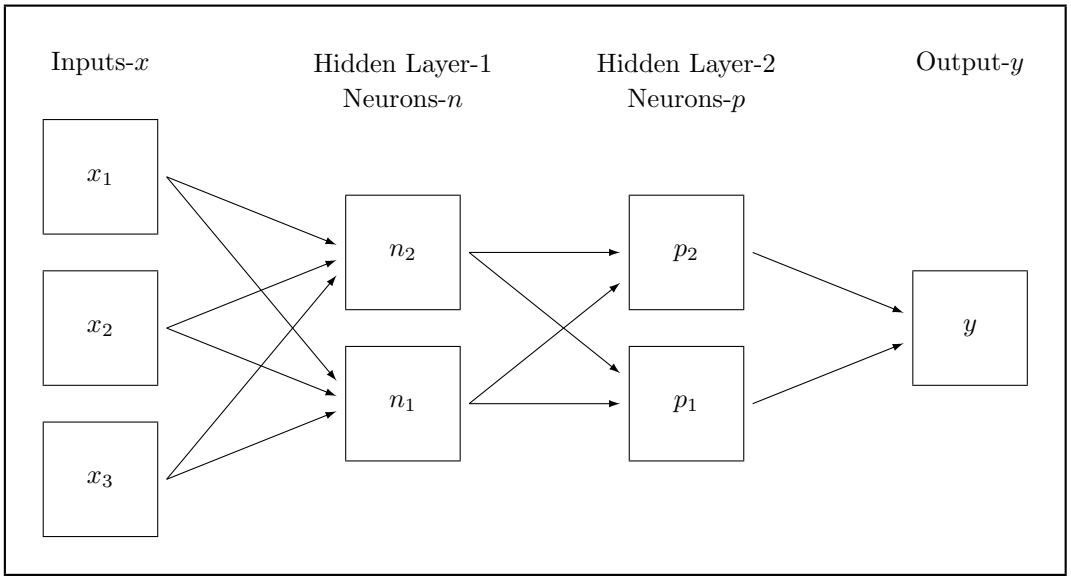


Figure 2.5: Multilayered feedforward neural network

layer several difficulties emerge. The training time of the net will increase and due to the increased number of parameters there is a looming danger of being caught up in a local extremum, rather than finding the global optimum. Even if a MLFFN has the ability to approximate general functional forms very well the usefulnes of them is widely discussed in the literature. Since the purpose of this section is only intended to give an overview of NN we refer to other sources.[16]

Recurrent Networks If one wants to model "memory" in networks – as it is the case for many financial applications – it is useful to allow for lagged values of the input variables. To this end the usual architecture is a recurrent network which is based on the widely used moving average (MA) process known from time series analysis. The MA process makes use of the lagged values of an unobservable disturbance term, i.e.

$$\hat{\epsilon}_{t-j} = y_{t-j} - \hat{y}_{t-j}. \tag{2.31}$$

Therefore, the dependent variable y depends on the vector of independent observations x and on the lagged errors from the j past periods. The system adopts the following form:

[16]For a deeper insight into this discussion please consult e.g. Dayhoff and DeLeo (2001).

$$y_t = \beta_0 + \sum_{i=1}^{i^\star} \beta_i x_{i,t} + \epsilon_t + \sum_{j=1}^{q} \nu_j \hat{\epsilon}_{t-j}. \tag{2.32}$$

The coefficient set ν_j is to be estimated recursively. In an iterative process the parameter vectors β_i and ν_j are adjusted until no or only small changes are achieved.[17]

This idea can be carried over to NN systems. In a NN framework the most influential design of such an implementation is the *Elman* recurrent network. This approach processes the lagged and current unobserved and unsquashed neurons in the hidden layer, i.e. the simple *Feedforward Network* as described in subsection 2.3.4 is augmented by lagged hidden-layer neurons which feed back into the computation of the current hidden-layer neurons. Mathematically, it turns out to be

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i^\star} \omega_{k,i} x_{i,t} + \sum_{k=1}^{k^\star} \phi_k n_{k,t-1} \tag{2.33}$$

$$N_{k,t} = \frac{1}{1 + e^{-n_{i,t}}} \tag{2.34}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^\star} \gamma_k N_{k,t}. \tag{2.35}$$

Again, x represents the observed independent variables and n are the neurons presented to the squashing function. The vectors ω and ϕ are the parameters which determine the weight for the entering elements.[18] The variable N gives the neuron after transformation and y is calculated as a linear combination of the N_k weighted by the coefficients γ_k (for $k = 1, ..., k^\star$). From this system it becomes obvious how the lagged values are connected with today's observations. They are combined in a linear fashion and then introduced to the logsigmoid squashing function. The lagged hidden layer neurons feed back into the hidden layer of the actual values. It is worth noting that the lagged neurons enter the system in their unsquashed state. The application of the logsigmoid squashing function takes place **after** the combination with the current neurons. This combination of the two types of values augments the system with "memory", which is assumed to play a very important role for the analysis of financial markets.

Neural Networks, but in particular the Elman network, have proven to be powerful especially for forecasting high-frequency financial data.[19] Due to the feed-back of the lagged neurons into the hidden layer it is very easy to model memory structure, in which the hidden layer itself changes over time. For instance, the feedforward network is regarded to be a *static* system because it does not allow for changes in

[17] McNelis (2005) p. 34

[18] Specifically, the parameters ϕ_k $(k = 1, \ldots, k^\star)$ belong to the lagged hidden-layer neurons.

[19] See e.g. Pérez-Rodríguez et al. (2005), Donaldson and Kamstra (1996) or Zhang et al. (1998).

the hidden layer over time. The Elman recurrent network is explicitly known to be a *dynamic* network. For both, the estimation of a moving average process as well as the estimation of an Elman recurrent network, it is necessary to use multistep procedures.

Networks with multiple outputs Of course, many other interesting extensions exist. Most of them are not covered here in detail because they are beyond the scope of the thesis but it is worthwhile mentioning at least networks with multiple outputs.

Up to now we have only taken into account networks with a single output (y). Some of the problems we are confronted with require networks which are able to generate multiple outputs. Obtaining an additional output requires a further ($k^\star+1$) parameters. The Elman network described in system (2.33) - (2.35) is augmented by a further equation, namely

$$y_{2,t} = \gamma_{2,0} + \sum_{k=1}^{k^\star} \gamma_{2,k} N_{k,t}. \tag{2.36}$$

Of course, equation (2.35) has to be adjusted so that y_t becomes $y_{1,t}$ and the parameters accordingly. The design of such a network makes sense when the two (or more) output variables all depend on the same set of input variables. For instance, if one wants to estimate the term-structure of interest rates with different maturities it is useful to utilise this kind of neural networks.

2.4 Data Preprocessing

When dealing with estimation problems in economics researchers often face non-stationarity in the data and, moreover, the scale of the data needs to be adjusted. The latter task is critically important for the correct use of non-linear estimation. Computationally and mathematically the reduction of the search space (the *scaling*) leads to decreasing CPU time and to less potential errors, such as the termination of the optimisation in a local minimum or maximum. As we will describe later on Self-Organizing Maps is a special kind of neural network. Therefore, many restrictions from the neural networks theory apply also for the SOM. That is why we have to take a closer look at data preprocessing here.

The preprocessing of data embraces mainly three basic procedures before the neural network can be applied (scaling, checking for stationarity, and elimination of seasonal influences). In the following we will describe what stands behind these three keywords.

Scaling Very high or very low values in a time series can lead to breaks in the calculation of neural networks due to computational over- or underflow problems. Furthermore, extreme values might bias the outcome of the neural network because they get a weight attached that puts simply too much importance on a single outlier. This problem is widely known as *overfitting*. For Self-Organizing Maps overfitting is not the crucial argument for scaling but it has to be mentioned here for completeness. A further problem arises when certain activation functions come into play (such as the logsigmoid or the tansig function[20]). In the logsigmoid case too extreme values are simply set to 0 or 1, respectively, which can correspond to a loss of information content in the input data. Therefore, scaling helps to preserve features of the data that are likely to be cut off if exposed to a neural network in its raw version.

Many different scaling methods are known from the literature. They can mainly be subdivided into two large groups: the linear and the nonlinear scaling methods. Linear scaling methods use the minimum and the maximum value of a time series and take these two values as borders for the succeeding scaling operations. For instance, the transformation is done by squeezing a variable x_k by the following function:

$$x^{\star}_{k,t} = \frac{x_{k,t} - \min(x_k)}{\max(x_k) - \min(x_k)}. \tag{2.37}$$

The result is a variable $x^{\star}_k$ which will be within the interval $[0, 1]$ and which, of course, will preserve the relations between the data points. If the interval $[-1, 1]$ is of interest a possible transformation function is

$$x^{\star\star}_{k,t} = 2 \cdot \frac{x_{k,t} - \min(x_k)}{\max(x_k) - \min(x_k)} - 1. \tag{2.38}$$

There are also many nonlinear methods for scaling data points but we will present only a simplistic one in order to give an idea of how scaling can be accomplished. One of the most simple nonlinear scaling methods is the one that consists of two steps. Firstly, a time series is standardised by applying

$$z = \frac{x - \bar{x}}{\sigma_x} \tag{2.39}$$

and then the logsigmoid transformation is used to squeeze the standardized time series z:

$$x^{\star} = \frac{1}{1 + \exp(-z)}. \tag{2.40}$$

Unfortunately, there is no "golden rule" for choosing the optimal scaling method. The researcher has to find the best method for the problem in question. It is

[20] See section 2.3.4 for a detailed explanation.

recommended to estimate the same problem several times with different scaling functions and then choose the function that leads to the best performance of the model.[21]

Checking for Stationarity Stationarity is a further important feature a time series has to exhibit if one wants to use the data for analysis. Statistical inference is based on the assumption of constance of the (at least) first and second moments, i.e. statistical inference is based on the constancy of mean, variance, and covariance over time.

Stationarity can be divided into its *weak* and its *strong* form. The strong form (or *strict* stationarity) is present if (and only if) the joint probability density function is invariant under time shifts:

$$f(x_{t_1}, ..., x_{t_k}) = f(x_{t_1+1}, ..., x_{t_k+l}) \quad \forall \; l, k \in \mathcal{N}, \tag{2.41}$$

where $f(\cdot)$ denotes the joint probability distribution function of a time series x at time t. It is almost impossible to check for strict stationarity in time series because one would have to revise every possible combination of time shifts l and different lengths of sequences k. For that reason the concept of weak stationarity was introduced.

Weak stationarity focuses on the first two moments of the distribution. A time series is said to be weakly stationary if (and only if) the following conditions are fullfilled:

$$\begin{aligned} \mathsf{E}[x_t] &= \mu < \infty \\ \mathsf{E}\left[(x_t - \mu)^2\right] &= \sigma^2 < \infty \\ \mathsf{COV}[x_t, x_{t-l}] &= \gamma_l < \infty. \end{aligned}$$

The mean of a time series μ, the variance σ^2 and the covariance γ are assumed to be constant over time and smaller than infinity. Moreover, the covariance γ should **only** depend on the lag l and not on the time t. Note that the appearance of phenomenons such as *volatility clustering* are still possible under the existence of weak stationarity.

The well-known *Dickey-Fuller-Test*[22] is the commonly used tool to check for stationarity of time series. Dickey and Fuller propose the following regression:

$$\Delta x_t = \rho x_{t-1} + \alpha_1 \Delta x_{t-1} + \alpha_2 \Delta x_{t-2} + \cdots + \alpha_k \Delta x_{t-k} + \epsilon_t, \tag{2.42}$$

[21] McNelis (2005) pp. 64–66
[22] Dickey and Fuller (1979)

where Δ stands for the lag-1 differences and $\rho, \alpha_1, ..., \alpha_k$ are the regression parameters which have to be estimated. The term ϵ is a random disturbance with zero mean and constant variance. Under the null hypothesis the test assumes $\rho = 0$ producing the expression

$$\Delta x_t = \alpha_1 \Delta x_{t-1} + \alpha_2 \Delta x_{t-2} + \cdots + \alpha_k \Delta x_{t-k} + \epsilon_t \tag{2.43}$$

or

$$x_t = x_{t-1} + \alpha_1 \Delta x_{t-1} + \alpha_2 \Delta x_{t-2} + \cdots + \alpha_k \Delta x_{t-k} + \epsilon_t. \tag{2.44}$$

In case the null is valid at any moment in time the variable x_t will be equal to its own past value x_{t-1} plus the effects of the sum of the difference terms $\sum_{i=1}^{k} \alpha_i \Delta x_{t-i}$. For that reason the mean at any point of time depends on the past values of x_t, i.e. the expected value of the series in the long-run becomes indeterminate. If the null holds we can speak of a *nonstationary* or a *unit-root* process. If the null does not hold and we estimate $\rho < 0$ (e.g. $\rho = -1$) the expression (2.43) collapses to the following term

$$x_t = \alpha_1 \Delta x_{t-1} + \alpha_2 \Delta x_{t-2} + \cdots + \alpha_k \Delta x_{t-k} + \epsilon_t. \tag{2.45}$$

If in the long-run $x_t = x_{t-1}$ holds, it follows that $\Delta x_{t-i} = 0$ for all $i = 0, \ldots, k$.[23] Therefore, the expected value of the series becomes exactly the expected value of the error term, i.e. $\mathsf{E}[x_t] = \mathsf{E}[\epsilon_t] = 0$.[24] In order to ensure stationarity it is important that the coefficient ρ is significantly less than zero. The test procedures proposed by Dickey and Fuller test the null ($\rho < 0$) by means of modified one-sided t-tests in a linear regression. An extension is the so-called *Augmented Dickey-Fuller Test* that allows a constant and trend terms in the regressions:

$$x_t = \mu + \beta t + \alpha_1 \Delta x_{t-1} + \alpha_2 \Delta x_{t-2} + \cdots + \alpha_k \Delta x_{t-k} + \epsilon_t, \tag{2.46}$$

where μ is the drift parameter and βt the trend component. Imposing $\mu \stackrel{!}{=} 0$ and $\beta \stackrel{!}{=} 0$ leads to the random-walk representation of the equation.[25]

Usually, it is easy to transform financial time series into stationary data. This is commonly done by first differencing the data sequence which normally removes nonstationarity from the data.[26]

[23] Note that this holds only in expectations, i.e. $\mathsf{E}[x_t] = x_{t-1}$ and $\mathsf{E}[\Delta x_{t-i}] = 0 \; \forall \; i = 0, \ldots, k$

[24] If $\rho \neq 0$ but falls in the interval [-1;0] there is *persistence* in the system. Assuming that in the long-run $\mathsf{E}[x_t] = x_{t-1}$ and hence $\mathsf{E}[\Delta x_{t-i}] = 0 \; \forall \; i = 0, \ldots, k$, the long-run formulation of equation (2.43) is then $x_t(1-\rho) = \epsilon_t$ which leads to the expectation value $\mathsf{E}[x_t] = \mathsf{E}[\epsilon]/(1-\rho)$.

[25] See Greene (2003) p. 643.

[26] Logarithmic first-differencing is an adequate method to deal with nonstationary financial time series. Let Z be a vector of random variables and the usual transformation will be $\Delta z_t = \ln(Z_t) - \ln(Z_{t-1})$ with $z_t \equiv \ln(Z_t)$.

Eliminating Seasonal Influences Another well-known obstacle for the analysis of time series is the existence of seasonal or calendar effects. One can encounter these problems in almost every economic model that deals with the behaviour of data over time, i.e. time series. The reasons that are causing these seasonal (ir-)regularities are manifold and depend on the question the researcher is working on. Surveying the consumption spending over the year it is not surprising to find spending peaks in November and December due to Christmas. But there are also more latent and more sophisticated roots of seasonal biases which have to be identified and eliminated before applying neural network methods. Otherwise, one would expose the analysis to the danger of overfitting, which was already commented in the paragraph "Scaling" (see above).

Depending on the structure of seasonal influences several techniques for removing them are known in the literature. For quarterly and monthly data the simplest way to achieve a filtering of calendar effects is the regression of a stationary random variable Δx on a dummy matrix Q, where $D = [D_2, D_3, D_4]$. The D-matrix represents the dummies for the second to the fourth quarter of a year. The Ds take on the value 1 if the data fall into the respective period (here *quarter*) and 0 else. In case of monthly data D would be formed of eleven entries $D_2, ..., D_{12}$ in order to account for monthly observations. The regression adopts simply the form

$$\Delta x = D'\beta + u. \tag{2.47}$$

Here, u (the residual) is everything which could not be explained by the seasonal dummies. Hence, we have extracted the "cleaned" data from the time series.

A further problem appears when the data frequency is higher and the time series in question is daily data or even of higher frequency nature. In such a case the structure of the week, trading days, and the influence of monthly effects have to be taken into account. The Gallant, Rossi, and Tauchen procedure accounts for these problems but will not be discussed here.[27]

2.5 Pattern Recognition

The term *pattern recognition* can be defined as the machine recognition of different patterns. It can also be described as "the act of taking in raw data and taking an action based on the category of the data"[28]. Usually, the term pattern recognition is connected with *supervised learning* but is also applicable with methods that adopt *unsupervised learning*, such as Self-Organizing Maps.

[27] Please see Gallant et al. (1992) for further reading on this topic.
[28] Theodoridis and Koutroumbas (2006)

Organisms tend to be very good at pattern recognition. The human eye is a very powerful device for the identification and processing of visual data and it is able to constantly classify sensory perceptions. Machine pattern recognition is a process that consists of three components:

- sensoric gathering of the observations to be classified,
- a mechanism that is able to extract relavant information from these observations and
- a scheme that effectively classifies the observations with regard to the extracted features.

The aim of the method is to group observations in such a way that the most important characteristics of the individual patterns are represented in one group, i.e. the most similar pieces of observations are clustered. When classifying data with supervised learning a learning strategy is developed based on the experience with a set of data which has already been classified (*training set*). On the other hand it is possible to let the system set up the classes itself by making use of statistical regularities of the patterns.

The fields of application of pattern recognition are manifold and embrace speech recognition, machined handwriting reading, recognition of human characteristics (e.g. the human face) and image processing, just to name a few. As it is easy to see all these applications try to close the gap between human and artificial sensory perception. Pattern recognition is currently studied in a multitude of scientific fields (e.g. psychology, physics, neuro-biology, ethology, and – of course – computer sciences).

2.6 Chapter Summary

This chapter provided an insight into the basics of computational intelligence and neural network theory. We started with a disambiguation that was followed by the classification of the specific members of the CI family. In the section that followed a brief overview of linear and non-linear models was presented and the most influential neural networks were shown. In that part of the chapter it became clear that neural networks are a natural tool for the analysis of time series and that the technique is suitable to approximate non-linear processes which often underlie financial time series.

We then touched the very important issue of data preprocessing. When dealing with any sort of neural network technique preprocessing is necessary in order to guarantee for an unbiased training of the network. Using raw data would lead

to wrong weighing of input factors and would, therefore, lead to an inadequate functional approximation.

This chapter was closed by defining the concept of "machine pattern recognition". Pattern recognition will be of central interest when we come to the second part of the thesis. Self-Organizing Maps offer a very convenient and flexible way to discover and identify structures (i.e. geometrical patterns) in the data that are not discoverable with the naked eye. We will make use of this feature and apply it to the analysis of financial data.

Chapter 3

Kohonen's Self-Organizing Maps Algorithm

3.1 Introduction

There has often been confusion when differentiating between *exploratory data analysis* and *data mining*. An exact definition of these two terms is still not available, but usually the expression *exploratory data analysis* is used to describe the whole process of the extraction of knowledge from data, whereas the term *data mining* is employed to describe a subgroup of the whole process, namely the discovery stage of the process.[1] Altogether, this can be subsumed under the generic term *knowledge discovery* in data. The goal is to discover information containing structures in the data which can be used to understand the underlying complexity of data driven processes. This can either be achieved by visual inspection or by numerical analysis (or a combination of these two). The latter relies on the combination of "classical" statistical analysis and data processing methods, such as neural networks or SOM, respectively.

In contrast, visual inspection of data structure requires a certain technique to make such an analysis feasible. When multidimensionality of data comes into play the methodological requirements become more sophisticated. Dimension reduction techniques have been subject to a wide field of research in data analysis and led to Principal Component Analysis (PCA), Cluster Analysis (CA), Discriminant Analysis (DA), and Factor Analysis (FA).[2] The Self-Organizing Map methodology, which is subject matter of this thesis, represents the group of CA algorithms that group similar patterns. Symbolic identification, in turn, can be used in order to reduce the dimensionality of the (multivariate) data in question (a detailed description of Self-Organizing Maps will be given in section 3.2).

The SOM methodology is an attractive tool for data mining because of four different reasons. Namely, the technique is

- a numerical method,

[1] Deboeck and Kohonen (1998) p. xxix

[2] For a comprehensive review of these methods please see Krishnaiah and Kanal (1982).

- a non-parametric method,
- a method that does not need a-priori assumptions about the distribution of data,
- a method which is able to detect unexpected features in the data because of its unsupervised learning character.

All of these features offer the user a large degree of flexibility. The aforementioned advantages of the technique have one virtue in common: the user does not have to make any ex-ante assumption on the structure of the data, the parameter restrictions, or any outliers or other oddities, respectively.

In recent years data mining from time series has treated mainly two problems. First, in order to extract informative structures from time series fully or partially similar patterns are looked for, and, secondly, another approach (which can be used in conjunction with the first one) would be to identify full or partial periodic patterns in the time series.[3]

Kohonen's Self-Organizing Maps are a sophisticated way to accomplish both, pattern recognition and data mining. The remainder of this chapter is devoted to describe the algorithm in great detail. After that we will turn to possible application of SOMs to economic problems.

3.2 The SOM Algorithm

The basic algorithm of Self-Organizing Maps was developed in 1982 by Teuvo Kohonen, a finish computer scientist who works mainly in the field of artificial intelligence. It is one – and probably the most famous – approach to a technique called *Unsupervised Neural Networks* which is often used in natural sciences for classifying, organizing, and visualizing large and multidimensional data sets.[4]

In particular financial markets researchers are confronted with data that make it difficult to filter informative structures. Moreover, during the last two decades the accessibility and availability of data has increased significantly. For economists this is a valuable development, but on the other hand there is a need for more sophisticated methods to deal with this growing amount of information. Kohonen's SOMs describe a special form of artificial neural networks which are trained by applying certain learning rules. On the basis of these rules the ANN adapts successively to the structure of the underlying training data set.

Methods using artificial intelligence enter more and more economic research, particularly in quantitative analysis of financial markets. *Neural Networks* and *Genetic*

[3] Cheng and Wang (2002) p. 263
[4] Deboeck and Kohonen (1998) p. xxviii

Algorithms have become manifest in the latter area of research. These techniques are able to simulate the behaviour of agents acting on financial markets due to the explicit integration of a learning element in the mathematical algorithms. Teuvo Kohonen's Self-Organizing Maps belong to the class of methodology which has attracted a lot of (academic and non-academic) attention in recent years. SOM form a special class of neural networks and the fields of appliance range from mathematics and natural sciences to virtual 3-D generation and medicine. The algorithm has come into the focus of attention for economists because it represents a powerful device for the analysis of time series and conduction data mining. The SOM methodology is of particular interest for marketing as well as for modelling of financial markets.

In a book by Deboeck and Kohonen (1998) a first collection of papers concerning the use of SOM models in finance was published. SOM models are of particular interest for the financial industry because they offer a solution to process complex and high-dimensional time series data in a convenient way. Having its roots in natural sciences Kohonen Maps have not yet entered economic research deeply, but some interesting innovative approaches have already been introduced. The economic literature in this field is still very scarce. Influential contributions were made by Blayo et al. (2003) who tried to find a probalistic inference for a prediction of a time series by means of classification by SOM. Furthermore, Cottrell et al. (1998) offer a solution for a period-forecasting problem by training a SOM on the daily energy consumption of a given population and using the identified patterns for out-of-sample prediction (in a 24 hour setting). A concrete example for the application of the method on financial markets can be found in Resta (2002). Principe et al. (1998) apply Self-Organizing Maps to predict chaotic deterministic time series, such as generated by the Lorenz equations. They gain a considerable success by using the pattern recognition method.

3.2.1 General Idea of SOM

Regression is a well-known concept from econometric analysis of economic time series (see section 2.3.1). Generally, we can state that regression of a dependent variable on some independent variable is the most famous method when dealing with empirical analysis. A (simple) mathematical relationship between a set of different variables is assumed *a priori* in terms of a function which is fitted to the distribution of the sample values of some input data by adjusting the respective coefficients.

As an example a linear relationship between the endogenous and exogenous variables is assumed (in matrix formulation):

$$y = \mathbf{X}\beta + u. \tag{3.1}$$

Here, y represents the vector of an endogenous variable and $\mathbf{X}$ the matrix with exogenous variables in its columns. An error term has to be added (here denoted by

u). Estimating this equation results in a vector β which defines a straight line (or a hyper-plane in the multivariate case) in through the observed data and minimises the sum of squares of the vertical distances between the observations and the regression line.[5] In that sense SOM can be interpreted as a special form of linear regression. This technique is known as *parametric regression* and finds a number of modifications and extensions in econometric literature.[6]

Due to the relatively simple mathematical coherence in equation (3.1) the adjustment of the regression line to the observations is achieved by the variation of parameter β. By using this technique many of the observed data points are mapped quite well, but the larger part is not represented by such a linear functional form.

Many different ideas exist to formulate the model in equation (3.1) in a more flexible way. As a prominent example polynomials can be mentioned which are normally able to map the respective data very well, but suffer in many cases from *overfitting*. This reduces tremendously the qualitative content of the model since ranges of dispersion do not have any informative character.

The SOM method is a *nonparametric* approach closely related to the basic idea of regression. The idea behind SOMs is the compression of multidimensional data sets in order to filter informative characteristics and to be able to give reasonable statements about causalities. SOMs are in particular useful because the data are grouped into a dimension-reduced *feature-map*, and phenomena such as pattern clustering are mapped in a convenient way. In contrary to other neural network methods the Self-Organizing Maps algorithm uses the so-called unsupervised learning. To make the difference between supervised and unsupervised learning clear in the following a brief overview of the two different methods and their underlying assumptions will be given.

- *Supervised Learning:* during the training phase the system strictly follows certain rules given by the respective user.
- *Unsupervised Learning:* the learning procedure is not subject to external guidelines or defaults, but rather exhibit a self-organising behaviour of the data.

SOM try to fit *ordered discrete reference vectors*[7] to the distribution of some input data (also given in vectorial form). These reference vectors can be interpreted as nodes which are mutually interconnected through some kind of hypothetical neural network. Due to these connections it is possible to introduce "elasticity" to the network which then gains the ability to approximate continuous functions. The strength

[5] This technique is commonly known as *normal least-square-method*; to be distinguished from the *total least-square-method* which minimizes the sum of squares of the orthogonal distances between the observations and the regression line.

[6] For a review see e.g. Hamilton (1994).

[7] Kohonen (2001) p. 85

of the connections is defined via a *neighbourhood function* (see section 3.2.4). The particular advantage of this method lies in its ability to map a multidimensional data set onto a two-dimensional map. Vector quantification and vector projection are combined (see section 3.2.2) and form an extremely powerful instrument in the fields of multidimensional data visualisation. This is a convenient feature when analyzing financial data because the complexity of economic time series data urge researchers to rely rather on visual inspection than on mathematical description.

The main advantage became clear when Kohonen developed this algorithm for *adaptive* learning: as opposed to the regression method SOM do not need an *a priori* assumption of a functional form. The form is given only by the data and certain restrictions imposed on the system. In a first step a map is spanned by vectors consisting of random numbers. Note that Kohonen's Self-Organizing Maps are robust against the initial vectors. The vectors establish a neural net in which every vector represents one specific node. Figure 3.1 shows two examples of a 2-dimensional grid of size 5×5. The architecture of such a net can be chosen to appear as rectangular (see figure 3.1 (a)) or hexagonial (figure 3.1 (b)). Each node is connected to either four or six of its direct neighbours through a communication link. The nodes are determined by *model-vectors* in the space $\Re^n$.

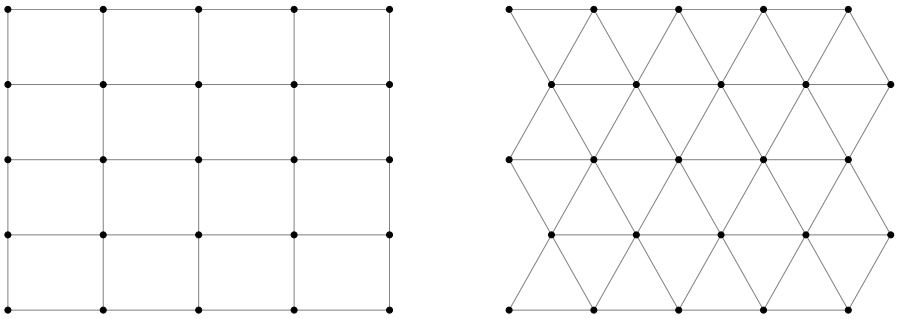
The advantage of using the hexagonial type of network construction is the higher degree of accuracy of the trained net. Through the six communication links the network is more flexible and has the ability to better replicate the actual shape of the data surface. The hexagonial architecture also leads to a faster learning (compared to the rectangular case) because of the higher number of communication links to the neighbours.

A further advantage of SOMs is based in the simplistic algorithm which is used to train the map. We will have a detailed look on this issue in the following subsections.

3.2.2 Vector Quantisation and Projection

Before we turn to the actual SOM algorithm it is necessary to have a closer look at matrix algebra which is essentially important for the forthcoming details. Apart from what is called *Matrix Algebra* we are mainly interested in a special field of methods, the *Vector Quantisation* (VQ). Generally speaking, this technique is applied in order to reduce the dimensionality of original data sets. This leads to a smaller, but still representative, set of data. The reduction is in particular necessary when one has to deal with high-dimensional and, therefore, computer-demanding data sets. Furthermore, multidimensionality makes a visual inspection of the data set difficult and it turns out to be almost impossible for the analyst to gather valuable information from these sets of data. VQ is a convenient tool to filter the essence out of the time series and at the same time it suppresses noise.

Since the SOM algorithm is mainly developed as an instrument for visual in-



(a) Rectangular Grid Architecture (b) Hexagonial Grid Architecture

Figure 3.1: Neural grid of size 5×5

spection of high-dimensional data sets it heavily relies on techniques to measure distances in $\Re^n$ spaces. These distances somehow have to be quantised, which can be accomplished by applying signal-approximation methods.[8] VQ is one of these methods and approximates the distribution of the input data vectors $x \in \Re^n$ by using so-called *codebook vectors* $m_i \in \Re^n$ (where $i = 1, 2, ..., k$). This approximation is usually done by means of the Euclidean Metric[9], i.e. by finding the codebook vector closest to the data input vector by applying

$$\|x - m_c\| = \min_i \{\|x - m_i\|\} \tag{3.2}$$

or

$$c = \arg\min_i \|x - m_i\|, \tag{3.3}$$

respectively. The codebook vectors are denoted by m_i $(i = 1, ..., N)$ and the data vector is x. The index c is chosen to denote the *winning codebook-vector*, i.e. the model vector that is closest to the data vector. The vector m_c is adjusted to the data vector. Depending on the VQ method used, either the winning codebook-vector is adjusted solely or its neighbourhood is affected as well.

There are different VQ methods known in literature which are shortly summarized in table 3.1. From this table we see the striking differences between the VQ methods. For instance, the k-means algorithm exclusively updates the "winning" cluster, whereas the SOM algorithm adjusts the winning neuron and its immediate neighbours. This results in a smoothing effect that leads to a better fit of the distribution of the data set in question.

[8] See e.g. Ossen (1993), Ozdemir and Erkmen (1993), Ritter (1993), or Sirosh and Miikkulainen (1994).

[9] For a rigorous treatment of Euclidean Metric see appendix A 3.1.

Algorithm	Notes
k-means	Only the closest cluster center of the sample is updated
Maximum Entropy	All cluster centers are updated according to their distance to the sample vector.
Neural Gas	All cluster centers are updated according to their ranking order in distance to the sample vector.
SOM	The cluster centers are updated according to their distance from the Best Matching Unit of the sample vector on the map grid.

Table 3.1: Vector quantisation algorithms

Let $p(x)$ be the probability density function of x then we can define the distortion measure E which can be minimized in order to find the optimal selection of m_i:

$$E = \int \|x - m_c\|^2 p(x) dx. \tag{3.4}$$

The integral is taken over the complete metric of x. Since c is a function of both, x and all the m_i, the calculation of the gradient of the distortion function E cannot be achieved easily. There may be discontinuous jumps in c with varying m_i. The reason for that is an m_i closest to x may be substituted by another m_i abruptly. Therefore, c jumps from one discrete value to another.

For a general formulation of $p(x)$ no closed-form solution for the codebook vectors m_i is available. However, it is possible to apply iterative approximation methods to resolve this problem. A derivation of this scheme is presented in the appendix to this chapter (see A 3.2).

For visualisation of multidimensional vectors a projection from the original high-dimensional input space to a (at most) 3-dimensional sample space has to be done. The distances between the original data or their topological order, respectively, should be preserved. There are several techniques available for projection as presented in extract in table 3.2. Here, Ξ_{ij} is a function (some metric) that determines the distance between the two vectors i and j. This distance corresponds to the vectors in the *input space*, whereas the function Ψ_{ij} refers to the distances in the *output space*. As can be seen from table 3.2 the SOM adopts a special functional form with a new index k. Defining k as the *prototype vector* (or: codebook vector) then Ξ_{ik} denotes the distance between the input data and the prototype. In exchange the function Ψ_{ck} can be interpreted as the distance between the *Best Matching Unit*

(BMU) of the data sample c and all other map units. A crucial difference between the SOM method and the other three methods presented here is the behaviour of the Ξ_{ij}: they are changing in the SOM case but remain fixed for all other cases.

The SOM is an appliance of a combination of VQ and projection. Each prototype vector can be interpreted as one unit on the neural grid which in turn forms the codebook. This codebook reflects the properties of the data, i.e. the prototype vectors show up as a topologically ordered lower dimensional map which is covered by the data distribution. In the SOM case VQ and projection are processed successively. First, the VQ takes place and in a second step the projection is accomplished. The two tasks are carried out interactively, which pronounces the main advantage of the SOM algorithm. It differs from a simple combination of VQ and projection algorithm where the projection is completely subordinate. A further difference between SOM and a serial combination of the two techniques is the regular shaped grid on which the projection takes place. A regular shaped grid makes it easier to compare different visualisations. The question how SOM can be used in order to facilitate visual inspection will be treated in section 3.3.

3.2.3 The Fundamental SOM Algorithm

The aim of this subsection is to give a basic insight into the ideas and properties of the fundamental SOM algorithm. In this part of the thesis Kohonen's algorithm is presented in its most basic version, but features already all the necessary and important elements of this quantitative tool. Let us first start with the formal framework.

Let $\mathbf{x} = (x_1, x_2)$ be a stochastic vector (or: input data vector) that consists of only two observed variables x_1 and x_2. Furthermore, let $\mathbf{m}_i = (m_{i1}, m_{i2})$ $(i = 1, 2, ..., N)$ be the N *model vectors* in 2-dimensional space. The task is to find the node $\mathbf{m}_i$ that is closest to the sample $\mathbf{x}$. This node $(\mathbf{m}^\star)$ is called the *winning neuron*. This procedure guarantees an adjustment of the neural grid to the observed data. In the two-dimensional space this means the fitting of a flexible curve towards the data.

The distance between $\mathbf{x}$ and $\mathbf{m}_i$ can be measured by means of vectorial differences. Usually, the differences are calculated in terms of the *Euclidean Distance* (see section 3.4 for other reasonable approaches). For an n-dimensional vector $\mathbf{x}$ the Euclidean norm is defined as

$$\| \mathbf{x} \| = \sqrt{(x_1)^2 + (x_2)^2 + ... + (x_n)^2}. \tag{3.5}$$

The norm $\| \mathbf{x} \|$ of a vector is the length of the respective vector. The *Euclidean Distance* between two vectors can therefore be written down as

Algorithm	Function	Notes
MDS	$\sum_i \sum_{i<j} (\Xi_{ij} - \Psi_{ij})^2$	Distances in the input space are approximated by distances in output space.
SP	$\sum_i \sum_{i<j} \frac{(\Xi_{ij} - \Psi_{ij})^2}{\Xi_{ij}}$	Local distances in input space are emphasized.
CCA	$\sum_i \sum_{i<j} (\Xi_{ij} - \Psi_{ij})^2 f(\Psi_{ij})$	Local distances in the output space are emphasised. $f(\Psi_{ij})$ is a function monotonically decreasing with output space distance.
SOM	$\sum_i \sum_k \Xi_{ik}^2 h_{ck}(\Psi_{ck})$	Input space distances Ξ_{ik} are measured between prototypes k and data vectors i rather than between all pairs of data vectors. Similarly input space distances are measured between map units (Ψ_{ck}).

The following abbreviations are used: MDS - Multi-Dimensional Scaling, SP - Sammon's Projection, CCA - Curvilinear Component Analysis, SOM - Self-Organizing Map

Table 3.2: Vector projection algorithms

$$\| \mathbf{x} - \mathbf{m} \| = \sqrt{(x_1 - m_1)^2 + (x_2 - m_2)^2 + ... + (x_n - m_n)^2}. \tag{3.6}$$

The *winning node* $\mathbf{m}_c$ can be described by the following condition[10]:

$$\| \mathbf{x} - \mathbf{m}_c \| = \min_i \{ \| \mathbf{x} - \mathbf{m}_i \| \} \tag{3.7}$$

or equivalently

$$c = \arg\min_i \{ \| \mathbf{x} - \mathbf{m}_i \| \}. \tag{3.8}$$

Due to the communication links between the specific neurons (nodes) not only the winning node $\mathbf{m}^\star$ but also its neighbouring neurons are adjusted by the proportion $\mathbf{x} - \mathbf{m}_i$. Technically, this can be done by employing a *neighbourhood function* (we will come back to the neighbourhood function in the next section).

Let n be the index of the n^{th} iteration step, then the *updating equation* for $\mathbf{m}_i$ can be described as

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + [\mathbf{x} - \mathbf{m}_i(n)], \tag{3.9}$$

The equation holds only for those nodes which lie in an *a priori* defined distance around the winning neuron. For all other nodes

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) \tag{3.10}$$

applies.

Two factors still remain to be defined. Since SOM is a learning-algorithm an essential element of the procedure is the presence of a *learning rate*. In the following section we take a closer look at the learning rate γ and a further not yet introduced building block: the neighbourhood-function (denoted by h_{ij}). The factors determine the radius and the impact of updating around the winning neuron, respectively. Therefore, these two functions are necessary to achieve a successive "smoothing" of the model vectors towards the observation data and must be endowed with certain properties which will be discussed in the following.

3.2.4 Neighbourhood Function and Learning Rate

Obviously, the neighbourhood function h_{ij} and the learning rate $\gamma(n)$ deserve a closer look because of their key roles in the framework of the SOM algorithm. Eventually, it is these two forces that drive the system to convergence and that determine the speed at which the neural grid is shaped.

[10] A similar equation was already shown in the previous subsection. Equation (3.2) emphasised the idea of VQ, whereas, here, the embedding of VQ into a SOM framework is shown.

The most important parameter to define is the neighbourhood function. Interpreting the communication links as the *strength of communication* between pairs of nodes, this relationship can be formulated as a function of nodes indices i and j ($h_{ij} = h(i, j)$). In SOM literature this function is usually written in Gaussian form:

$$h_{ij} = \exp\left\{\frac{-(i-j)^2}{2\sigma^2}\right\} \tag{3.11}$$

where $\sigma = \sigma(n)$ is an adequately chosen, monotonically decreasing function of iteration step n. The further away a node i is located from another node j, the smaller is the value h_{ij}. In figure 3.2 the function

$$\sigma(n) = \sigma_0 + \frac{\sigma_1 - \sigma_0}{N-1}(n-1) \tag{3.12}$$

is used for calculating σ, but any other suitable function might be employed as well. The neighbourhood function determines the radius around the neuron in question wherein neighboured nodes are affected by the updating of the winning neuron. Furthermore, the function defines the strength of the updating effect of the respective neighbours, i.e. direct neighbours experience a stronger adjustment than nodes on the border of the neighbourhood radius.

The neighbourhood function is typically connected with the learning rate in a multiplicative manner. The learning rate attains very high values in the beginning of the iteration process and decreases later on. The reason for this is straightforward: as SOM is a special sort of Artificial Intelligence (AI) the system has to incorporate a learning factor. Otherwise, the system would act in a myopic mode without being able to use the information obtained during the past iteration steps and convergence would last much longer (if reached).

The learning rate can adopt a linear or exponential form or it can be inversely proportional to the iteration step n. The outcome of the map does not depend on the selected form of $\gamma(n)$. Kohonen proposes

$$\gamma(n) = 0.9 \cdot \left(\frac{1-n}{1000}\right) \tag{3.13}$$

to be a good choice[11] but there are many other learning rates conceivable (for instance, in figure 3.2 we use a differing formulation for the learning of the process).

Since in the beginning a rough training of the neural network is necessary the learning rate $\gamma(n)$ should attain higher values in the beginning of the iteration process and monotonically decrease thereafter with each epoch of iteration ($0 < \gamma(n) \leq 1$).[12] Note that the model vectors are generated randomly so that the initial

[11] Kohonen (2001) p. 112.

[12] Usually, during the rough training phase the learning rate adopts the initial value $\gamma = 0.9$ and for the fine tuning phase $\gamma = 0.1$.

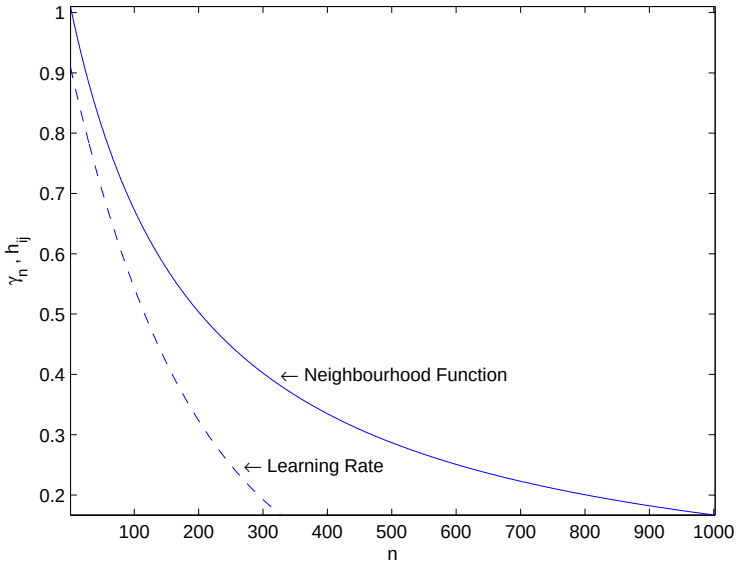


Figure 3.2: Graphical representation of a learning rate ($\gamma_n = \gamma_0(0.005/\gamma_0)^{(n-1)/N}$) and a neighbourhood function ($h_{j,i(x)} = \exp\{-d^2_{j,i(x)}/2\sigma^2(n)\}$, cf. equation (3.11)) decay plotted for $N = 1000$ iterations and $\gamma_0 = 0.9$ and $\sigma_0 = 1.00$, $\sigma_1 = 6.00$ and monotonically decreasing distance function $d = (i - j)$ (based on simulated random numbers), respectively.

vector differences will be very large. The rough training serves to speed up the training procedure of the SOM through an adjustment of the map that roughly covers the structure of the data. For this step it is not important to find a reliable and nearly exact approximation. This is accomplished during the fine tuning phase later on. Kohonen suggests to keep $\gamma(n)$ on a constant high level for approximately the first 1000 steps, and thereafter monotonically decreasing. The function can be linear, exponential, or inversely proportional to n.[13] After the initial training phase the learning rate should attain small values in order to accomplish the *ordering phase*. An example for a possible learning rate trajectory is shown in figure 3.2. This parameter is of special importance when one is dealing with very large maps because the convergence of the system depends crucially on the parameter.

The learning rate can hardly be optimised because of the multiplicative linkage with the neighbourhood function. One factor influences the other and vice versa, which makes it very complicated to give any certain statement about the effects. On the other hand optimisation is not necessary (for the purposes of this thesis) because the amount of data we are dealing with is easy to handle for Kohonen's Maps and does not require a long CPU time. Adjusting for the learning rate would lead to higher computational efficiency but does not change the qualitative results.

[13] Kohonen (2001) p. 88

For that reason we use the learning rate as proposed in the literature.

Both, $\gamma(n)$ and h_{ij}, are some monotonically decreasing functions of time or iteration steps, respectively. For SOM networks with less than a few hundred nodes at most the selection of the process parameter does not influence the system's behaviour very much.[14] However, the radius of the neighbourhood, which is influenced by the updating of one certain node, needs to be large enough, otherwise the map will be ordered locally only and would appear as various "mosaic-like" areas. It is reasonable to start with a very large radius and let it decrease with increasing iteration steps. An usual procedure in practice is to separate the convergence process in two phases: the *rough training phase* and the *fine-tuning phase*. In the latter phase only direct or at most very close neighbours of the updated neuron are influenced by the adjustment of the winner-node, whereas in the former case the updating of a winner-node has a strong impact on a wide surrounding of this node.

Figure 3.3 exemplarily shows the iterational reduction of the Euclidean length in a 4×4 map. The data taken for illustration is the natural logarithm of the German-US exchange rate from 11/23/90-11/25/05 and the log-differences of the exchange rate. The time series is "embedded" following the technique presented in chapter 4.3.[15] From the plot (a) in figure 3.3 we can see that the adjustment of the net not necessarily has to take place in a monotonical fashion. First, the Euclidean Distance between the model vectors and the data vectors increases and is then reduced later on (even though not monotonically either). Taking the same time series but using stationary log-differences of the data shows a completely different training behaviour. The Euclidean distance decreases gradually and monotonically, and converges to one.

Combining equation (3.9) with the neighbourhood function leads to the SOM algorithm

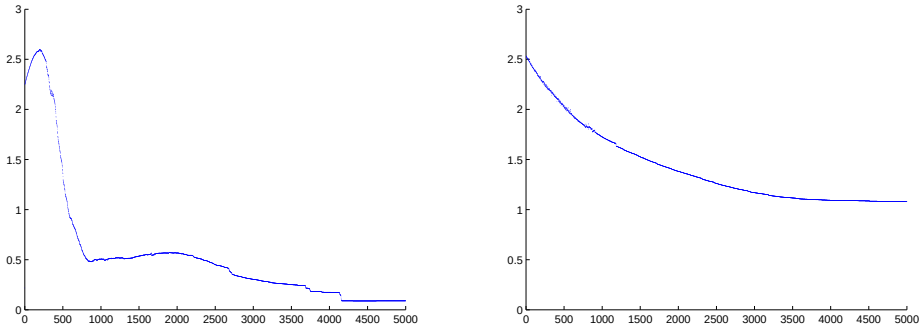
$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \gamma(n) h_{ci}(n)[\mathbf{x} - \mathbf{m}_i(n)]. \tag{3.14}$$

Technically, the SOM algorithm is separated into two phases. Initially the winning node $\mathbf{m}^\star$ is found by means of condition (3.7). Once $\mathbf{m}^\star$ is identified its immediate neighbours get adjusted using equation (3.9). These steps are repeated n times until the map is formed according to the topological structure of the underlying training data set.[16]

[14] Kohonen (2001) p. 111

[15] We abstain from presenting the details of the embedding at this point and refer to chapter 4.3 instead.

[16] Kohonen (1982)



(a) Log of Exchange Rate (b) Log of Exchange Rate Difference

Figure 3.3: Graphical representation of the development of the euclidean length over 5000 Iterations. The data used in (a) is the natural logarithm of the daily GER/US-$ exchange rate (p) between 11/23/1990 and 11/25/2005 and includes 3916 observations . The data in subgraph (b) is the same SOM applied to the same data as in (a) but transformed in accordance with the transformation $e_t = \ln(p_t) - \ln(p_{t-1})$.

3.2.5 Batch Training Algorithm

So far, the description of the algorithm has only concerned the so-called *sequential training algorithm*. This is an appropriate technique to use for the analysis of data sets if one is not concerned about the computation time. A faster learning algorithm proposed by Kohonen[17] is the *Batch-Training Algorithm*. An advantage of the batch algorithm is that it does not need a specification of the learning rate factor $\gamma(n)$. In the remainder of this subsection a short description of this alternative approach is given.[18]

Assuming the correctness of the assumption that the system converges to some ordered finite state it must hold that the expectation values of $\mathbf{m}_i(n+1) = \mathbf{m}_i(n)$ for $n \to \infty$. This result must also hold in the case of $h_{ci}(n) \neq 0$, i.e. in the stationary state the following equation holds true

$$\mathsf{E}\left[h_{ci}(\mathbf{x} - \mathbf{m}_i^\star)\right] = 0 \quad \forall\, i. \tag{3.15}$$

In the simplest case the factor h_{ci} is defined as an indicator function that adopts the value

$$h_{ci} = \begin{cases} 1 & \text{if} \quad i \in N_c \\ 0 & \text{otherwise} \end{cases}, \tag{3.16}$$

[17] See Kohonen (2001) pp. 138.

[18] At this point, a detailed illustration of the Batch Training Algorithm is not necessary because throughout this thesis the Sequential Training Method is used.

where N_c is the topological neighbourhood set of a certain cell on the network c. Now, let V_i be the set of sample vectors $\mathbf{x}$ for which $h_{ci} = 1$ (i.e. the vectors that are updating $\mathbf{m}_i$) and $n(V_i)$ is the total number of samples in V_i, then it follows

$$\mathbf{m}_i^\star = \frac{\sum_{V_i} \mathbf{x}}{n(V_i)}. \tag{3.17}$$

The interpretation of equation (3.17) is that for each $\mathbf{x} \in V_i$ the winner node c is necessarily a member of the neighbourhood set N_i of cell i. Provided all the sample observations $\mathbf{x}$ are known before the iterations, they can be used as a "batch" in the computation in the following manner. First, the model vectors $\mathbf{m}_i(1)$ are initialized as in the sequential training algorithm case. In the next step, for each model vector i the most similar sample data $\mathbf{x}$ are determined by means of the Euclidean distance. The new model vectors $\mathbf{m}_i(2)$ are formed by taking the mean over the set of sample vectors found in the second step. The procedure is then repeated several times until convergence of the system.

The main advantage of the batch algorithm is a higher degree of efficiency in computational memory allocation. The algorithm accounts for the whole set of neurons instead of updating only the nodes identified as winners in advance. From a computational point of view the calculation of the mean over the neighbourhood set N_i of node i is more efficient.[19]

3.2.6 Convergence of SOM

So far, the term *convergence* has been used frequently within this chapter. Different questions depend on the existence of the convergence property: if the input vectors are kept the same all the time, does the self-organising process converge and does the net formed by the SOM algorithm represent the topological order of the data well? Furthermore, it is an interesting question whether the outcome of the training will be the same if another distance measure than the Euclidean distance is used. For the Batch Training Algorithm (see previous subsection) convergence has been proved by Cheng[20], but as Kohonen remarks in his book "Self-Organizing Maps"

> [...] the self-ordering phenomena are actually very subtle and have strictly been proven only in the simplest cases.[21]

As Erwin et al. (1991) point out, the SOM has successfully been applied to many practical problems but it is still subject to ongoing research to prove convergence

[19] But not always applicable because problems might arise when the sample data $\mathbf{x}$ are unknown in advance or only fractionally available.

[20] The results by Cheng (1997) are very closely related to the original formulation of the SOM (see subsection 3.2.3).

[21] Kohonen (2001) p. 128

to optimal representations and to determine the speed of convergence to the final state.[22] Also, the question of a general theory of map formation remains unanswered so far. On the other hand, Kohonen argues that he never experienced any convergence problems when using SOMs.[23] Nevertheless, in a series of papers Erwin et al. present a proof of "convergence to stable, ordered configurations" which hold in the simple case of a topographic representation of the unit interval $[0;1]$ by a linear chain of neurons, i.e. the one-dimensional SOM case.[24] The construction of the proof is based on the previous work by Cottrell and Fort (1987) and Kohonen (1983) who derived a proof of convergence but only for cases where the neighbourhood function is a so-called *step neighbourhood function* of the form

$$h_{cj} = \begin{cases} 1 & \text{if} \quad \mid c-j \mid \leq 1 \\ 0 & \text{otherwise} \end{cases} . \tag{3.18}$$

We stick to our notation from the previous sections. Erwin et al. extend this approach to a more general setting which requires the neighbourhood function to be monotonically decreasing and positive-valued. This includes also the Gaussian kernel which is used throughout this thesis as a neighbourhood function.

In order to get a clearer idea of the basic concept of the proof we return to the formulation of the basic SOM algorithm as given in equation (3.14):

$$\mathbf{m}_j(n+1) = \mathbf{m}_j(n) + \gamma(n) h_{cj}(n)[\mathbf{x} - \mathbf{m}_i(n)], \tag{3.19}$$

with c as the winning neuron, $\gamma(n) > 0$ and $0 < h_{cj} < 1$ (h_{cj} as given in equation (3.11)). Erwin et al. (1991) assume in particular *convexity* of the neighbourhood function. Following Erwin et al. (1991) a neighbourhood function is said to be convex if $\mid j-q \mid \geq \mid j-c \mid, \mid c-q \mid \Rightarrow [h_0 + h_{jq}] \leq [h_{cj} + h_{cq}]$ for all $\mid j-q \mid, \mid j-c \mid, \mid c-q \mid$ within a certain interval I (and concave otherwise). Next, it is necessary to define a map of the input space of the unit interval $[0;1]$ which preserves the original relations of the distances in the input space (*ordering configuration*), i.e. $\mid c-j \mid < \mid c-q \mid \Longleftrightarrow \mid \mathbf{m}_c - \mathbf{m}_j \mid < \mid \mathbf{m}_c - \mathbf{m}_q \mid$ for all $c, j, q \in \{1, \ldots, N\}$, with N as the number of neurons. The so-called *optimal representation* is defined as the *ordered* stationary state of equation (3.19). Given a certain pattern ν the update process (3.19) can be reformulated as

$$[\mathbf{m}_j(n+1) - \nu(n)] = \underbrace{[1 - \gamma(n) h_{cj}]}_{\xi} [\mathbf{m}_j(n) - \nu(n)]. \tag{3.20}$$

The important part of equation (3.20) is the factor ξ. If the ordered configuration is fulfilled and under the condition that the neighbourhood function is monotonically

[22] Erwin et al. (1991)
[23] Kohonen (2001) p. 139
[24] Erwin et al. (1991), Erwin et al. (1992a) and Erwin et al. (1992b)

decreasing with $\mid c - j \mid$, it follows that $0 < \xi < 1$ and $\frac{\partial \xi}{\partial([\mathbf{m}_j - \nu(n)])} < 0$. Hence, it is not possible that the application of equation (3.20) changes the sequence of model vectors $\mathbf{m}$ in an ordered configuration.

Kohonen[25] and Cottrel and Fort[26] showed for step neighbourhood functions of the form as given in equation (3.18) that the SOM algorithm arranges a randomly generated initial set of model vectors into an ordered configuration for $n \to \infty$. Erwin et al. extended this proof for arbitrary monotonically decreasing, positive-valued neighbourhood functions. The proof of convergence is cumbersome and beyond the scope of the work at hand. For a detailed mathematical derivation we refer to the work of Erwin et al., especially the appendix to the paper Erwin et al. (1992a).

One way to determine the accuracy of the ordering process is the *distortion measure*. The optimal selection of the $\mathbf{m}_i$ minimises the average expected square of the *quantisation error*[27]

$$E = \int \|x - \mathbf{m}_c\|^2 p(x) dx. \tag{3.21}$$

It is not easy to form the gradient E with respect to any of the $\mathbf{m}_i$ in this case because the subscript c is a function of a pattern x and **all** the $\mathbf{m}_i$'s.[28] Due to the iterative process behind the optimisation $\mathbf{m}_i$ may jump throughout the process and hence index c makes discontinuous jumps (remember, the distances between the input vectors and the model vectors have to be reassessed every iteration step). However, it is not difficult to obtain the distortion measure from computer-based calculations. Unfortunately, the quantisation error cannot be used as an absolute measure of goodness. Each map design will generate different values which are only comparable within the same setting with different initial model vectors, but not for comparison of different map sizes or to determine the optimal number of nodes on the net.

In the next section we will turn to another extremely versatile feature of the SOM method: the graphical representation of multidimensional data sets.

3.3 Using SOM as a graphical tool

The SOM technique is a very powerful tool for the graphical representation of high dimensional data sets. It is often difficult to deal with large databases and in many cases visual inspection is only possible if the dimension is reduced and only the most important or the most informative components are considered, respectively.

[25] Kohonen (1983)
[26] Cottrell and Fort (1987)
[27] Cf. equation (3.4).
[28] Kohonen (2001) p. 59

It is the *number of visual dimensions* which determines how much information can be inserted into one visualisation in an efficient manner. As shown in section 3.2.2 there are a couple of projection methods available but the task is to determine which one is best to use for SOM and – even more important – which one is the most efficient one. The efficiency issue has to be evaluated first since it is, obviously, of crucial impact on the outcome of the visualisation. In case a too parsimonious method is chosen, much valuable information will probably be concealed just because of technical limitations; on the other hand the contrary might be encountered by using a very extensive method. This would not serve either since a user would have to accept a loss of compact visualisation, which is basically one of the main goals of this sort of analysis.

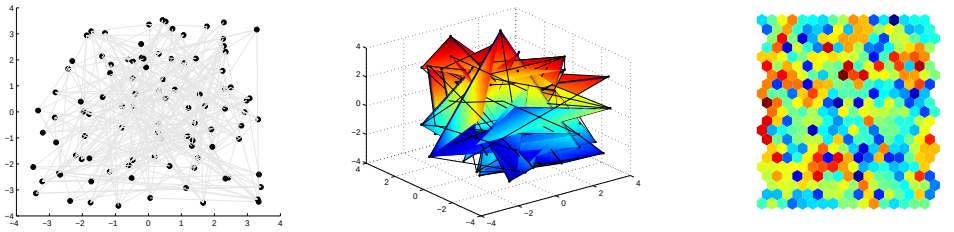
It was already mentioned in section 3.2.2 that the SOM algorithm is a kind of vector quantisation. Generally, vector quantisation reduces the dimensionality of data without suppressing its main characteristics. This is a very convenient feature of the technique because in many practical applications we are rather interested in certain informative features but in the very detailed structure of the data. An additional side-effect of vector quantisation is the reduction of noise in the data. The next subsections will give an overview on how the visualisations by means of SOM can be done and which sort of problems are encountered.

3.3.1 Projection

Projection of data is accomplished in order to visualise high-dimensional data sets. To this end, the original input space is projected to an output space that has at most three dimensions. The task of the projection is to find such a space that is capable to roughly preserve the original structure of the data, i.e. the distances between the original vectors and/or the topological ordering. Vector quantisation, however, is the genus for reducing a large data set to a smaller one. In this regard, SOM is a combination of these two techniques.

Visual dimensions typically include position, size, colour or texture, shape and motion. The last two dimensions are very hard to capture in a normal graph and their use is rather problematic. Despite the very limited number of visual dimensions one wants to compress as many useful information as possible into the graphics in question. Therefore, one is forced to use *multiple visualisations* instead of just one. As a rule of thumb the observer should be able to see the *link* between objects. But how can such a link be established? This problem includes features such as *where* a specific object is located in the different visualisations and *which* part of the respective visualisation corresponds to the same object. In the SOM case the links are usually created by marking identical or similar positions of the specific object.

Regarding the projection in terms of SOM we are facing a crucial problem: the



(a) 2D projection (b) 3D projection (c) U-Mat colorcode

Figure 3.4: Possible forms of prototype vector projection

projection is restricted to the junctions of the map grid. It is one of the characterising features of SOM that all the prototype vectors are interconnected. These links impose certain restrictions on the shape of the net. A crude adjustment of the net to the real data is only one of the possible risks. In the following three different projection methods are presented.

2D Projection

In figure 3.4 (a) the idea of the 2-dimensional projection is shown. Each black dot is defined through a vector in the plane space. These dots ("neurons") are initially generated randomly and the gray lines between the dots indicate the connection of the spanned neural net. We see a Self-Organizing Map that has already been trained and the black dots indicate the "codebook vectors" which, in turn, represent the center of a cluster of similar data vectors. The codebook vectors form a data cloud in the input space. On the other hand, the input space is defined through a number of vectors given by the observation data in question. The SOM algorithm is able to order the net in several iteration steps while it preserves the topological order of the input data structure. It does not matter where the neurons are located in the beginning of the training. Of course, the 2-dimensional illustration corresponds to the simplest case in terms of the graphical representation. Eventually, the task is to identify clusters in the data and to find outliers, respectively.

Even if Self-Organizing Maps are usually formed rectangularly, in most of the known applications it is not possible to find a reasonable interpretation of the x- and y-axes. The axes serve as "auxiliary variables" that span a co-ordinate system in which the SOM net arranged according to its codebook vectors. This is one of the major drawbacks of the method and has to be taken into account when interpreting the SOM-based visualisations.

3D Projection

The 3-dimensional projection is just the logical extension of the former case. Again, a net formed by randomly generated vectors generates a structure in the $\Re^3$-space. Note, that the information content of the 3D projection is higher since we can rely on an additional dimension. The third dimension is also the limit of the visualisation possibilities of the SOM and the perception capabilities of the human brain. For the projection into the 3D space the colorcode from graphic 3.4 (c) is taken.

Distance Matrices

Usually, the distances between codebook vectors and sample data vectors are measured in terms of the Euclidean distance. This is a very useful information because distances tell us a lot about the structure of the underlying process and are necessary in order to identify clusters. These graphics are known as *U-Mat* and they show the distance of every node to its direct neighbours. In the case of a hexagonial neural net each neuron has six connections. The colour used in this figure gives the distance following the known thermo-dynamic colour maps in engineering disciplines. The dimension of the U-Mat is larger than the map dimension because the links are presented and not the respective neurons themselves.

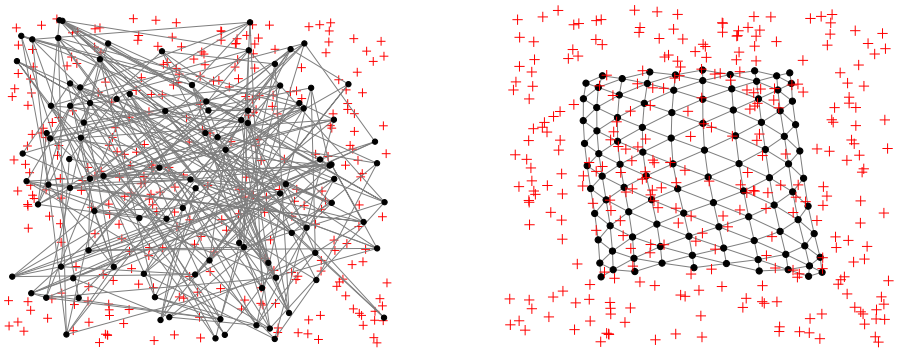
By using this visualisation it is now possible to make a more detailed statement on the structure of the data. The distances between the nodes can be used to interprete the clustering structure of the map and to identify outliers. This is an easy task since the color-coding provides a very convenient scheme for the human eye to process this relatively complex information quickly. This feature can be regarded as one of the major advantages of the SOM.

3.3.2 Visualisation

In figure 3.5 the procedure of the SOM algorithm is visualised. As an example we retain the grid dimension 4×4 (as in graphic 3.1 but with a lower number of nodes) and 300 uniformly distributed random numbers exemplify the observation data.

On the left hand side the neural grid given by the initial model vectors is shown. As mentioned above the SOM method is insensitive in respect to the initial model vectors, but reasonable selection of starting values can shorten the computing time considerably. In the graphic the observation data are plotted as crosses whereas the model vectors are represented by black dots. The initial neural grid was created randomly which explains the completely disorganized state of the net.

After employment of the SOM algorithm the right hand side graphic develops. The SOM grid adapts to the observation data and organises itself by adjusting successively to the data (*self-organising*). Since the observation data are uniformly distributed, the final form of the grid makes perfect sense in terms of visualisation.



(a) SOM net before training (b) SOM net after training

+ = input data,
• = codebook vectors generated by projection.

Figure 3.5: SOM before and after training phase applied to a data set of 300 uniformly distributed random numbers

It has then settled down in a state where all model vectors maintain approximately equidistant Euclidean lengths to their respective neighbours. In a more realistic scenario the neural grid would adapt to the topological order of the (multidimensional) data.

Even if the regularly shaped projection grid of the SOM makes the comparison between different visualisations easy such a rigid grid confronts the user with some problems. Firstly, it is a very difficult task to find a clear interpretation of the axis of the SOM map. Secondly, the VQ process is guided by the grid which means that the shape of the grid can be distorted. This is straightforward because if the VQ has discovered two clusters data points in between these two clusters attract interpolating neural units between these two clusters. As a consequence the visualisation might produce an incorrect data shape. A third problem is encountered when a closer look is taken at the projection implemented by the SOM alone. With the raw map grid alone it is very hard to draw any conclusions about the global shape of the data.

So far, the visualisations described here are interesting and they might be helpful in many cases, but they are not able to disclose relevant information. The arising question is what the distance between the model vector and attributive sample data vectors is? Model vectors can be interpreted as the mean value curves of the respective group of sample data vectors that belong to a certain node. Clearly, not every sample data vector has the same distance to the model vector in one group and, furthermore, not every node unifies the same number of sample data vectors

as other neurons do. These two problems have to be solved by using alternative graphical representations: the *SOM-Hit-Histogram* and the *Similarity Coding*.

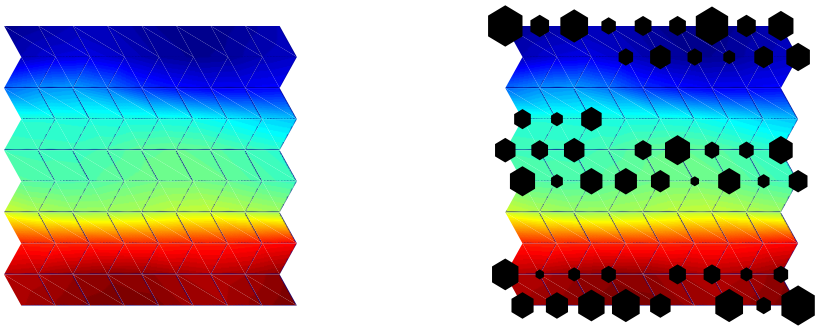
Hit-Histograms

The Hit-Histogram serves to give a visual impression about the real structure of the data set. A trained neuron is not necessarily as important as its neighbours probably are. In contrast, the most likely case is an imbalanced data distribution over the trained net and some clusters produced by the SOM method will unify more data and others less. This is an additional feature of the self-organising process which has to be handled carefully, but it also opens the opportunity for the extraction of new information. The knowledge of the distribution of the data vectors can be used in terms of a probabilistic way which can be applied to forecasting problems.

Figure 3.6 serves as an example. A data set consisting of three different sub-matrices is used to train the net. Each sub-matrix contains three columns and includes 100 rows. The data were generated by drawing a matrix of the dimension $[100 \times 3]$ from uniformly distributed random variables. The second and third sub-matrices are modifications of the first one and were generated by simply adding 4 and 9, respectively, to each element of the original matrix. The construction of the data set will lead to an unambiguous outcome of the trained net and will help to differentiate between the three data sets on the map later on.

Hence, the artificial data represent a sample of 300 observations (rows) with three characteristics/components (columns) each. The input data matrix has the dimension $[300 \times 3]$. A SOM is trained and after the iterations the distances between the neurons are calculated and represented in the U-Matrix shown in figure 3.6 (a). One can easily see the formation of three clusters corresponding to the different data sets fed into the training process (the net size is $[10 \times 10]$, i.e. 100 neurons). The borders between the three different clusters are obvious: the greater the distances between the neurons are the more the colour changes from blue to red (according to standard heat-diagrams). Darker colours refer to smaller distances between the neighbouring nodes, i.e. it is possible to identify the clusters in figure 3.6 in the upper, the middle, and the lower third of the representation. The blue part of the graphic belongs to the original data set, the green part represents the matrix to which we added 4, and the red third of the map refers to the third data set (original matrix plus 9). The question we discuss in this subsection is how to interpret the relative importance of each neuron.

In figure 3.6 (b) the "hits" on the neurons are added. The bigger the black dot lying on a neuron the more input data vectors are unified on the respective node. Additionally, we can observe from the graphic in 3.6 (b) that the relative importance of nodes becomes smaller the closer a node lies to the cluster border identified by the SOM. Obviously, this has to be the case in a well ordered map when the data



(a) U-Matrix (b) U-Matrix with histogram markers

Figure 3.6: U-matrix of 100 uniformly distributed random variables with and without the respective hit histogram markers

has inherently clear discriminatory power.

Similarity Coding

The graphic considered in the previous subsection has a further very important and useful feature. The colour code used to generate the two pictures in figure 3.6 are not chosen randomly, but correspond to the degree of similarity between a neuron and its neighbours. The colours have to be read as indicators for the distances between the neurons and their neighours. In terms of the Euclidean distance measure the darker the color gets the smaller is the distance between a neuron and its immediate neighbours. This representation is known from other disciplines, such as geography. The SOM method offers the possibility to present high-dimensional data in a very convenient way to the human eye and opens therefore opportunities to extract relevant characteristics of the data by visual inspection. The graphic can be interpreted in terms of a topological surface and uses the colour scheme known from sciences such as geography. It does show in a two-dimensional way the "ground properties" of the data and exhibits, therefore, relevant information on its structure. Of course, this is not meant to be an exhaustive analysis of complex data sets but gives, in many cases, a first impression of what one has to look for.

Other Display Formats

So far we have only considered a neural network design that is based on the idea of sheet-modeled net. But there are other geometric forms conceivable, such as a cylinder or a toroid. Figures 3.7 (a) and (b) graphically represent this concept. The form shown in these two figures are not necessarily advantageous for every field of

application, but it might lead to a better visualisation of the results.

The researcher has to decide under consideration of the problem at hand which map structure fits the problem best. There is no rule of thumb or any theoretical prior that unerringly points into the direction of a certain map structure. In case of doubt the best way to determine the adequate structure is to run experiments with toroids, sheets, and cylinders and choose the structure with the highest quantitative accuracy (in terms of smallest sum of Euclidean distances over the entire grid).

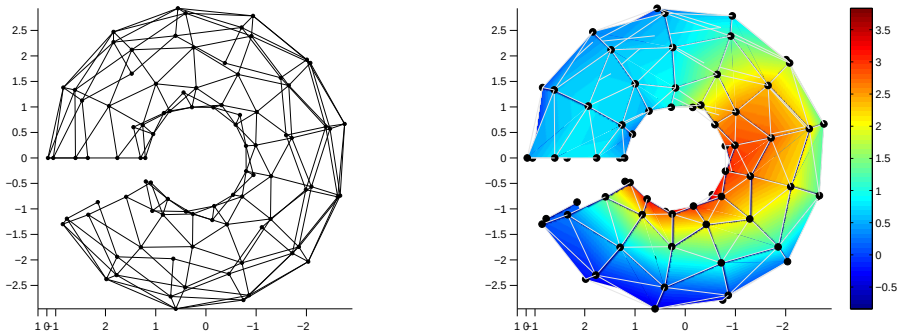
3.4 Extensions and Alternative Approaches

Self-Organizing Maps is a relatively new approach to deal with (multivariate) data analysis and visualisation. There is still space for further developments and refinements of the method and the algorithm itself. So far, we have only considered SOM working in Euclidean spaces. In the remainder of the thesis we will stick to this technical assumption because it is very convenient for computational and programming reasons. Moreover, in terms of the applications shown here it suffices to stay with the Euclidean metric.

However, this section shall be devoted to give an overview of most recent research in the field of neuro-informatics and, especially, modifications of Self-Organizing Maps.

Ritter (1999) suggests to design Self-Organizing Maps in non-euclidean spaces. This is a rather extreme variation of the original Kohonen algorithm because the entire idea was originally based on this metric. Ritter argues that many of the data we are dealing with are not represented well when projected by means of the Euclidean space. For instance, there are *directional data* which would need the surface S_2 of a sphere as a vehicle that transports the information content of the data well. For these cases it would make sense to use a spherical rather than a euclidean-topology. It is also doubtful if a d-dimensional Euclidean projection space gives enough freedom to map the neighbourhood of an item of a more complex information space into spatial relationships. This neighbourhood might just be too restricted to rebuild the information content of the input. Ritter (1999) suggests so-called *hyperbolic* spaces to overcome this limiting factor of the Euclidean spaces and are of special interest for *hierarchical structure* data types. For this sort of space the size of a neighbourhood increases exponentially within a certain radius r around a point.[29] The experiments with artificial data accomplished by Ritter under consideration of hyperbolic spaces exhibited promising results. Research in this field is still in the very beginning and – to the best of the author's knowledge

[29] Note, that the size of the neighbourhood increases with power law in the Euclidean space based SOM.



(a) Structure of toroid

(b) Toroid with color-coded data

Figure 3.7: Toroid representation of SOM trained on 300 normally distributed random sets with three components each

– real world data have never been used for training and mapping with this class of SOM.

A natural extension of Kohonen's Maps is the embedding of the method into a larger model framework. In literature one can find the use of Radial Basis Functions and Neural Networks for the approximation of the codebook vectors and their adjunctive functions.[30] We will seize some of the suggestions in this thesis and come to a more detailed specification in the second part of this thesis.

3.5 Chapter Summary

The major part of this chapter was devoted to describe the SOM algorithm in great detail. As the basic underlying technique used in this dissertation it was explained in a detailed manner. Vector Quantisation and Projection were discussed and the fundamental SOM algorithm was derived. The purpose and the structure of the neighbourhood function and the learning rate was explained in detail and a big part of the chapter was addressed to the implementation of SOM as a graphical tool. The chapter was closed by mentioning modifications and extensions of the basic algorithm.

Self-Organizing Maps have gained a lot of interest and contributions in the last 25 years since their appearance in 1982. They have entered in many different disciplines and have proved their usefulness in a diversity of practical applications. Their range of applications reaches from classical neuro sciences to digital imaging and data

[30] See e.g. Blayo et al. (2003), Cottrell et al. (1998), Deboeck and Kohonen (1998) or Ozdemir and Erkmen (1993).

base management. From an academical point of view SOM offer the possibility to analyse data without any presumptions on the distribution or structure. Kohonen's Maps are flexible and versatile and in the future they will play an important role for economic model building.

In the remainder of the thesis we will try to apply Kohonen's Self-Organizing Maps to economic problems. Most of the application offered in the next chapters will be concerned with effort to improve forecasting of financial data, but in chapter 7 the method is used to decide on whether countries are in economic *crises zones*.

Appendix to Chapter 3

A 3.1 Further Notes on the Euclidean Metric

Observable vectors usually must be represented in a space that has a *metric*, i.e. there exists a *distance function* $d(x, y)$ between all pairs of elements in the set of elements. The distance function should be chosen in consideration of the following four restrictions:

- $d(x, y) > 0$,
- $d(x, y) = 0$ iff $x = y$,
- $d(x, y) = d(y, x)$ and
- $d(x, y) \leq d(x, z) + d(z, y)$.

One of the most famous examples for a function that meets all these conditions is the *Euclidean Distance* which is exclusively used in this thesis for the SOM. Let $x = (\zeta_1, ..., \zeta_n)$ and $y = (\eta_1, ..., \eta_n)$ be two n-dimensional vectors, then the Euclidean Distance is defined as

$$d_E(x, y) = \sqrt{(\zeta_1 - \eta_1)^2 + (\zeta_2 - \eta_2)^2 + ... + (\zeta_n - \eta_n)^2} \tag{A 3.1}$$

or in short form

$$d_E(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (\zeta_i - \eta_i)^2}. \tag{A 3.2}$$

The Euclidean Distance can be interpreted as a measure of similarity, actually that of *dissimilarity*. The *Minkowski Metric* is the straightforward generalization of the Euclidean Distance and has the form

$$d_M(x, y) = \left(\sum_{i=1}^{n} |\zeta_i - \eta_i|^{\lambda} \right)^{\frac{1}{\lambda}}, \tag{A 3.3}$$

where $\lambda \in \Re$. If $\lambda = 1$ we obtain the special case of the *City-Block Distance*.[31]

A 3.2 Derivation of the Vector Quantisation Algorithm

The derivation of the Vector Quantisation Algorithm follows closely the one given in Kohonen.[32]

Let the probability density $p(x)$ be a continuous function. As we have seen in section 3.2.2 the main problem consists of the discontinuous jumps of the index $c = c(x; m_1, ..., m_N)$. Consider $\{a_i\} \in \Re_+^n$ as a set of scalar numbers. It then holds generally that

$$\min_i \{a_i\} \equiv \lim_{r \to -\infty} \left[\sum_i a_i^r \right]^{\frac{1}{r}}. \tag{A 3.4}$$

Furthermore, the functional form

$$f(x, r) = (1 + |x|^r)^{\frac{1}{r}} \tag{A 3.5}$$

[31] Kohonen (2001) p. 18
[32] Kohonen (2001) pp. 60-62

is needed for another result we need for the derivation. The function f or $\lim_{r\to-\infty} f$ is not differentiable at $x \in \{-1, 0, 1\}$, i.e. we have to exclude these values. Generally, the following holds

$$\lim_{r\to-\infty} \frac{\partial f}{\partial x} = \frac{\partial}{\partial x}\left(\lim_{r\to-\infty} f\right). \tag{A 3.6}$$

Note, that a function of the form $\left(\sum_i \|x - m_i\|^r\right)^{\frac{2}{r}}$ is continuous, well-defined, single-valued, and continuously differentiable in its arguments (apart from a situation when $x = m_i$). This fact helps us forming the gradient $\nabla_{m_j} E$.

Moreover, the cases when one of the $\|x - m_i\|^r$ is exactly equal to the sum of the other terms have to be excluded. Under the assumption of stochastic x and continuous probability density function $p(x)$ all these singular cases have zero probability.

Taking this conditions into account the following gradient and limit operations can be exchanged:

$$\|x - m_c\|^2 = \left[\min_i \{\|x - m_i\|\}\right]^2 = \lim_{r\to-\infty}\left(\sum_i \|x - m_i\|^r\right)^{\frac{2}{r}}, \tag{A 3.7}$$

and also

$$\nabla_{m_j} E = \int \lim_{r\to-\infty} \nabla_{m_j}\left(\underbrace{\sum_i \|x - m_i\|^r}_{A}\right)^{\frac{2}{r}} p(x)dx. \tag{A 3.8}$$

Hence,

$$\begin{aligned}\nabla_{m_j} A^{\frac{2}{r}} &= \frac{2}{r} A^{\frac{2}{r}-1} \cdot \nabla_{m_j}(\|x - m_j\|^r) \\ &= \frac{2}{r} A^{\frac{2}{r}-1} \cdot \nabla_{m_j}(\|x - m_j\|^2)^{\frac{2}{r}}.\end{aligned} \tag{A 3.9}$$

It is worth noting that there is no summing over j. After some algebra and rearrangement

$$\nabla_{m_j} A^{\frac{2}{r}} = -2 \cdot \left(A^{\frac{2}{r}}\right) \cdot \underbrace{\frac{\left(\|x - m_j\|^2\right)^{\frac{r}{2}-1}}{A}}_{B} \cdot (x - m_j) \tag{A 3.10}$$

is obtained. Because of equation A 3.7

$$\lim_{r\to-\infty} A^{\frac{2}{r}} = \|x - m_j\|^2 \tag{A 3.11}$$

holds. In a next step, B can be rewritten as

$$B = \frac{\|x - m_j\|^r}{\sum_i \|x - m_i\|^r} \cdot \|x - m_j\|^{-2} = \left(\sum_i \frac{\|x - m_i\|^r}{\|x - m_j\|^r}\right)^{-1} \cdot \|x - m_j\|^{-2}. \tag{A 3.12}$$

When $r \to -\infty$ the term $\|x - m_i\|^r / \|x - m_j\|^r$ has its maximum when $m_i = m_c$ and begins to predominate the other terms. It follows that

$$\lim_{r\to-\infty} B = \lim_{r\to-\infty}\left(\frac{\|x - m_j\|}{\|x - m_c\|}\right)^r \cdot \|x - m_j\|^{-2} = \delta_{cj}\|x - m_j\|^{-2}, \tag{A 3.13}$$

where δ_{cj} is the Kronecker delta.[33]

Combining the partial results we obtain the following statements

$$\begin{aligned} \lim_{r\to-\infty} \nabla_{m_j} A^{\frac{2}{r}} &= -2 \cdot \|x - m_c\|^2 \cdot \delta_{cj} \cdot \|x - m_j\|^{-2} \cdot (x - m_j) \\ &= -2 \cdot \delta_{cj} \cdot (x - m_j) \end{aligned} \tag{A 3.14}$$

and

$$\begin{aligned} \nabla_{m_j} E &= \int \lim_{r\to-\infty} \nabla_{m_j} A^{\frac{2}{r}} \cdot p(x) dx \\ &= -2 \int \delta_{cj}(x - m_j) p(x) dx. \end{aligned} \tag{A 3.15}$$

The sample function of the gradient at time t is

$$\nabla_{m_j} E|_t = -2 \cdot \delta_{cj}[x(t) - m_j(t)]. \tag{A 3.16}$$

The steepest descent occurs in the direction of $-\nabla_{m_j} E|_t$.

When the indices are changed and we define the step size by the factor $\alpha(t)$ (this parameter includes the constant -2), we obtain

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot \delta_{cj}[x(t) - m_i(t)], \tag{A 3.17}$$

which corresponds to the basic SOM algorithm.

[33] The Kronecker delta equals 1 for $c = j$, and 0 otherwise.

Part two

Empirical Evidence and Computational Simulations

Chapter 4

Pattern Recognition Applied to Financial Time Series

4.1 Introduction

The following chapter provides a first application of SOM to real world data. The idea presented here represents the most basic form of an application of the SOM in the scope of this thesis. The subsequent chapters are basically extensions of this chapter's content and are based to a large extent on the findings in this part of the present work. Only chapter 7 is an exception and uses the Kohonen algorithm in a completely different manner.

Here, we suggest an application of Self-Organizing Maps to identify certain patterns in financial time series (prices). These patterns are characterised by their geometrical forming. Once these "symbols" are found, their information content is evaluated and trading rules are developed. Implicitly, we assume that financial patterns recur in unknown frequencies and when a certain event has shown up we try to extract relevant information from that pattern. Methods of filtering relevant information from these small pieces of time series are manifold and will be subject to further analysis.

The motivation of this research is simple. In the past decades an enormous effort has been made to find explanations of the development of prices, returns, and volatility by building models based on fundamental values. These models have gained a prominent position in literature, but, unfortunately, they often fail to forecast the development of markets. One of the main problems of this sort of models is the underlying assumption of just one single representative agent who buys or sells assets, currencies or goods. This is a very critical hypothesis since market forces might be driven significantly by the interaction of heterogeneous individuals and not by a symmetric movement of behaviour of a homogenous group of market participants.

Therefore, many papers on agent-based models and the underlying economic theory suggest that there are – at least – two types of investors acting on the markets: fundamentalists and chartists.[1] Hence, in the past ten years this family of models has become popular. Agent-based models need heterogeneous agents in order to

[1] See e.g. Grauwe et al. (1993)

generate market dynamics. The common way to model heterogeneity is the classification of agents into the two groups: fundamentalists and chartists. Our concern in the following chapter lies mainly in the explicit contemplation of a typical chartists' approach: trend interpretation. Many professional traders rely on technical analysis and follow trends, use extrapolations, and interpret patterns.[2] This group of traders does not perform as bad as one could expect ex-ante and surveys indicate a strong connection between day-to-day trading decisions and chartist methods.[3] It is worthwhile to pay regard to the link between the development of asset prices and the use of more or less sophisticated technical analysis.

We try to analyse the relevance of profit possibilities on the following pages. The analysis is accomplished in three different steps: first, a SOM is employed to find recurrent patterns in a price time series. These patterns are grouped by the SOM and representative model vectors are built for each cluster. In a second step for each group the *normalised return curve* is constructed. Lastly, from the normalised return curves buying or selling signals are extracted which can be used to formulate automated trading rules.[4]

4.2 Data Description

Asset price returns usually exhibit a number of so-called *stylised facts*. These facts are well-known from finance literature and have been studied extensively. A comprehensive documentation of stylised facts is given by Engle, Bollerslev and Nelson.[5] The authors describe typical distributional characteristics observable with asset price returns which consist mainly of four points:

- **Leptokurtosis:** is also known under the term *excess kurtosis* and refers to a non-normal asset return distribution (i.e. the distribution has fat tails and narrower shoulders compared to the normal)
- **Clustering:** periods with a consistently high degree of volatility are followed by periods of consistently low degree of volatility; this phenomenon is often explained through the arrival of new information on the market or varying trading volume
- **Leverage Effect:** also known as *Fisher-Black-Effect*, states the negative correlation between asset returns and volatility; decreasing stock prices add more uncertainty to the agents' portfolio which manifests in higher volatility

[2] Bulkowski (2000)
[3] Taylor and Allen (1992)
[4] The model was originally developed in Cheng and Wang (2002) pp. 203–216.
[5] Bollerslev et al. (1994)

- **Long Memory:** is usually explained through the *hyperbolic decay* of the autocorrelation function of the volatility (e.g. absolute returns or squared returns); this fact shows a high degree of persistency on financial market

The data we use for this experiment is from the Taiwan Stock Index from 1/5/71 to 26/3/97 and includes 6842 observations.[6] The data is presented in figure 4.1. In subfigure (a) the raw price time series is plotted and in (b) the prices are transformed to returns by employing the log-differences $r_t = \ln(P_t) - \ln(P_{t-1})$. The return series exhibits the expected features of a stock index time series. The returns are leptokurtic (excess kurtosis: 9.91) from figure (b) it is easy to see the tendency of volatility clustering. The distribution of the returns is clearly non-normal, as can be seen from the quantile-quantile plot in subfigure (c). This confirms the presence of the above described stylised facts and identifies the time series at hand as a representative one for usual index return series.

4.3 Embedding Theorem and Regressor Building

The model presented in this chapter is designed to identify informative patterns from a financial time series. Therefore, it is a fundamental work to preprocess the time series in such a way that we reshape the vector of observations to a matrix that translates the single values into patterns.

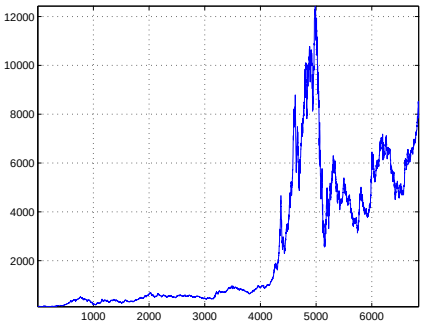
Suppose $\mathbf{x}$ is a vector of observations from time $t = 1, ..., T$. In order to translate this vector into a matrix of patterns we apply a *sliding window* that moves along the entire array of observations and "cuts" it into smaller pieces of length p. Each piece (herinafter called $y(t)$) represents a pattern that might contain information about the overall dynamics of prices. Each row of the matrix $\mathbf{y}$ can be formulated as

$$y(t) = \{x_{t-p+1}, ..., x_t\} \quad \text{for} \quad t = p, ..., T. \tag{4.1}$$

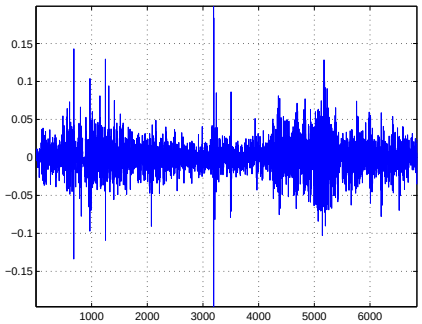
The result of this transformation is a regressor matrix $\mathbf{y}$ of size $[(T - p + 1) \times p]$. This transformation follows the manipulation suggested by Floris Takens ("*Takens' Embedding Theorem*").[7] The idea of embedding a scalar valued time series in historical values mainly stems from the literature of the estimation of chaotic time series. If we consider the observations $x(t)$ to be the projection of a multidimensional state space onto the one-dimensional axis of the $x(t)$'s then a time-delay embedding serves to convert the observations back into its multidimensional state space, i.e. the embedding *unfolds* a representative of the original system.

[6] All data are obtained from *Thomson Financial Data* (DATASTREAM). The time series used here is the weighted price index (Code: TAIWGHT(PI)).

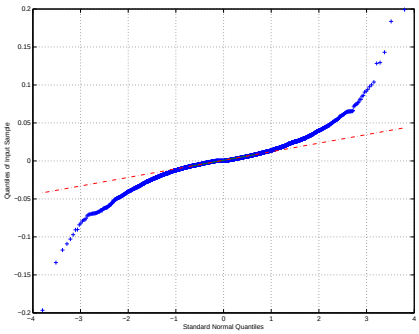
[7] Takens (1981)



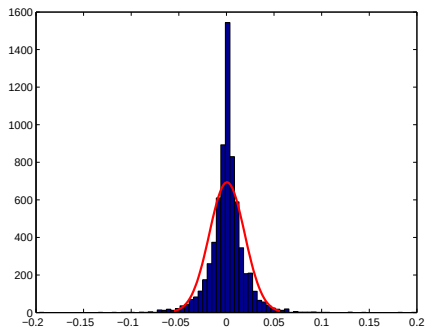
(a) Price Time Series



(b) Return Series



(c) QQ-Plot of Sample Data vs. Standard Normal



(d) Histogram of the Returns

Figure 4.1: Taiwan Stock Index time series from 1/5/71 to 26/3/97 (6842 observations)

The resulting matrix of this procedure looks as follows:

$$\mathbf{y} = \begin{pmatrix} x_1 & x_2 & \cdots & x_p \\ x_2 & x_3 & \cdots & x_{p+1} \\ \vdots & & \ddots & \vdots \\ x_{T-p+1} & & \cdots & x_T \end{pmatrix} \tag{4.2}$$

Each row in the matrix $\mathbf{y}$ represents a small piece of price chart pattern taken from the original time series. In the remainder of this chapter we call the rows of matrix $\mathbf{y}$ *data sample vectors* (or: *data input vectors*).

In graphic 4.2 it is shown how the sliding-window proceeds. The window starts at time $t = 1$ and covers the next two observations. This horizon is denoted as the embedding dimension p. In the next step the window is shifted one step ahead and includes, therefore, time $t = 2$ to $t = 4$. The procedure is repeated until the end of the time series extracting $(T - p + 1)$ *patterns* for the input data vector which can conveniently be analysed by means of Self-Organizing Maps.

Takens (1981) found that when an attractor has a dimension d_A all self-crossings of the attractor[8] will be eliminated when the dimensionality is increased, or, more specifically, $d > 2d_A$. Note that this is only a sufficient, but not a necessary condition. In other words if an attractor exhibits a dimension $d_A = 4.02$ an adequate embedding dimension could be $d = 5$. But the theorem only gives us the information that e.g. $d = 9$ will for sure work with the underlying system. However, the minimal embedding dimension is not yet determined. For a mathematician this does not play an important role but for economics applications any additional dimension larger than necessary leads to a strong increase in the use of computational resources. Therefore, it is necessary to find a method that identifies the minimum embedding dimension d_E.

4.4 False Nearest Neighbour Method

To this end, Kennel et al. (1992) suggest the *False-Nearest-Neighbour* method.[9] The basic idea is the analysis of the variation of neighboured points when passing over from dimension d to dimension $d+1$. The task of this operation is the identification of false neighbours of points.

In figure 4.3 the simple Hénon Map is plotted. Formally, the differential equation system corresponding to this map can be written down as

[8] *Self-crossings* in the orbit of an attractor is the result of the projection of the multidimensional data onto a one-dimensional axis.

[9] Kennel et al. (1992)

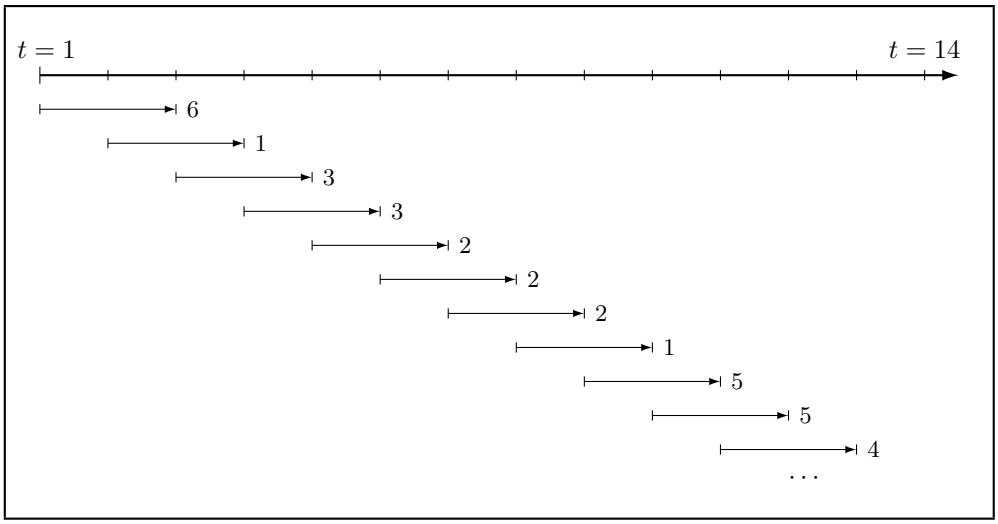


Figure 4.2: Sliding window device example

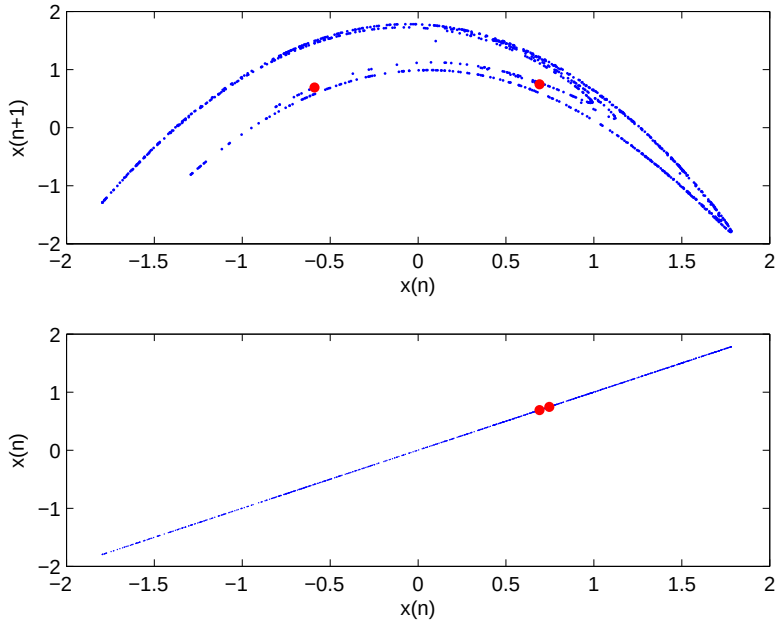


Figure 4.3: Hénon Map in one and two dimensional space

$$x_{t+1} = \gamma - x_t^2 + y_t \tag{4.3}$$
$$y_{t+1} = \beta x_t. \tag{4.4}$$

The parameter values are set to $\gamma = 1.4$ and $\beta = 0.3$ and the initial values x_0 and y_0 are both equal to 0.1. In the plot we see 1000 data points, represented in the one-dimensional case (lower plot) and its two-dimensional counterpart (upper plot). The red dots indicate data points which appear to be direct and nearest neighbours in the one-dimensional representation. When we pass to the case $d+1$ it becomes obvious that these two data points are only false nearest neighbours. If the dimension is too small to unfold the attractor, some points only appear to be neighbours because the geometric structure of the attractor has been projected onto a smaller space. These neighbours do not lie close to each other due to the dynamics of the data generating process. Kennel et al. (1992) propose to gauge the effects of adding an additional dimension by means of the Euclidian distance.

Let us assume we are dealing with a d-dimensional system and let r be the index for the r-th nearest neighbour of some embedded vector $x(n)$. The Euclidian distance between $x(n)$ and its neighbours $x^{(r)}$ can then be defined as

$$R_d^2(n,r) = \sum_{k=0}^{d-1} \Big(x(n+kT) - x^{(r)}(n+kT) \Big)^2. \tag{4.5}$$

By adding an additional dimension each of the vectors $x(n)$ is augmented by one coordinate. The new coordinate is simply $x(n+dT)$. Of course, the Euclidian distance changes after adding on dimension and can be formulated as

$$R_{d+1}^2(n,r) = R_d^2(n,r) + \Big(x(n+dT) - x^{(r)}(n+dT) \Big)^2. \tag{4.6}$$

It is useful to compare the numbers calculated in equations (4.5) and (4.6). If we observe a large increase in the distance we can conclude that it is worth to add a further dimension. Kennel et al. (1992) propose the following criterion as a possible measure for the distinction of real nearest neighbours and false nearest neighbours:

$$\sqrt{\left(\frac{R_{d+1}^2(n,r) - R_d^2(n,r)}{R_d^2(n,r)} \right)} = \frac{\mid x(n+Td) - x^{(r)}(n+Td) \mid}{R_d(n,r)} > R_{tol}. \tag{4.7}$$

The value R_{tol} is a threshold variable which has to be determined numerically. In the literature the results found by Kennel et al. (1992) ($R_{tol} \geq 10$) are standard because by using this value all false nearest neighbours are clearly identified. Note that the method so far does not imply a sufficient condition for determining the embedding dimension. The optimal embedding dimension is still unknown.

Another problem arises when one has only a limited amount of data points at hand. The nearest neighbour $\mathbf{y}^{(1)}(n)$ may be a *real* one but is actually not *close* to $\mathbf{y}(n)$. The reason for this lies in the distribution of the data. For instance, if the data is uniformly distributed (which can be interpreted as white noise) the false nearest neighbour method suggests the embedding it into a very small dimension. Increasing the dimension d and keeping the number of data points constant the Euclidean distance has to rise as well because adding a further dimension leads to an expansion of the space and due to the uniform distribution condition the data is more spreaded. When we allow the data points or *observations* to increase, too, d would also diverge to infinity.

In economics time series usually do only include a finite number of observations, therefore, a further criterion is necessary in order to account for this typical characteristic. Let $\mathbf{y}^{(1)}(n)$ be the nearest neighbour to $\mathbf{y}(n)$ but they are not close to each other (i.e. $R_d(n) \approx R_A$) then the distance between the two of them after adding an additional dimension will be $R_{d+1}(n) \approx 2R_A$.[10] This result leads to the second condition of the false nearest neighbour method: the distance $R_{d+1}(n)$ has to be larger than $2R_A$. This is the only way to ensure the identification of distant neighbours which in fact are false. More generally, this criterion can be noted as

$$\frac{R_{d+1}(n)}{R_A} > A_{tol} \tag{4.8}$$

where

$$R_A^2 = -\frac{1}{N}\sum_{n=1}^{N}(x(n)-\bar{x})^2\,, \tag{4.9}$$

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N}x(n) \tag{4.10}$$

and A_{tol} is some threshold value. Numerical analysis by Kennel et al. (1992) prove R_A as written in equation (4.8) to be a good choice. Other measures are conceivable but would not change the results much.

In figure 4.4 the method is applied to the Hénon map (equations (4.3) and (4.4)). For generating this graph we used 10000 data points. We can see how the unfolding of the attractor affects the reduction of the false nearest neighbours. The percentage of false nearest neighbours decreases rapidly in the passage from dimension $d = 1$ to $d = (d+1)$. When going one step further and adding an additional dimension we observe that almost all false nearest neighbours vanished. This is a good indicator for using $d_E = 3$. Rhodes and Morari (1997) express some doubts on the validity of the method when the original time series is contaminated with (*white*) noise. They argue that in large data sets, even very small amounts of noise can make the

[10] R_A is as defined in equation (4.9).

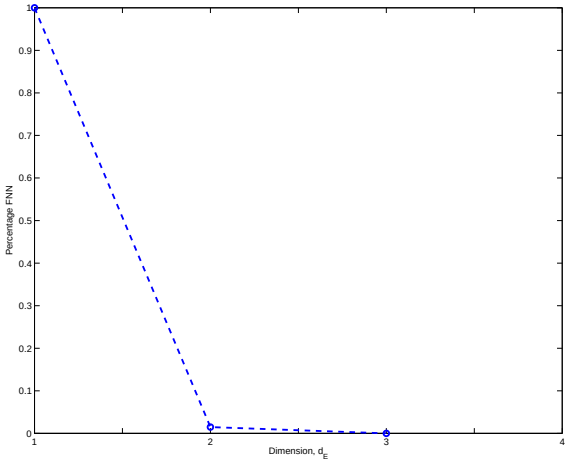


Figure 4.4: False Nearest Neighbour percentages for the Hénon Map

embedding dimension prediction erroneous. The larger the time series in question grows, the stronger is the impact of failing. Since we are only dealing with time series of relatively limited availability of data, for our purposes the determination of the embedding dimension by means of the original method proposed by Kennel et al. (1992) is sufficient.

Even if this method represents a reasonable approach to determine the best embedding dimension for chaotic time series, it remains questionable if it is also the right method for every kind of financial time series. Furthermore, the False Nearest Neighbour method suggests in some cases an embedding dimension which is very small (i.e. $d_E = 2$ or $d_E = 3$) and prevents the use of the biggest advantage of the SOM: the extraction of geometrical similarities. The problem with the False-Nearest-Neighbour method is that, unfortunately, it is not applicable to every problem. The method works for stationary time series but is not (always) suitable for instationary data (what prices clearly are). Furthermore, the objective of the research is an important determinant of the autoregressive order to be chosen. When it is important to save CPU time the False-Nearest-Neighbour technique performs well, but it will not be able to reveal further information. In order to expel mistakes by trusting blindly in the False Nearest Neighbour method we will vary the embedding, disregarding the initially proposed dimension and choose the number of lags with the best (ex-post) forecasting performance.

4.5 Application of SOM

In the next step our aim is to reduce the number of data sample vectors in order to find a representative set of *model vectors* which are able to explain most of the dynamic behaviour of the time series. For this purpose SOMs are a natural method to use. Kohonen's Maps are a powerful tool when high-dimensional data is in question. Since no ex-ante assumption has to be made in order to train the network it is very convenient to apply. The SOM tries to adjust the net to the data in the hyperplane. Each neuron on the net corresponds to one model vector. After training, these model vectors are centres of clusters which unify a group of "similar" data sample vectors. Neighbouring nodes are more equal than neurons with larger distances on the net. Each pattern we extracted from the data vector by means of the method described in equation (4.1) is now assigned to one of the model vectors found by the SOM. This is demonstrated in figure 4.2. Behind every pattern we can see a number that indicates the affiliation of the respective pattern to a certain neuron.

Before training the net it is necessary to normalise the embedded data matrix between 0 and 1, but without changing the geometric characteristics of the data. This is important because Self-Organizing Maps are based on Euclidean metrics. Hence, if we compare a pattern from the very beginning of the original time series with a pattern from the 1990's they may look similar (and should therefore be in one cluster) but the real values are so different from each other that the SOM is not able to discover the obvious similarity. The numerical values are just too far away from each other. Hence, even with similar symbolic characteristics there is no chance for the SOM to discover the mutual nature of the patterns. The procedure would give too much weight on higher values. We are only interested in the geometrical characteristics of a pattern and are not concerned about the values. Thus, we normalise the data set according to equation (2.37) by taking

$$x^{\star}_{k,t} = \frac{x_{k,t} - \min(x_k)}{\max(x_k) - \min(x_k)}, \tag{4.11}$$

where k denotes the pattern or row, respectively.

The parameters used to train the net are summarized in table 4.1. For a first approach to this kind of analysis we chose the parameter set-up as exactly equal to the one used in Cheng and Wang (2002). This has the advantage of comparability of the results. Note, that the overall length of the initial price time series is shorter in our analysis. Cheng and Wang (2002) worked with 7435 observations; we were only able to obtain a vector of 6842 observations.[11] This fact should not affect the

[11] We chose to use the data provided by *Thomsen Financial Data* (DATASTREAM) which is to the best of the author's knowledge the most reliable selection. Cheng and Wang (2002) never replied to the question how the additional data were obtained.

analysis extensively since there is still a sufficient number of data points available in order to assure a faithful convergence of the system.

The embedding dimension or *pattern length* (n) is chosen to be 125. Other than the values suggested by the False-Nearest-Neighbour Method described in the previous subsection we choose a significant higher embedding dimension. The reason for this is twofold. First, we want to assure a model design that is as close as possible to the one suggested in the paper by Cheng and Wang (2002), and second, for financial applications it makes sense to cover larger time periods in order to discover meaningful geometric patterns which reflect market reactions to certain events. Apparently, the False-Nearest-Neighbour method is solely applicable to data series of chaotic nature.

We constructed our net by 36 neurons in the two-dimensional space. Each side of the net is built of six nodes (symmetric net). We split the training of our SOM into two different phases: the *rough training phase* and the *fine tuning phase*. This is standard procedure when using Self-Organizing Maps and serves to increase the speed of convergence and also the exactness of adjustment. As a starting point the model vectors are generated randomly. This means in the beginning of the iteration process almost every node is updated including large parts of the respective neighbourhood. Once the system has settled it is no longer necessary to let a large neighbourhood be affected by the update of one single node. Only the node itself and its direct neighbours are now of interest. With increasing steps of iteration the programme decreases the value of the neighbourhood function σ. The same argument applies for the learning-rate function η.

4.5.1 Some Results

The system is trained 1000 epochs during the rough training phase and the fine tuning phase each. This suffices to find a reliable network. After these 2000 training steps the SOM has identified 36 different model vectors which are presented in figure 4.5. Every chart corresponds to one winning neuron. By construction there must be 36 model vectors afterwards, no more and no less. The number of neurons is determined in advance and is a critical parameter and must be set arbitrarily by the user of the method. The more neurons are chosen the more similar the neighbourhood of a node will be. This, of course, includes the danger of *overfitting*.[12] On the other hand choosing a too small number of neurons relevant information might get lost because the net is no longer appropriate to represent the real structure of the data and becomes an imprecise mapping.

In figure 4.5 we recover many of very well-known patterns from financial mar-

[12] In this framework the term *overfitting* is used in differing way than usual. By overfitting the danger of assigning patterns to nonrelevant model vectors is meant. This will give weight to unimportant features and distorts the forecasting capability of the model.

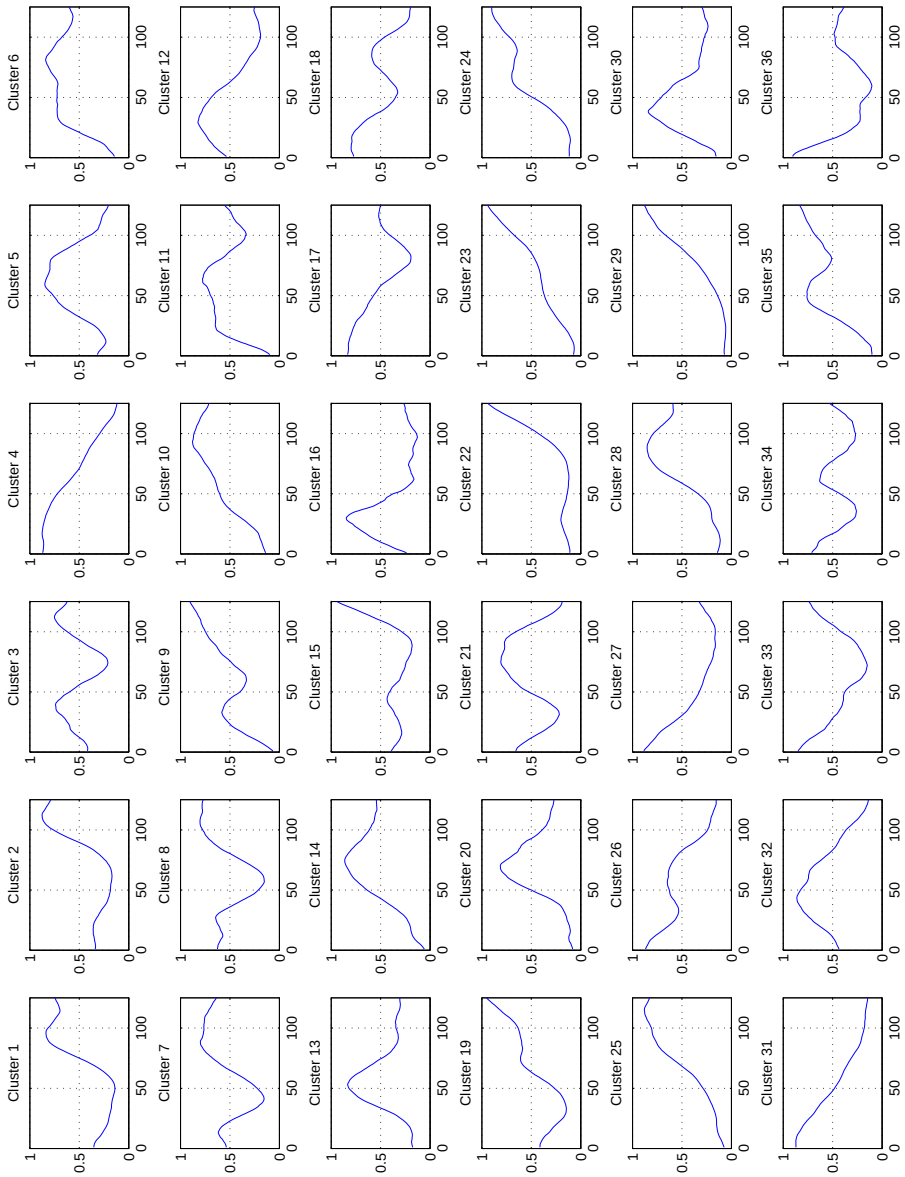


Figure 4.5: 36 model vectors identified by SOM

Parameter	Value
Number of Trading Days (T)	6842
Number of Data Sample Vectors (N)	6718
Pattern Length (n)	125
SOM Dimension	2
Grid Size	$[6 \times 6]$
Initial Learning Rate (Rough Phase) η_0^r	0.90
Initial Neighbourhood Radius (Rough Phase) σ_0^r	6.00
Final Neighbourhood Radius (Rough Phase) σ_1^r	1.00
Epochs (Rough Phase)	1000
Initial Learning Rate (fine tuning) η_0^f	0.1
Initial Neighbourhood Radius (fine tuning) σ_0^f	1.00
Final Neighbourhood Radius (fine tuning) σ_1^f	0.10
Epochs (fine tuning)	1000

Table 4.1: Model parameters

kets, such as uptrends, downtrends, hump-shaped patterns or broadening bottoms.[13] This early result gives us a first hint of the SOM's capability to classify time series patterns in a proper way. The grouping of similar events on financial markets and the reduction of the data to a small number of representative patterns is an interesting application of SOMs, but has not yet elaborated any information that could be used for forecasting.

The next step of the analysis consists of seeking a method that makes use of chart pattern groups in order to filter their significant content. To this end, we follow closely the analysis of Cheng and Wang (2002) in this chapter. In the next section we try to develop *normalised return curves* and will then use these curves to develop simple trading rules.

4.5.2 Mean Return Curves

From visual inspection of the winning neurons only it is not possible to deduct trading rules. We are interested in the return behaviour of a chart once it was observed. To analyse the returns the most convenient way is the construction of a representative return curve for each of the discovered patterns. To this end, we have to depart from the "normalised world" and turn back into the values of the original time series. Before we can start calculating the returns we have to determine which pattern belongs to which neuron, or, in more technical terms, we have to calculate the *Best-Matching-Unit* (BMU) for each sample data pattern. The BMU

[13] For an extensive description of "typical" chart patterns appearing on financial markets see Bulkowski (2000).

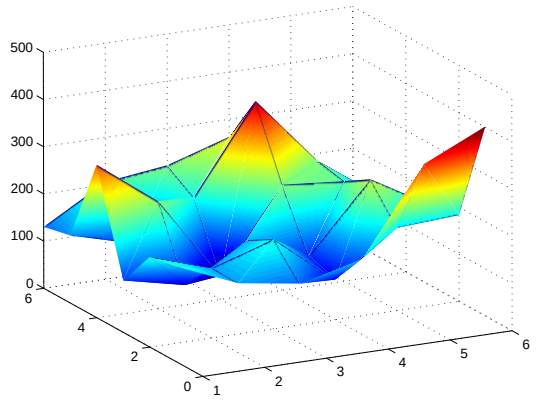


Figure 4.6: Hit-histogram of the TAIWAN stock exchange data

is corresponding to equation (3.8) defined as

$$BMU = \arg\min_i \{\| \mathbf{x} - \mathbf{m}_i \|\}, \tag{4.12}$$

where $\mathbf{m}_i$ for $i = 1, \cdots, 36$ are the model vectors and $\mathbf{x}$ is the matrix of data sample patterns N. The resulting BMU is a vector of the length N and its entries are integers within the interval $[1; 36]$. Having identified the winning neuron for each data sample vector we can now calculate the (non-normalised) return series after a chart pattern from group m_i appeared, i.e.

$$R_{h,i} = \ln(P_{t+h}) - \ln(P_t) \quad \text{for} \quad h = 1, 2, \cdots, k, \tag{4.13}$$

where t is the trading day at which pattern i ends to appear and h is the time horizon in question (h obtains at most the value $(n-1)$). The transformation leads to a matrix of dimension $[\delta_i \times h]$ where δ_i is the number of data sample vectors associated with neuron i. Hence, each row corresponds to a return series connected with model vector i. We now have to complete the derivation of the mean return series by computing

$$\bar{R}_{h,i} = \frac{\sum_{j=1}^{\delta_i} R_{h,j}}{\delta_i}. \tag{4.14}$$

Note that the values of $\bar{R}_{h,i}$ are not normalised between $[0; 1]$ and reflect the true returns calculated directly from the original data set.

In figure 4.7 the mean return curves of the aftermath of each representative model vector in figure 4.5 are shown. We find several return patterns that are monotonically increasing or decreasing, respectively, and others which are not exhibiting unambigious dynamics. This result is in line with the patterns developed in Cheng

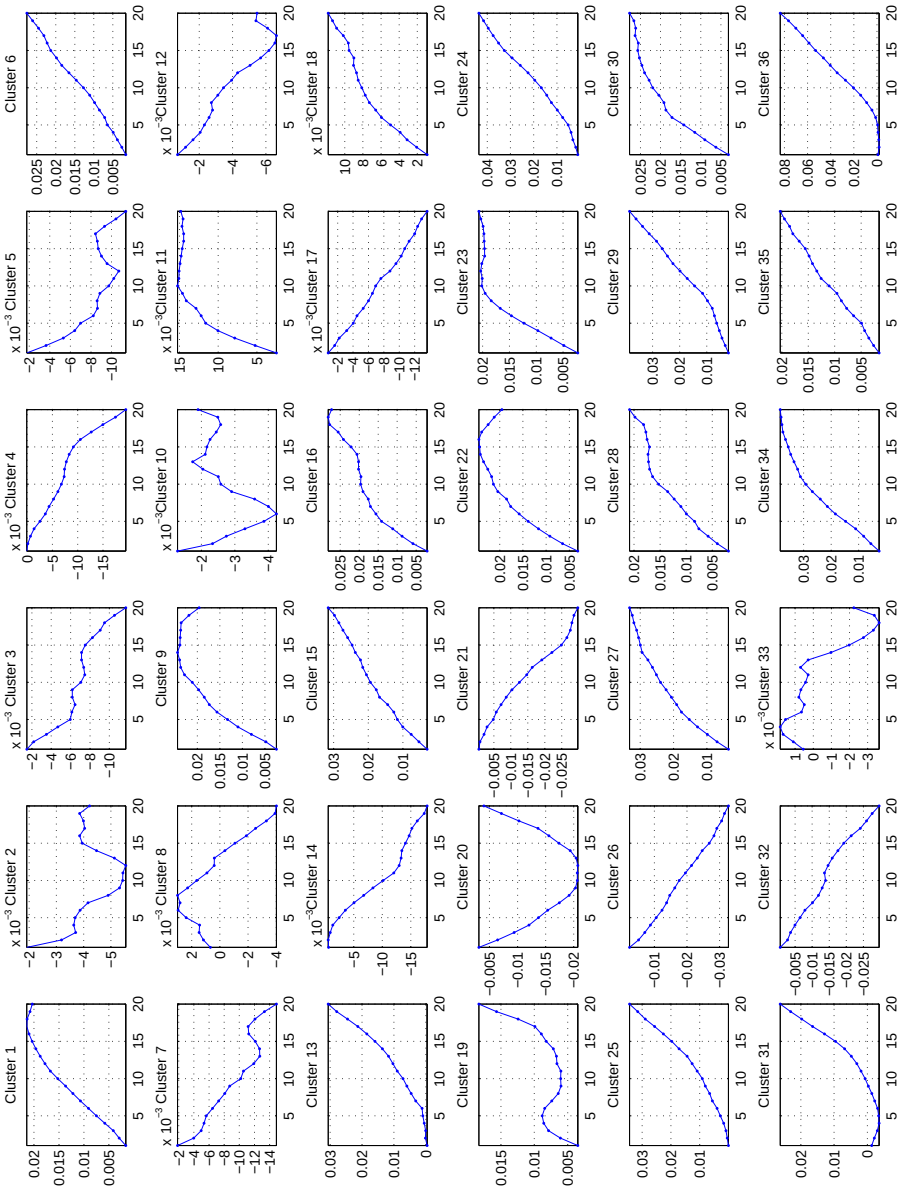


Figure 4.7: Mean return curves $\bar{R}_{h,i}$ in the aftermath of the appearance of chart m_i

and Wang (2002). What do the representative return series tell us about the overall movement of the returns? We are certainly able to deduce uptrends and downtrends from the mean return curves. Even in the non-monotonic cases it is possible to at least state the direction of movement in terms of trends. In the next subsection we will develop some trading rules based on the observations after an "event" (a chart pattern) has shown up.

4.5.3 Development of Trading Rules

The trading rules given in Cheng and Wang (2002) are based on the idea that an agent can buy, sell or hold an asset on the market. Two points of departure are conceivable. First, the agent is in a *long position*, i.e. she owns a unit of the respective asset. Once the agent receives a *sell signal* she will sell the asset remaining with zero units. But additionally she will *sell short*, i.e. she sells a further unit of the asset and waits to buy it back later. This transaction leaves the agent with a total amount of -1 units. On the other hand the agents might start from a short position (holding -1 units of the asset). Once she receives a *buy signal* the trader will buy back the unit from the previous short-trading action and buys an additional unit of the asset. This transaction leaves her with one unit of the respective asset in her portfolio.

The trading signals are transmitted via the return behaviour in the direct aftermath of the appearance of a certain pattern. In subsection 4.5.2 we calculated the normalised return curves for each cluster identified by the SOM. In this subsection we make now use only of the first representative return from the respective cluster, i.e. $\bar{R}_{1,i}$. We can define the trading action as

$$TradeAction = \begin{cases} buy & \text{if} \quad \bar{R}_{1,i} > 0 \\ sell & \text{if} \quad \bar{R}_{1,i} < 0 \\ hold & \text{if} \quad \bar{R}_{1,i} = 0 \end{cases} \tag{4.15}$$

From this definition an automated trading rule can be established. Let the trader hold minus one unit of an asset in period $t-1$. If at time t the trader observes a buy signal she purchases a one unit of the asset for the spot price in t. Of course, we have to consider the short-selling oportunity of the trader as well. Therefore, she buys back the unit she has sold short before. The profit of the overall transaction is then simply the difference of the short-selling income of the period before minus the spending on the additional purchase of one unit of the respective asset. Hence, profit can be defined as

$$\pi_0 = 0 \tag{4.16}$$

and

$$\pi_t = \begin{cases} 0 & \text{if } H_{t-1} = 0; H_t = 1, -1, \\ \pi_{t-1} & \text{if } H_t = H_{t-1}, \\ \pi_{t-1} + P_t(1 - c_1 - c_2) - P_{s(t)}(1 + c_1) & \text{if } H_t = -1, 0; H_{t-1} = 1, \\ \pi_{t-1} - P_t(1 + c_1) + P_{s(t)}(1 - c_1 - c_2) & \text{if } H_t = 1, 0; H_{t-1} = -1 \end{cases} \tag{4.17}$$

where c_1 is the tax rate incurring by every transaction and c_2 is the tax rate of securities exchange income.[14] The variable H indicates the stock of assets of the trader and can take one of the three values $\{-1, 0, 1\}$. In the case $H = -1$ the trader has short-sold one security, for $H = 0$ he holds zero securities in his portfolio, and consequently the trader holds one security when $H = 1$. The (index-)function $s(t)$ is defined as

$$s(t) = \max_j \{j | H_j \neq H_t, H_{j-1} = H_t, 1 \leq j < t\}. \tag{4.18}$$

and determines the time an asset was traded the last time, i.e. the definition (4.18) takes into account the correct price of the asset at the time when it was purchased or short-sold, respectively. In this way price movements during periods of holding an asset are disregarded and only the effective prices are taken into account for evaluating the performance of the investment. The time line of the procedure is shown graphically in figure 4.8. Applying the trading strategy and employing it for the Taiwan Stock Index leads to a profit development as plotted in figure 4.9.

From figure 4.9 we can easily see that the profits obtained by applying the SOM-based automated trading rule is **not able** to outperform the Buy-and-Hold strategy systematically. In fact, during very long periods the rule leads to equal or even significantly worse results than the simple benchmark strategy (*Buy-and-Hold*). There are unambiguous outmatching periods for the trading rule as well as for the benchmark strategy. Only after $t = 5000$ the profitability of the SOM induced trading rule properly exceeds the profits gained from the Buy-and-Hold strategy. This picture is in line with the results found by Cheng and Wang (2002). Of course, it is not quite clear whether the condition suggested in equation 4.17 the appropriate method to determine the profitability really is. In the literature a vast number of trading strategies can be found which might lead to a completely different outcome. But the results found certainly give an incentive to go further with the research of SOMs.

[14] In our analysis we used the Taiwan rate of $c_1 = 0.001425$ and $c_2 = 0.003$ according to the work by Cheng and Wang (2002).

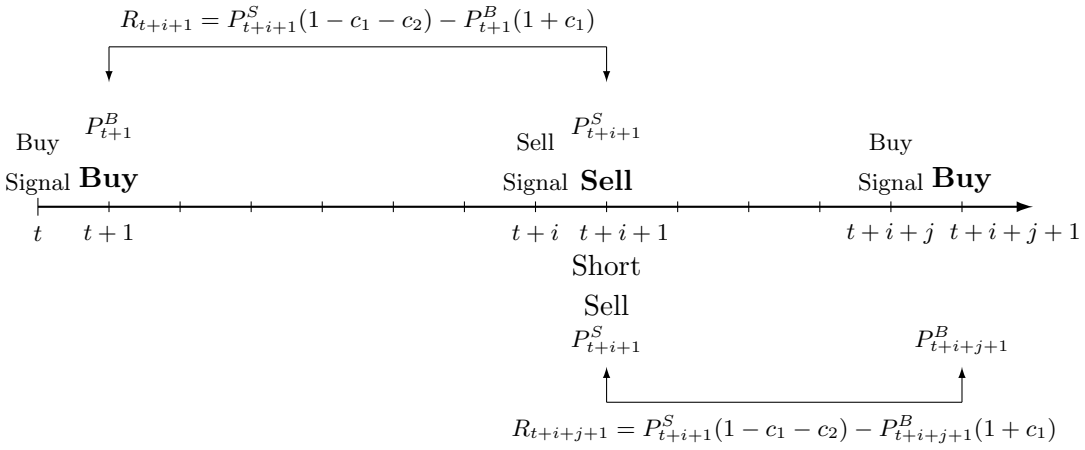


Figure 4.8: Time flow of trading rule

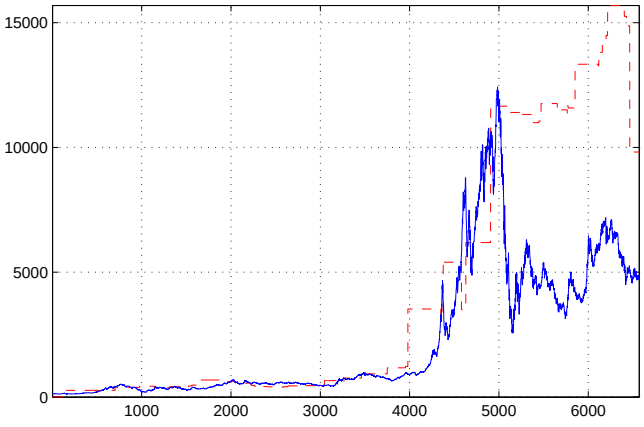


Figure 4.9: Profit of automated trading rule against buy-and-hold strategy

4.6 Chapter Summary

In this chapter a first application of Self-Organizing Maps was used with real-world data. The idea was to extract relevant and meaningful geometrical patterns from a return series which could be used to identify recurrent market regularities. To this end, the time series was cut into smaller pieces with the aid of the sliding-window device. Subsequently, the data pieces were normalised on the interval $[0; 1]$ and grouped to 36 geometric families (or: representative patterns). The non-normalised patterns belonging to the respective families were used to compute "mean-return-curves". In a next step a trading algorithm was implemented that acted according to the information given by the mean-return-curves.

In this chapter we can confirm the results from Cheng and Wang (2002). The direct comparison of the trading performance of the SOM-based method and a simple Buy-and-Hold strategy does not exhibit unambiguous advantages in favour of one of the techniques. In the next chapter we go one step further and try to integrate probabilities into the system and use it for forecasting.

Chapter 5

SOM Model with Probabilistic Connection

5.1 Introduction

The model in this chapter presents an extension of the approach that was described in chapter 4. The previous model's aim was to identify certain patterns and, moreover, to check whether these charts exhibit sufficient information to develop profitable trading rules.

Within the scope of the chapter at hand a different approach is pursued. The main goal is the forecasting of time series rather than the automated development of trading rules. SOMs are applied twice in order to compute probabilities on whose basis the prediction is accomplished. The model starts with a data transformation that is similiar to the one described in section 4.3. After the data preprocessing a SOM is used for computational ordering of the patterns and their respective grouping. This leads to a limited number of representative patterns that are interpreted as "local models", each standing for a certain constellation of market factors. Each pattern $y(t)$ is assigned to the local model which is its closest one in terms of the Euclidean distance. The unified patterns form matrices $\mathbf{y}_i$ $(i = 1, ..., N)$, where N is the total number of local models.

In the next step a slightly modified data set is used. The immediate subsequent value of each pattern $y(t)$ is taken into account and a SOM is used to classify the patterns within the group $\mathbf{y}_i$ into clusters of possible outcomes. Subsequently, empirical probabilities are computed by counting and used for determining the one-step ahead forecast in an out-of-sample set.

5.2 Data Preprocessing

The strategy that is adopted here looks into the past in order to forecast the next point in time, i.e. the underlying model is an autoregressive one as given in equation

(2.7).[1] Initially, there is a time series of historical prices, e.g. taken from the German Stock Exchange Index DAX. The series is embedded by means of the sliding-window device as explained in chapter 4.

It is a difficult task to determine the optimal order of autoregressive lags p. One possibility to select the correct number of backshifts was introduced in section 4.4. However, the proposed method has some drawbacks. Alternatively to the False-Nearest-Neighbour method, the selection of p can also be accomplished by trial and error. Different lag lengths are taken to train the SOM and the best outcome in terms of smallest root mean squared error is chosen to be best. The use of different lags for the estimation of the parameters can be subject-matter of a later sensitivity analysis.

As shown in section 4.3 the time series $\mathbf{x}$ of length T is reshaped into a new regressor matrix $\mathbf{y}$ of length $T - p + 1$. Note, there is a crucial difference between the model in the previous section and the one presented here. The goal of the model at hand is to extract probabilities from the clustered patterns which can be used for forecasting. In order to obtain the matrix of probabilities the transformation of vector $\mathbf{x}$ is accomplished twice.[2] The first turn is the embedding of the time series in order to generate patterns out of the raw data vector. This is to a large extent what has been done in the previous chapter.

The second time the sliding window is applied, the $(p + 1)$-th lag is taken into account, too. The idea behind this design is, that we know the *exact* next outcome of the time series for $T - p$ cases. We can then treat this vector of next outcomes as the vector of dependent variables and estimate a parameter set for later prediction.

Matrix (5.1) shows schematically the structure of $\mathbf{y}'$ after the transformation. It is worth noting the close relation between submatrix Γ and the matrix $\mathbf{y}$ that was introduced in section 4.3. The only difference between these two matrices is their dimension. When compared to each other, $\mathbf{y}$ embraces one row more than Γ. This is due to the fact that only for $T - p$ patterns the true next value is known. The subvector Υ is the extension generated by the double-embedding technique and can be interpreted as the combined vector of future realisations of every data constellation in this setting.

[1] In this chapter we include only lagged variables in our model and abstract from the presence of other independent variables.

[2] It will become clear in the next section why the employment of two embedding procedures, and therefore, the use of two different data sets is necessary.

$$\mathbf{y}' = \left(\underbrace{\begin{matrix} x_1 & x_2 & \cdots & x_p \\ x_2 & x_3 & \cdots & x_{p+1} \\ \vdots & & \ddots & \vdots \\ x_{T-p} & & \cdots & x_{T-1} \end{matrix}}_{\Gamma} \underbrace{\begin{matrix} x_{p+1} \\ x_{p+2} \\ \vdots \\ x_T \end{matrix}}_{\Upsilon} \right) \tag{5.1}$$

Under the assumption that a class of patterns represents a certain situation on the market and, moreover, under consideration that the appearances of these patterns are not singular events but rather a systematic recurrent formation, the next realisation of the time series *after* a pattern is observed provides valuable information for forecasting purposes.

5.3 Applying the SOM

The SOM method that is used in this work is built upon the matrix $\mathbf{y}'$. It is important to note the very close connection between these two regressor matrices. Matrix $\mathbf{y}'$ is an extension of $\mathbf{y}$ and covers exactly the same data, but additionally embraces the respective subsequent realised value of x after the appearance of a certain pattern. We will try to use this information in order to formulate probabilities which shall then be exploited for forecasting out-of-sample.

Once the Self-Organizing Map has grouped the sample vectors we deal with a set of N representative model vectors. Each sample vector belongs to one of the neurons on the map and we can use this data for the estimation of local models.

5.3.1 Double-SOM Application

Given the two regressor matrices $\mathbf{y}$ and $\mathbf{y}'$ we are able to present the data to the SOM method for grouping the data. First, we turn to the matrix $\mathbf{y}$ which is quantised by the SOM algorithm. The result of this procedure are M codebook vectors that represent similar patterns which tend to recur throughout the whole data set. In literature this procedure is known as *pattern recognition* and the SOM algorithm is a powerful tool to carry out this work. The codebook matrix will be of dimension $M \times p$. In the following the resulting map after convergence is called "IN" map because this part of the model can be considered as the input part. After employment of the Self-Organizing Map each row vector in matrix $\mathbf{y}$ is assigned to one of the M representative patterns, i.e. M data matrices can be formed by taking

$$\mathbf{z}_i = \mathbf{y}'_{c_i}, \quad \text{where} \quad c_i = \arg\min_i \{\|\mathbf{y} - \mathbf{m}_i\|\}. \tag{5.2}$$

This criterion was already used in the previous chapter (equation (4.12)). We are primarily interested in the index numbers c_i. These numbers reveal the members of the subgroups created by the SOM. The c's are taken to build matrices $\mathbf{z}_i$ ($i = 1, ..., M$) which are simply subsets of the matrix $\mathbf{y}'$ and embrace the vectors that are closest to the respective codebook vector m_i in terms of the Euclidean distance (Best-Matching-Unit Criterion).[3] The dimension of $\mathbf{y}$ matches with the codebook vectors m (column-wise). Thus, matrix $\mathbf{y}$ can be perceived as an auxiliary data set which is only necessary to determine the index vectors c. The size of $\mathbf{z}_i$ varies with i because the different model vectors do not always attract the same number of input vectors, i.e. not every node of the SOM net represents the same number of training patterns.[4] In other words, the distribution of input data over the SOM net is usually not equal.[5] For a more convenient notation, in the following the employment of the first SOM is denoted by the letter A.

The Self-Organizing Map is now applied a second time (abbreviated by B). It is interesting to analyse the distribution of realisations of the real next outcome of a pattern (subvector Υ) within the different groups $\mathbf{z}$. To this end, for each of the $A_1, \ldots, A_M$ groups identified in the first step a SOM is trained exclusively with the data belonging to a certain group A_i. This results in $B_1, \ldots, B_N$ model vectors for **each** of the M main groups. For every subgroup the criterion $c_j = \min_j \{\|\mathbf{y} - \mathbf{m}_j\|\}$ is employed again and the distribution is obtained by counting the "hits" or the number of responses a neuron exhibits, respectively.[6] Let β_j^i be the total number of patterns that belong to code vector j in subgroup A_i then the conditional empirical probability $p(B_j|A_i)$ for the next realisation after pattern j was observed can be computed by

$$p(B_j|A_i) = \frac{\beta_j^i}{\sum_j \beta_j^i}. \tag{5.3}$$

[3] The BMU takes a piece of data and compares it to the trained SOM net. Every codebook vector (or: weight vector) represents a group of most similar data from the original time series and, moreover, indicates the distance to its closest neighbours. After training the SOM net should provide a relatively good reconstruction of the topology of the underlying data set. The BMU criterion searches over the entire set of model vectors that one which minimises the vectorial distance between a given piece of data (in the case at hand a certain pattern) and the codebook entries in terms of the Euclidean distance. The result c indicates the "winning" neuron to which the vector x is assigned. We already know these equations from chapter 3 (c.f. equations (3.2), (3.7) and (3.8)) but there they appeared in a slightly different context.

[4] For instance, this is the case for a group of rare patterns. This analysis can be accomplished by means of the *Hit Histogram* (see section 3.3.2).

[5] Since the length of matrix $\mathbf{y}$ is larger than that of matrix $\mathbf{y}'$ by one row the last element of $\mathbf{y}'$ is not assignable by the proposed method. Since our data sets are always large enough we simply neglect this vector/pattern. This disregard does not change the results in any way.

[6] Note, that the index has changed. It is now necessary to deal with $j = 1, ..., N$ models because we are concentrating only on a certain subgroup.

That is, given a pattern is assigned to a class A_i it is possible to calculate the probability of the development of the next realisation of the time series. The construction of equation (5.3) ensures that the empirical probabilities are always in the interval $[0, 1]$. In the next section this probabilistic connection is contemplated in a more detailed manner.

5.3.2 Probabilistic Connection

The approach considered here is the implementation of a frequency matrix $\mathbf{T}(i, j)$. Given that some pattern or regressor, respectively, from $\mathbf{y}$ belongs to group i in the codebook generated by the first SOM application (A) the probability that the corresponding *output* vector $\mathbf{y}'$ belongs to the j-th class in the second SOM (B). The transition matrix $\mathbf{T}(i, j)$ is formed as

$$\mathbf{T}(i,j) = \begin{pmatrix} p(B_1|A_1) & \dots & p(B_N|A_1) \\ \vdots & \ddots & \vdots \\ p(B_1|A_M) & \dots & p(B_N|A_M) \end{pmatrix} \tag{5.4}$$

where

$$\sum_{j=1}^{M} p(B_j|A_i) = 1 \quad \forall \quad i. \tag{5.5}$$

The dimension of the probability matrix is determined by the number of codebook vectors generated by the two Self-Organizing Maps. Note that every row sums up to one. This is concordant with the basic rules of probability calculus. In a more formal way we can formulate the above matrix as the conditional probability that a certain pattern belongs to group j on the output panel given it is member of group i in the input space, or even shorter $p(y(t)'_j|y(t)_i)$.

We can now use these probabilities to predict the next value. For instance, a sequence of a time series of length six $\delta = [x(1), ..., x(6)]$ is presented to the first map and the closest codebook vector is found by employing the Best-Matching Unit Criterion (c.f. equation (3.2)). The sequence δ is assigned to the codebook vector that minimises the Euclidean distance. The "winning" codebook vector determines the group membership of δ. This means the Best-Matching Unit selects a group c among M possibilities. Once the group is chosen only the corresponding row in the transition table $\mathbf{T}(i, j)$ is of interest. The one-step ahead forecast is taken from a multinomial random drawing. The underlying empirical probability distribution is given by $\mathbf{T}(c, j)$ or $[p(B_1|A_c), ..., p(B_M|A_c)]$, respectively. In this example we can write down

$$\hat{x}(7) = \Upsilon_{c,\lambda}, \tag{5.6}$$

where λ is drawn randomly from a multinomial distribution depending on $\mathbf{T}$ within the model family c:

$$\lambda \sim MN\left(1, \mathbf{T}(c, j)\right). \tag{5.7}$$

Summarising, the one-step forecast is undertaken by the following procedure:

1. A formerly unknown data vector (of dimension $[1 \times p]$) is presented to the first SOM map A and assigned to one of the M possible models by means of the Best-Matching Unit Criterion (i.e. model family c).
2. A model λ is drawn randomly according to the probability distribution of the models $\lambda \sim MN\left(1, \mathbf{T}(c, j)\right)$ with $j = 1, \ldots, N$.
3. The one-step ahead forecast $\hat{x}$ is given by the value $\Upsilon_{c,\lambda}$.
4. Go back to step 1 and repeat the procedure until forecasting horizon is reached.

There is no guarantee that the value $\Upsilon_{c,\lambda}$ is the correct prediction after the appearance of a pattern but it is worth checking whether this method is capable to outperform orthodox forecasting methods.

5.3.3 Mean Returns

It was pointed out earlier that the SOM method requires normalised data for meaningful convergence. The patterns that are used for training are all restrained to be on the interval $[0; 1]$ in order to avoid a biasing influence of higher value patterns.[7] Therefore, the value that is used for forecasting ($\Upsilon_{c,\lambda}$ in the notation used here) has either to be translated back into "real" terms, i.e. *denormalised* values have to be applied, or an alternative calculation for the future development of the price time series has to be found.

There are several possible ways to account for this problem. Here, a return-based method is suggested. As a starting point, returns are simply defined as the first-differences of prices, i.e.

$$r_t = p_t - p_{t-1}, \tag{5.8}$$

where p_t is the logarithm of the stock price at time t and r_t denotes the respective return. Suppose, the SOM has been trained successfully and M families of patterns (A) each with N subgroups (B) are identified. The simplest thing to do would be to take the return $r_{p+1,j} = p_{p+1,j} - p_{p,j}$ from codebook vector j from the second SOM.[8] This value corresponds to a representative return that can be expected when

[7] Remember, the feature we are interested in is the geometrical characteristic of a pattern and not the quantitative one.

[8] By using the index p for the backshift parameter we stick to the notation introduced in section 4.

a pattern of family A_i is assigned to subgroup B_j by random drawing. The price prediction for $t+1$ can then be formulated as

$$\hat{p}_{t+1} = p_t + r_{p,B_j}. \tag{5.9}$$

Equation (5.9) describes a reasonable proxy for the price predictor but neglects many disposable information. There is a large number of patterns distributed over the respective subgroups in each family. The exact number of patterns depends on the first SOM application. From this vast amount of data it is possible to extract more exact returns than it is by employing equation (5.9). Hence, the mean over the returns belonging to a certain subgroup is calculated, i.e.

$$\bar{r}_{p,B_\lambda} = \frac{1}{\Lambda_{B_\lambda}} \sum_{k=1}^{\Lambda_{B_\lambda}} r_{\lambda,k}, \tag{5.10}$$

where Λ_{B_λ} is the total number of members of the chosen subgroup B_λ. These computations lead to a *mean return* matrix

$$\bar{\mathbf{r}}(i,j) = \begin{pmatrix} \bar{r}_{1,1} & \dots & \bar{r}_{1,N} \\ \vdots & \ddots & \vdots \\ \bar{r}_{M,1} & \dots & \bar{r}_{M,N} \end{pmatrix} \tag{5.11}$$

and therefore, equation (5.9) has to be modified as

$$\hat{p}_{t+1} = p_t + \bar{r}_{c,\lambda}. \tag{5.12}$$

One problem of this approach is the undifferentiated weighing of the subgroup members of B_λ. It was already shown in previous chapters that even if a pattern is assigned to a certain neuron on the SOM net, this is just a "best match" in relative terms, but does not make any statements on the absolute Euclidean distance. The BMU criterion assigns simply the pattern in question to the neuron that minimises the Euclidean. Among all the patterns attached to a neuron one might encounter huge differences in terms of distances to the "centre" of the node (i.e. the codebook vector). These differences of distances should be reflected in the calculation of the mean return matrix. Patterns of subgroup B_λ that lie further away from the codebook vector should have less influence on the mean return and vice versa. At this point of the analysis we forbear from taking these divergences into account because they are assumed to be only marginal and will not change the results much. Nevertheless, it is important to be aware of this constraint because it may play an important role when dealing with other research objectives.

5.4 Simulations and Results

5.4.1 Data

The data we are using is the New York Stock Exchange index between 01/02/1980 and 12/31/1999. We use daily data and the sample size is 5054 observations. Fig. 5.1 (a) and (b) show the trajectories of the prices and the respective returns over time.

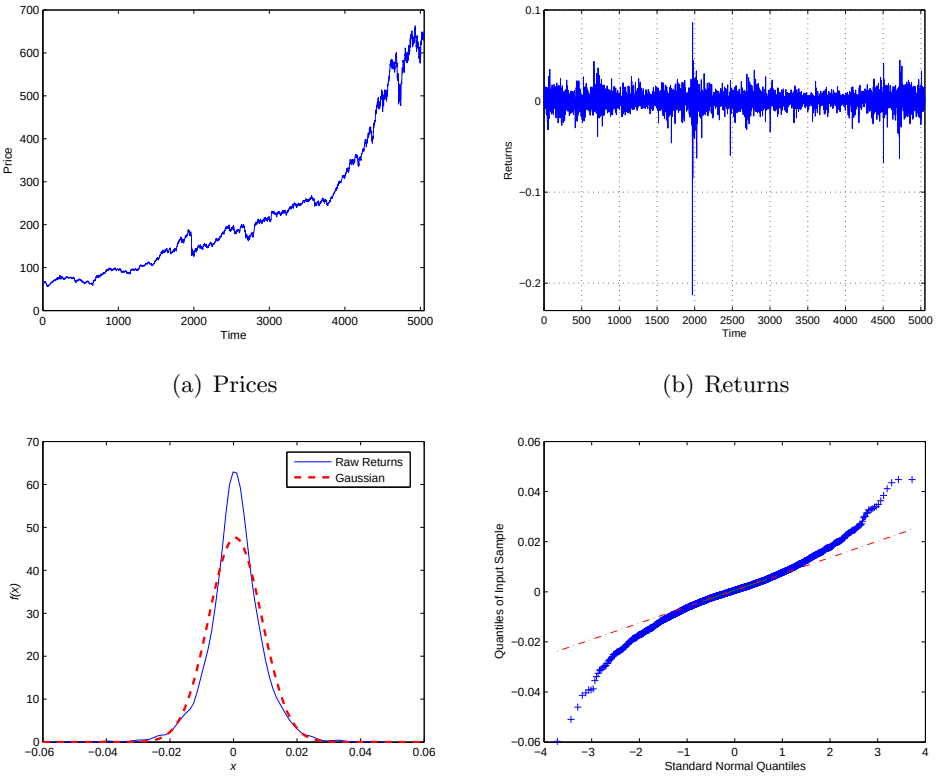
From visual inspection we see immediately the typical behaviour of financial time series. The prices are clearly nonstationary and the returns exhibit volatility clustering. From figure 5.1 (c) and (d) it becomes obvious that the raw returns are not normally distributed and deviate from the Gaussian Normal Distribution in terms of leptokurtosis. There is less probability mass in the "shoulders" of the distribution and more around the mean value and the tails. This *fat tail* behaviour becomes even more obvious when looking at the quantile-quantile-plot. The theoretical standard normal quantiles are plotted against the quantiles of the input sample. In figure 5.1 (d) the straight line corresponds to the normal distribution. The crosses refer to the input sample distribution. If the returns were normally distributed the crosses would be located exactly on the straight line. Since the return quantiles considerably defer from the straight line we can conclude fat tails in the returns.

The data that is used here exhibit the typical stylised facts of financial markets. The aim of the analysis at hand is the forecasting of prices rather than the prediction of volatility. Before the Self-Organizing Maps can be applied the data first have to be normalised between 0 and 1. This will preserve the information contained in the patterns and at the same time avoids to distort the training of the SOMs when the end of the price time series feeds much higher values into the algorithm than the very beginning. For prediction the values will be *denormalised* afterwards, i.e. the values obtained by our method are translated back into *real values*. The normalisation is accomplished by using equation (2.37).

The data set is split into two parts. The first subset embraces the observations from period one to period 4000 and the rest of the time series is used for later testing of the out-of-sample forecasting capabilities of the model (1051 observations).

5.4.2 Parameters

The parameter set that is used for the training of the map is summarised in table 5.1. For the first SOM, which separates the patterns into N families, the training process is segmented into the *rough training* and the *fine tuning*. Both sequences are restricted to 1000 iterations, but the rough training starts with a higher learning rate and a larger neighbourhood radius. The total number of observations of the training set is 4000 and after embedding of the time series 3979 patterns are obtained. Of course, the number of embedded patterns is determined by the backshift parameter.



(a) Prices

(b) Returns

(c) Distribution of Raw Returns

(d) QQ-Plot

Figure 5.1: New York Stock Exchange data from 01/02/1980 to 12/31/1999

The backshifts, in turn, are set to 22 because this size covers roughly a trading month when dealing with daily data. The neigbourhood function is given in Gaussian form and the size of the SOM net is restricted to consist of 36 nodes.

According to the number of families identified by application of the first SOM the second SOM is employed 36 times, each of which is also consisting of 36 nodes on the SOM net (symmetric architecture). This parameter is set arbitrarily and will be subject to later discussion. Within the families of patterns convergence of the map to its final state is not difficult to achieve because the group members are already fairly homogeneous. In order to keep the model simple the learning rate and the neighbourhood function are identical to the ones used in the first SOM. Furthermore, the length of the patterns in question are now 23 instead of 22, i.e. the total number of time series patterns is 3978 (this is the set of all patterns regardless of their respective family membership).

5.4.3 Forecasting

The objective of this section is to check for the one-step ahead prediction capabilities of the model out-of-sample. To this end, the procedure as described in section 5.3.2 is implemented and applied to the test data set. Table 5.1 summarises the main model parameters used for training of the map.

Figure 5.2 gives an overview of the one-step ahead forecasting results. On the left hand side the forecasted time series is opposed to the real values. This plot is given in an index-based scale and allows direct comparison of the deviations of the forecasting from the true trajectory of the prices. For convenience the forecasting residuals are presented in subplot (b). At a glance these errors seem to meet the presumption econometricians usually impose on forecasting errors. For a more rigorous treatment of the analysis several tests are conducted on basis of the error terms. All of the tests used in the following belong to the class of non-parametric tests because for the particular problem at hand it is not possible to make any distributional assumptions. Therefore, we are mainly interested in the performance of the forecasting compared to other approaches. The results of these tests are outlined in table 5.2.

In order to get a first quantitative evaluation of the prediction capability a simple non-parametric test of *correct directional change* (CDC) is applied. For calculating the CDC statistic we first compute the first-difference between two consecutive observations in the forecasted and the observed time series. Let these variables be denoted by δ_t and δ'_t, respectively. Formally, this can be expressed as

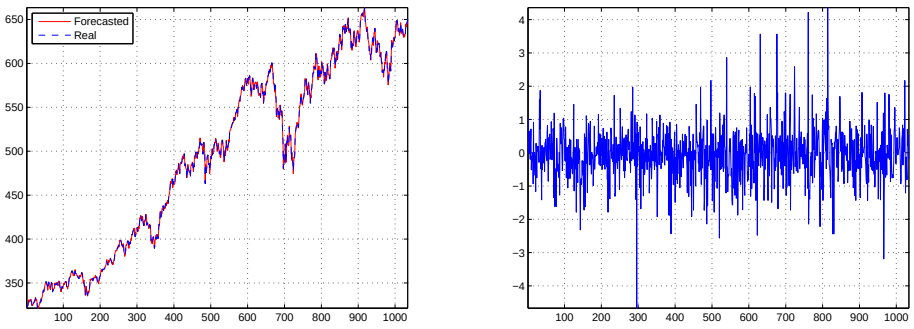
$$\delta_t = x_{t+1} - x_t \quad \text{and} \tag{5.13}$$

$$\hat{\delta}_t = \hat{x}_{t+1} - \hat{x}_t, \tag{5.14}$$

where the hats indicate the forecasted value of the time series. Then a indicator

Parameter	Value
Number of Iterations Rough Training	1000
Number of Iterations Fine Tuning	1000
Number of neurons (Input Map)	36
Number of neurons (Output Map)	36
Backshift Parameter	22
Initial Radius Rough Training	6
Final Radius Rough Training	1
Initial Radius Fine Tuning	1
Final Radius Fine Tuning	0.1
Initial Learning Rate Rough Training	1
Initial Learning Rate Fine Tuning	0.1
Total NOBs	4000
Number of Time Series Patterns	3979

Table 5.1: Parameter set-up for probabilistic model



(a) Forecasted vs. real values

(b) One-step forecasting errors

Figure 5.2: One-step ahead forecasting of NYSE

variable is formed based on the δ's calculated above and adopt the following values:

$$\zeta_t = \begin{cases} 0 & \text{if} \quad \delta_t \cdot \hat{\delta}_t \leq 0 \\ 1 & \text{else} \end{cases} . \tag{5.15}$$

The CDC statistic is then simply the percentage of correct directional change of the forecasting computed by

$$CDC = \frac{100}{T} \sum_{t=1}^{T} \zeta_t \tag{5.16}$$

with T as the length of the forecast horizon. In the example presented here over 92% of the directional changes of the process are explained correctly.

An usual instrument to determine the prediction power of a method is to compare its root mean squared error (RMSE) with that of a benchmark model (usually the simple random walk).[9] The RMSE of the forecasting method proposed here is considerably lower than the one generated by the random walk.[10] One reason for that could be that the SOM method draws the prediction for the next time step from an empirical distribution that is based on local modelling of the process. The models are assumed to be locally smooth which leads to smaller jumps compared to the random walk that uses independently drawn random numbers for forecasting the one-step ahead point prediction.

Furthermore, the *mean absolute percentage error* (MAPE) diagnostic is employed. The MAPE is computed by

$$MAPE = \frac{100}{T} \sum_{t=1}^{T} \frac{\parallel (x_t - \hat{x}_t) \parallel}{x_t}. \tag{5.17}$$

The MAPE of the SOM based model is with approximately 11% very low and performs much better than the benchmark random walk ($\approx$ 74%). This confirms the findings from the RMSE analysis.

For the Diebold-Mariano Test (DM-Test), again, the random walk is taken as the benchmark model. The idea of the test is to check for the goodness of forecasting performances based on the direct comparison of the two models. Please see appendix to this chapter for a detailed description of the test procedure (A 5.1). The test statistic for the DM-Test computed here is below the critical value -1.69. Hence, the null hypothesis of equal forecasting errors of the benchmark model and the SOM based model can be rejected. Obviously, the errors are significantly lower in the case of the proposed model.

[9] The RMSE was calculated as $\sqrt{\frac{1}{T}\sum_{t=1}^{T}(x_t - \bar{x})^2}$.

[10] The random walk used for comparison is given by $\hat{P}_{t+1} = P_t + \epsilon_t$ with $\epsilon_t \sim N(0,1)$.

The Pesaran-Timmermann Test is constructed in the spirit of the simple CDC test statistic (sign test), but enables the statistical evaluation of the result. It is a non-parametric, distribution-free test that has a large degree of inherent flexibility (see appendix A 5.2 for further comments on the test procedure). The null hypothesis states that the signs of the actual time series and the signs generated by the forecasting model are distributed independently. In this case the null can clearly be rejected at a 5% significance level.[11]

Finally, the *encompassing test* as used in Darrat and Zhong (2000) is conducted. The idea of the test is to judge if one out of two forecasting models significantly outperforms (*encompasses*) the other one. The test is based on two regression equations, where the dependent variable is given by the forcasting errors:

$$(\hat{x}_{jt} - x_t) = \beta_{jk}\hat{x}_{kt} + \epsilon_t \quad \text{and} \tag{5.18}$$

$$(\hat{x}_{kt} - x_t) = \gamma_{jk}\hat{x}_{jt} + \mu_t \tag{5.19}$$

with x_t as the true realisation of the forecasted time series at t and $\hat{x}_{it}$ as the forecasted value by model $i = \{j, k\}$. The error terms are represented by ϵ and μ, and β and γ are the parameters to be estimated. The null hypothesis presumes that neither of the two models j and k outperforms the other one, i.e. if both β **and** γ are not significant **or** both are significant, then the null cannot be rejected. However, if one of the parameters is significantly different from zero and the other is not then one of the model significantly explains the forecasting errors of the other model, i.e. model k encompasses model j. Applying this method to both models it is found that the p-value for the SOM-based approach is $p = 0.6149344$ and for the random walk model we find $p = 0.5007591$. None of the parameters is significant and therefore neither the SOM model is capable to encompass the random walk method nor vice versa. The null hypothesis cannot be rejected. None of the models outperforms the other one.

Summarising, the test results presented above support the application of the proposed SOM-based method. One problem with most of the testing procedures used for the evaluation of the goodness of the model is the dependence on an adequate benchmark model. It is standard in literature to consult the basic random walk in these cases. Since the test statistics of e.g. the DM-Test rely on the comparison of the random walk realisations and the model outcomes the final results have to be interpreted critically and carefully.

[11] In order to conduct these tests correctly it was necessary to transform the real and the forecasted price series into returns. The Pesaran-Timmermann Test works only with stationary data and focuses on the directional change of the predictions.

Test	Value
RMSE	0.7982750
RMSE Random-Walk	5.207134
MAPE	0.1121615
MAPE Random-Walk	0.7408273
Correct Directional Change (CDC)	0.9213483
Pesaran-Timmermann Test	26.41218
Diebold-Mariano Test	-1.717126
Jarque-Bera Test	79.66192
Ljung-Box Test (20 lags)	18437.84
(critical value)	(31.41)
Durbin-Watson (forecast errors)	1.892363
Encompassing Test (p-value – SOM regressed on RW)	0.6149344
Encompassing Test (p-value – RW regressed on SOM)	0.5007591

Table 5.2: Tests for forecasting capability

5.4.4 Monte Carlo Simulation

In the previous subsections we have only considered a "one-shot" experiment, i.e. the derivation of the forecasts relied on a single round of random drawings from the underlying empirical distribution function. Due to the nature of the method it is not guaranteed that the best one-step prediction is selected by the software. In our set-up there are 36 alternatives and the computational algorithm has to choose among them. This leads to a varying goodness of forecasting each time the model is employed.

In order to account for this problem we conduct a Monte-Carlo-Simulation. This method will help to obtain reliable figures on the performance of the method. For the out-of-sample forecast we used a time series of 1000 observations and employed the SOM model in an iterative manner for obtaining one-step point predictions (note that the backshift parameter was set to 22, which leads to a forecasting horizon of $T = 979$). The Monte-Carlo-Simulation was accomplished for three different cases: 2000, 5000 and 10000 runs.

In figure 5.3 the simulation results for the case of 2000 runs are plotted. Each point prediction was computed 2000 times on the basis of the empirical distribution as shown in equation (5.4). The red points in the graph represent the scatter of the forecasts around the true value (indicated by the blue line). Subplot 5.3 (b) is an arbitrarily chosen section from the entire forecasting horizon and includes 50 observations. This magnification serves to highlight the details of figure 5.3 (b). In the two plots we can see that there is a high concentration of prediction values around the actual time series which is subject to prediction. There are only single outliers further away from the blue line which indicates a relatively good fit. The

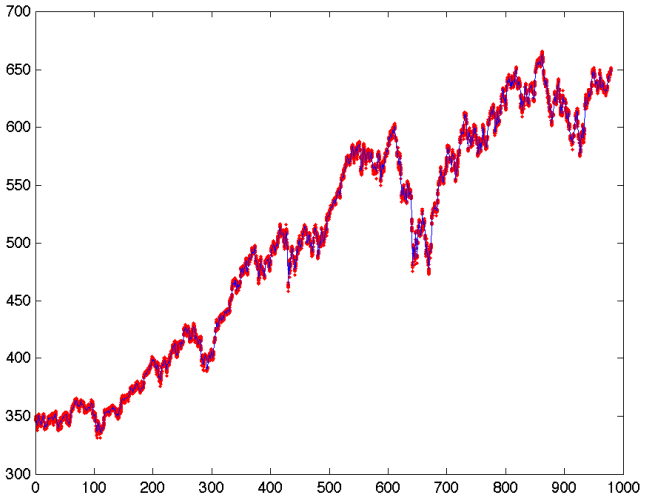
distribution of the forecasts around one sample period depends on the underlying empirical distribution and the appearance frequencies of a certain forecast perfectly resemble this empirical distribution. This has to be the case because the selection of the next "true" realisation of the time series depends completely on the distribution **T** (see equation (5.4)).

Figures 5.3 (a) and (b) serve only as a first overview of the forecasting behaviour of the model. Table 5.3 summarises the results of the Monte-Carlo-Simulation for selected sample periods in a more detailed manner. Unfortunately, it is not possible to give a comprehensive overview for the three experiments of $N = \{2000, 5000, 10000\}$ runs for the entire forecasting horizon. Therefore, we decided to show exemplarily the results of the simulations for six different arbitrarily chosen sample periods, namely $t = \{1, 200, 400, 600, 800, 950\}$. These snapshots from the time series cover the entire forecasting horizon and should represent the overall characteristics of the Monte-Carlo-Simulation well. Moreover, the average over the respective key figures are given in the last row of each panel. This number should help to get an idea about the development of the SOM method for varying number of runs in the Monte-Carlo-Simulation.

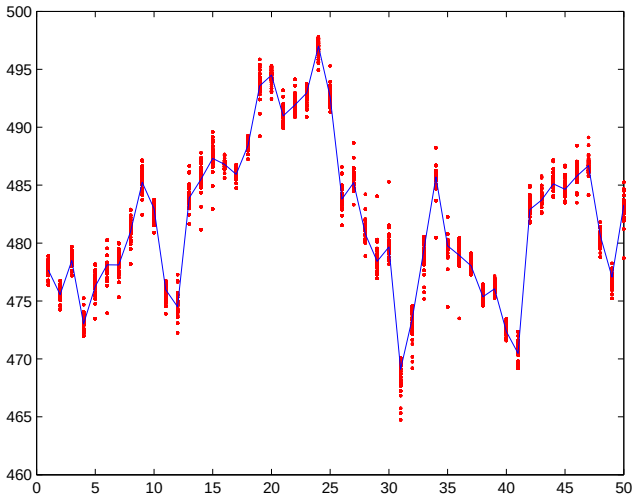
The general picture shown by the numbers in table 5.3 demonstrates a relatively stable forecasting of the SOM probabilistic approach. The means of the different point predictions do not vary much for different simulation lengths N. Most of the deviations occur only in the second decimal place. The standard deviation for the realisations in a certain sample period t for a given simulation length N is defined as

$$\sigma_{t|N} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\hat{x}_{i,t} - \hat{\bar{x}}_t)^2}, \tag{5.20}$$

with $\hat{x}_{i,t}$ as the i-th prediction obtained by repeated forecasting for time t and $\hat{\bar{x}}_t$ as the mean of these values. For different simulation lengths N this value fluctuates within a small interval. This was expected ex-ante because of the underlying drawing of one-step predictions. Due to the construction of the forecasting scheme it is not necessarily the case that these values converge with increasing N. However, it is important that the standard deviations do not depart too much from each other, which certainly holds true in the case at hand. The last row of each panel gives the average over all sample periods contained in the data set (in our case 979 observations). The overall average standard deviation decreases with increasing simulations, which is a satisfying and expected result. The fifth column of table 5.3 give the True Value Deviation, which is defined as



(a) Complete forecasting horizon



(b) Snapshot of 50 sample periods

Figure 5.3: Monte-Carlo-Simulation of 2000 runs plotted against true time series

$$\lambda_{t|N} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\hat{x}_{i,t} - x_t)^2}. \tag{5.21}$$

Here, the variables are as defined above and x_t is the real observation at time t. Again, we focus on the average deviation of the prediction from the true realisation of the time series *within* the respective sample periods t. The test results draw a similar picture like the standard deviations we analysed before. The numbers stay within a certain range, not deviating too much from their counterparts of the alternative simulations. It is worth emphasising that the $\bar{\lambda}_N$ are decreasing with increasing N. Even though it is not entirely clear from the sample periods we chose, the mean value over all observations shows that the overall performance improves the greater the number of simulation runs is.

Finally, we also take a closer look at the Absolute Error. This figure is closely related to the True Value Deviation and should therefore exhibit similar features as the previous analysis. It is defined as

$$\delta_{t|N} = \frac{1}{N}\sum_{i=1}^{N} \parallel \hat{x}_{i,t} - x_t \parallel \tag{5.22}$$

with the usual notation as it was used before. In fact, the Absolute Error confirms the findings from the True Value Deviation. All the numbers develop in the same direction as the $\lambda_{t|N}$ and the averages behave accordingly.

The results summarised in table 5.3 mainly confirm what was found in the "one-shot-prediction" of the previous subsection (see 5.4.3). Of course, the user of the method has to be aware of the fact that there is no such a thing as an *optimal* predictor. The one-step forecasts should be rather interpreted in terms of a *likely* development of the time series. In the framework of the analysis of this subsection we also experimented with histograms of the respective scatter around a value which is to be forecasted. The resulting graphs are not presented here because they exactly resemble the empirical distribution given by equation (5.4). By construction of the probabilistic SOM method this has to be the case.

In the framework of the Monte-Carlo analysis it is also interesting to look at the distribution of the criteria we used to assess the goodness of the proposed method. For this purpose we calculated the respective RMSE, MAPE, and CDC for 2000, 5000, and 10000 simulation runs, respectively. The results are summarised in table 5.4.

Even though the method relies on the random drawing of forecasting values the numbers given in table 5.4 are very stable do not change much with a varying number of runs in the Monte-Carlo-Simulation. This first result holds for each of the three selected criteria. Furthermore, the variance of the respective simulations is very low

Number of Simulations (N)	Sample Period (t)	Mean	Standard Deviation	True Value Deviation	Absolute Error
2000	1	347.66	0.5093	0.0116	0.4293
	200	397.35	0.5116	0.0118	0.4310
	400	482.40	0.4469	0.0110	0.3512
	600	585.48	0.6207	0.0141	0.4901
	800	615.81	0.5949	0.0134	0.4278
	950	649.11	1.2566	0.0285	0.7025
	Avg.	–	0.6567	0.0081	0.2702
5000	1	347.64	0.4955	0.0072	0.4198
	200	397.35	0.5042	0.0074	0.4274
	400	482.39	0.4542	0.0071	0.3624
	600	585.49	0.6171	0.0089	0.4896
	800	615.81	0.5755	0.0082	0.4212
	950	649.13	1.2024	0.0173	0.6989
	Avg.	–	0.6415	0.0021	0.1080
10000	1	347.64	0.5084	0.0052	0.4294
	200	397.36	0.4990	0.0051	0.4234
	400	482.39	0.4493	0.0050	0.3615
	600	585.49	0.6033	0.0061	0.4749
	800	615.79	0.5985	0.0061	0.4343
	950	649.10	1.1929	0.0121	0.6946
	Avg.	–	0.6419	0.0007	0.0540

Table 5.3: Results of Monte-Carlo-Simulation for selected sample periods (values rounded)

	Number of runs		
Criterion	**2000**	**5000**	**10000**
Mean RMSE	0.77945	0.77923	0.77874
Variance RMSE	0.0010761	0.001028	0.0010352
Mean MAPE	0.11566	0.11557	0.11553
Variance MAPE	1.4333e-05	1.3467e-05	1.3179e-05
Mean CDC	0.69983	0.70197	0.69908
Variance CDC	0.0061572	0.0058002	0.0059146

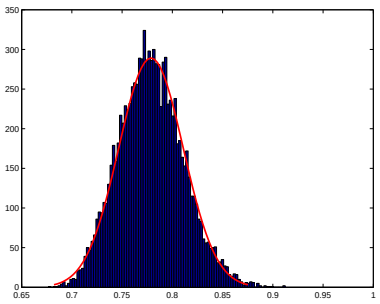
Table 5.4: Monte-Carlo-Simulation of test criteria

in all cases. Comparing the mean values of the simulation with the results given in table 5.2 shows that the criteria obtained from the estimation in the previous subsection are not only a one-shot outcome, but they are also very close to the Monte-Carlo-Simulation values as well. Of course, due to the construction of the method differences between the measures of different runs are possible. This can be seen best when considering the CDC criterion. In table 5.4 the mean value of the distribution is $\approx 70\%$, but in our estimation we reach a CDC of about $\approx 92\%$. This is an extremely high value which cannot be confirmed empirically, i.e. by using this method it is important to keep the possibility of deviations from the empirical mean value of the criteria in mind.

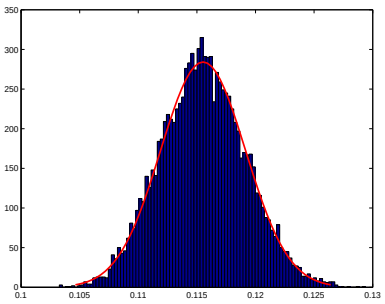
Figure 5.4 underpins the aforementioned findings. Here, the histograms for the case of 10000 Monte-Carlo-Simulation runs are plotted (blue bars) and for convenience the fitted normal density is superimposed (red line). There is no need to present the histograms for 2000 or 5000 runs, respectively, since we have already seen that the distributions are stable and do not change much when increasing or decreasing the length of simulation. The RMSE, the MAPE, and the CDC exhibit an almost normal distribution over the entire set of runs. Especially for subfigure 5.4 (c) it is important to note that the probability mass ranges from 0.4 to almost 1.0. In other words, it is possible (even though not very likely) that the performance of the SOM-based method in terms of the CDC criterion drops under the random-walk's performance.

5.4.5 Other Data Sets

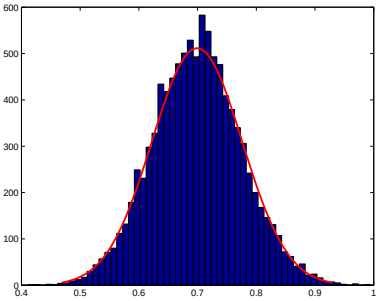
In order to check the method's performance when applied to other time series we tested the SOM with two other data sets. Because an index time series was used in the previous section we will now turn to the analysis of two commodity prices: gold and crude oil. As one of the most important precious metals gold has a very long tradition on international markets and is now quoted for many decades on a daily basis. Gold has almost always been considered as a backup for monetary systems of modern economies and is still a favoured investment in times of an uncertain future on (unstable) markets. Due to its physical character and its inherent natural value the metal is widely perceived as a secure and durable alternative to investments in other products traded on financial markets. The time series we are using is extracted from *Thomson Financial Data* (DATASTREAM) and includes 10310 observations starting on 02/01/68 until 10/07/07. The prices are Gold Bullion US-$/Troy Ounce. The second data set are crude oil prices for the period 12/03/91 until 10/07/07 (4261 observations) quoted as Brent UK Close US-$/Barrel. It is not necessary to point out explicitly the importance of oil for the global economy. Up to the present no real alternative to fossil burning has been found and the resources are becoming ever scarcer. Hence, the crude oil price will rather tend to increase than to fall in



(a) RMSE



(b) MAPE



(c) CDC

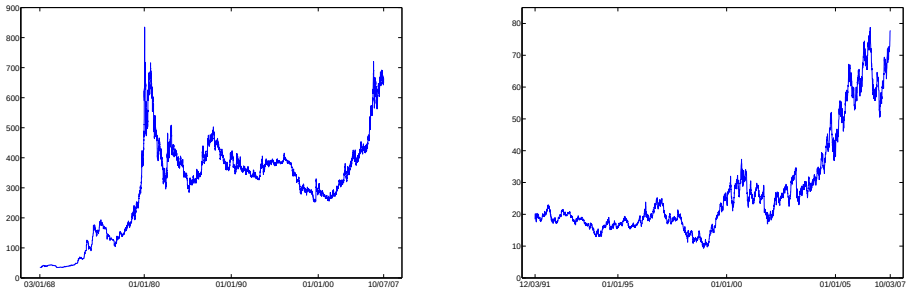
Figure 5.4: Distributions of RMSE, MAPE, and CDC computed with Monte-Carlo-Simulation (10000 runs) and superimposed fitted normal density (red line)

the future. Figure 5.5 shows the respective trajectory of both time series.

In order to assure comparability with the analysis of the previous section we use exactly the same parameter set-up as given in table 5.1. Of course, the respective length of the training data changes according to the total length of the series in question, i.e. in all the cases we leave the last 1000 observations of the respective time series for forecasting purposes and use the rest to train the SOM.

Table 5.5 gives an overview of the most important results. As in the NYSE case the criteria RMSE, MAPE, and CDC indicate a relatively good performance of the method. The CDC is – again – in both cases over 80% (for the gold prices the CDC reaches a value of even over 87%) which is a satisfying result (from the random walk one would expect a correct forecasting of the directional change around 50%).

For both cases the null of the Pesaran-Timmermann Test can be rejected at a 5% level, i.e. the hypothesis of independence between the forecasting signs of the true time series and the signs generated by the forecasting model does not hold. It is also possible to reject the null hypothesis of equal forecasting errors of the SOM model and the random walk benchmark by the DM-Test (test statistics are below -1.69 in both cases). This means that the forecasting errors generated by the SOM are significantly lower that those generated by the random walk. This underpins the results reported in section 5.4.3. The encompassing test indicates that none of the models encompasses the other one. This result holds for the Gold prices as well as for the Crude Oil prices. Generally, the test results in this section draw the same picture of the SOM-based method's performance as in the previous sections. Therefore, it does not seem to matter for which time series the SOM is used.



(a) Gold (03/01/68-10/07/07) (b) Crude oil (12/03/91-10/07/07)

Figure 5.5: Price time series of gold and crude oil

Test	Gold	Crude Oil
RMSE	1.736155	0.2672436
RMSE Random-Walk	5.935562	1.390614
MAPE	0.24221	0.4007720
MAPE Random-Walk	0.78328	2.165761
Correct Directional Change (CDC)	0.8018386	0.8743616
Pesaran-Timmermann Test	18.90029	23.43953
Diebold-Mariano Test	-1.747078	-8.430992
Jarque-Bera Test	117.6927	57.19573
Ljung-Box Test (20 lags)	18389.63	17476.64
(critical value)	(31.41)	(31.41)
Durbin-Watson (forecast errors)	1.925438	1.975548
Encompassing Test (p-value – SOM regressed on RW)	0.08658	0.33651
Encompassing Test (p-value – RW regressed on SOM)	0.22586	0.25470

Table 5.5: Test statistics for gold and crude oil

5.5 Conditional Expectation Values

So far, the forecasting procedure was based on the extraction of one-step point predictions from stochastic realisations of a special Markov-Switching-Process. This approach is subject to criticism because even though the forecasting depends on the frequency of historical observations it is not guaranteed that a random drawing of single point predictions leads to reliable forecasting. In contrast, one could argue that the random drawing of return values, which are then taken for computing the price prediction, "de-couples" the forecasting from reality because it excludes all the other possible realisations of the data generating process that were observed in the past as well.

In order to account for this problem we suggest an alternative approach[12]: instead of trusting just one single return value drawn from a pool of possibilities we now calculate the conditional expectation values for each of the model families, i.e. every possible outcome within a model family A_i $(i = 1, \ldots, N)$ is taken into account and weighed by its respective empirical conditional probabilities $p(B_j|A_i)$. More formally the calculation of the prediction can be written down as

$$\hat{p}_{t+1} = p_t + \mathsf{E}\left[r|A_i\right] = p_t + \sum_{j=1}^{M} p(B_j|A_i) r_{B_j|A_i}. \tag{5.23}$$

We stick to the notation used throughout this chapter. The weights $p(B_j|A_i)$ are simply the empirical probabilities as computed in equation (5.3), and $r_{B_j|A_i}$ stands for the respective returns of subfamily j within model group i. The weighed sum of

[12]Henceforth the **C**onditional **E**xpectation **V**alue Method will be abbreviated by CEV.

returns should account for a larger set of information than the random drawing of a single return is capable to do. In the remainder of this section we will have a closer look at the forecasting performance of this modification.

In order to assure comparability of the methods the same data sets and identical testing procedures are applied the problem at hand. We also avoid to change any of the (*hyper*-)parameters given in table 5.1.

The comparison between the test statistics of the previous variant of estimation and the CEV approach described in this subsection leads to interesting findings. First of all, the application of the CEV reduces the RMSE for the cases NYSE and Gold, but deteriorates the result for Crude Oil significantly. On the other hand, the absolute errors (MAPE criterion) improve for all three time series compared to the previous method. Another systematic improvement is achieved under consideration of the CDC criterion. The correct one step sign prediction of the method adopts very high values and exceed even the numbers given in tables 5.2 and 5.5.

The other test statistics exhibit very similar qualitative results. The MAPE of the random walk is always worse than those of the respective CEV counterparts in all three data sets. The results of the Diebold-Mariano Test are ambiguous because the null of equal forecasting errors in the benchmark and the alternative model can be rejected at a 5% significance level in the NYSE case only. The Pesaran-Timmermann null hypothesis that the sign distributions of the true time series and the forecasted values are independent can clearly be rejected. Following the Jarque-Bera Test the error structure clearly deviates from normality and indicates that the proposed method was not able to filter every information from the time series (and therefore not leaving the error to be only white noise). The Ljung-Box Test emphasises this result by rejecting the null hypothesis of no serial correlation in the the forecasting errors. Applying the Durbin-Watson Test to the forecast errors supports the Ljung-Box Test and cannot reject the null of no first-order serial autocorrelation in the errors. Lastly, the Encompassing Test results in an acceptance of the hypothesis that none of the models explains the forecasting error of the respective alternative model.

On the basis of these results we can conclude that the employment of conditional expectation values leads to improved results. Even though we were not able to achieve an improvement of the RMSE in the case of Crude Oil the general conclusion on the basis of the test criteria are in favour of the CEV-based forecasting. Generally, this variant of prediction leads to a smoothing in the overall forecasting process and therefore to a less volatile prediction. Due to the consideration of all possible outcomes within one model family a sudden jump of the forecasting is prevented. Such a discrete jump is more likely to occur in the random drawing set-up because through a random drawing of the prediction the entire process neglects the part of information that is hidden in the other values. For this reason it seems to be more

Test	NYSE	Gold	Crude Oil
RMSE	0.2119518	0.5156761	0.6120248
RMSE Random-Walk	5.207134	5.935562	1.390614
MAPE	0.03459565	0.07840988	0.1048537
MAPE Random-Walk	0.7408273	0.7832811	2.165761
Correct Directional Change (CDC)	0.965864	0.9264556	0.9601634
Pesaran-Timmermann Test	30.51925	26.74844	28.84322
Diebold-Mariano Test	-1.941865	-1.575055	-0.6474953
Jarque-Bera Test	79.69476	117.3288	57.50362
Ljung-Box Test (20 lags)	18438.61	18400.30	17496.62
(critical value)	(31.41)	(31.41)	(31.41)
Durbin-Watson (forecast errors)	0.7530737	0.9951939	0.7081382
Encompassing Test (p-value – SOM regressed on RW)	0.4499127	0.8315208	0.98699
Encompassing Test (p-value – RW regressed on SOM)	0.5090993	0.2262735	0.2547508

Table 5.6: Test statistics for three data sets calculated with CEV

natural to apply the CEV to time series prediction problems.

5.6 Chapter Summary

In the framework of the model developed in this chapter we derived the class affiliation of the patterns with the help of a SOM. As it became obvious the distribution of hits over the neurons was not equal, i.e. not every neuron unified the same number of geometric symbols. Since we applied a second SOM *within* the respective groups in order to achieve a further grouping from which we derived the empirical probabilities, it is possible to encounter numerical problems during the procedure. When the input data set is small and/or the predetermined number of neurons is too large there might be too few members in some model families in order to compute reliable probabilities. In such a case the researcher would either have to reduce the size of the net or augment the data set.

Moreover, it is doubtful whether the determination of the next price via randomly drawn returns is a trustworthy method. Indeed, the empirical probability does give a certain evidence for the distribution of the return's future development, but there is – as it is often the case when working with any form of neural networks – a large set of *hyperparameters* that have to be set a priori (such as train map size etc.). These parameters have a wide influence on the final convergence of the net and have to be chosen carefully under consideration of the problem at hand. Hence, the researcher's

experience is of crucial importance in this respect. Drawing "representative" returns from a multinomial distribution does inherently lead to a remaining uncertainty and may drive the forecasting into the wrong direction. Additionally, the discrete distribution of the possible outcome may lead to jumps in the forecasting which do not exist in the real time series.

Some experiments we undertook indicated that the method is only applicable for a one-step forecasting horizon. Everything that goes beyond a one-period prediction led to an abrupt decrease in the out-of-sample goodness of fit. This suggests a certain weakness of the model's capability to cover price series globally. Eventually, it depends on the immediate previous observations for computing a reasonable next predictive value. It would be nice to refine the method in such a way that longer prediction horizons were covered more adequately.

This chapter provided a probabilistic extension of the SOM method. The algorithm was employed twice. First, it was used to build families of models and in the next step it was applied for *intra-family* grouping of the sub-models. After this twofold application it was a simple task to develop empirical probabilities by counting.

In a next step the *mean returns* of each submodel were calculated in order to account for the back translation from the normalised into the non-normalised world. These mean returns were then used for forecasting that was accomplished on the basis of a random drawing from the distribution given by the empirical probabilities. As an alternative variant of the forecasting procedure we tried to extract predictions from the conditional expectations *within* a family of models. To this end we attached to each forecasting value in a certain model group a weight (the respective empirical probability) and used the sum of the weighted returns as a one step prediction.

The methods were tested with the NYSE, the Gold and the Crude Oil price series. A variety of (non-parametric) tests was accomplished which led to promising results. There are certain indications that the implicit assumption of an improved economic time series modelling on the basis of several different models rather than just one global formulation seems to hold true. In the next chapter we will exploit this idea, but abandon the influence of empirical probabilities in order to ensure a clear-cut interpretation of the actual power of *local modelling*.

The models presented in this chapter were designed in the spirit of the work by Blayo et al. (2003) and Cottrell et al. (1998). Blayo et al. (2003) were the first to develop a forecasting system which is based on the idea that the SOM can be applied twice to an embedded data set. They also pre-process a time series in order to obtain two different data matrices (in the style of matrix (5.1)) and group each of these sets with a SOM. We pursue the same approach but the difference comes into play when the sub-families (the B_j's) are analysed. In our model we are focussing on the geometrical characteristics of the patterns identified through

the SOM. The aim of the work is to find recurrent patterns that reflect a certain likelihood of future development of the time series in question. Blayo et al. (2003) are rather looking for a functional relationship in the sub-families and try to identify these by means of *Radial Basis Function Networks* (RBFN). Our approach connects the work of Cottrell et al. (1998) (random drawing of predictions) with the data pre-processing of Blayo et al. (2003). In section 5.5 we deviate from the original forecasting procedure and adopt a technique that is based on conditional expectation values. This methodology can also be found in Blayo et al. (2003), but they use the results of the RBFNs than the true next realisations of the time series. Furthermore, our scope of research is the prediction of prices, whereas Blayo et al. (2003) are attempting to forecast returns.

Appendix to Chapter 5

A 5.1 Diebold-Mariano Test

Original Version

The Diebold-Mariano Test for out-of-sample errors is a multistep testing procedure which was originally developed in order to determine if the model at hand performs significantly better than a benchmark model in terms of the out-of-sample fit.[13]

Suppose we have two models that predict the changes of a time series (usually, for financial market applications this is some model of interest which is tested against the random walk). Assume that

$$e_{it} = y_t - \hat{y}_{it} \quad \text{for model} \quad i = 1, 2. \tag{A 5.1}$$

Here, y_t is the true value of the time series at time t and $\hat{y}_{it}$ is the respective out-of-sample prediction of model i. The errors are then defined as the difference of the forecasted values and the true realisation. For the construction of the Diebold-Mariano Test it is convenient to take the absolute differences

$$z_\tau = |e_{1\tau}| - |e_{2\tau}|, \tag{A 5.2}$$

where τ is a point in time within the forecasting interval. The next step consists of taking the mean of z over the entire forecasting horizon:

$$\bar{z} = \frac{\sum_{\tau=1}^{\tau^\star} z_\tau}{\tau^\star}. \tag{A 5.3}$$

Once the mean over the absolute differences is calculated the *covariogram* is needed as a second component for the Diebold-Mariano test statistic. The covariogram is defined as

$$c = [\mathsf{COV}[z_\tau, z_{\tau-p}], \mathsf{COV}[z_\tau, z_\tau], \mathsf{COV}[z_\tau, z_{\tau+p}]] \tag{A 5.4}$$

with $p < \tau^\star$ as the length of the out-of-sample forecasting errors. The covariogram accounts for the correction for serial correlation in the differential of absolute errors. Taking the mean of the covariogram

$$\bar{c} = \frac{\sum c}{(p+1)} \tag{A 5.5}$$

leads to the Diebold-Mariano test statistic

$$DM = \frac{\bar{z}}{\bar{c}} \overset{a}{\sim} N(0,1),\ H_0 : \mathsf{E}[z_\tau] = 0. \tag{A 5.6}$$

Under the null there are no significant differences in the predictive accuracy of the two models. Given the case that the forecasting errors of the competing model are significantly lower than the benchmark model's errors the DM statistic adopt values **below** the 5% critical value of -1.69.

[13] Diebold and Mariano (1995)

Modified Version

One of the problems of the original Diebold-Mariano Test is that it tends to compute too large test statistic values when only a small sample size is available. This effect is due to the biased variance estimator in these cases. In order to avoid this problem Harvey et al. (1997) modified the Diebold-Mariano Test by augmenting the test statistic by a correcting factor. This correction accounts for "fat tails" in the distribution of the forecasting error terms.

The correction is accomplished by multiplying equation (A 5.6) with a term

$$K = \frac{\tau^\star + 1 - 2p + p(1-p)/\tau^\star}{\tau^\star} \tag{A 5.7}$$

i.e. the *Modified Diebold-Mariano* (MDM) test statistic is

$$MDM = K \cdot DM \overset{a}{\sim} t_{\tau^\star - 1}(0,1). \tag{A 5.8}$$

The test statistic MDM is asymptotically student-t distributed with $\sigma^2 = 1$ around $\mu = 0$, and $(\tau^\star - 1)$ degrees of freedom.

A 5.2 Pesaran-Timmermann Test

The Pesaran-Timmermann Test[14] is a simple distribution-free nonparametric test of predictive performance. The test is eligible for checking the significance of the correct directional change of the predictions. This is especially of interest if the underlying probability distribution of the forecasts is hard to derive analytically.

For our purposes it will suffice to present the test for the 2×2 case (a detailed description of the $m \times m$ case can be found in Pesaran and Timmermann (1992)). We start with a definition of the predictor of y_t. This predictor is based on the information set Ω available at time $t-1$, i.e. formally we can write down

$$x_t = \hat{\mathsf{E}}\left[y_t | \Omega_{t-1}\right]. \tag{A 5.9}$$

The Pesaran-Timmermann Test is based on the proportion of times the direction of change is forecasted correctly in the sample. To this end it is necessary to define indicator variables

$$Y_t = \begin{cases} 1 & \text{if } y_t > 0 \\ 0 & \text{else} \end{cases}, \tag{A 5.10}$$

$$X_t = \begin{cases} 1 & \text{if } x_t > 0 \\ 0 & \text{else} \end{cases} \quad \text{and} \tag{A 5.11}$$

$$Z_t = \begin{cases} 1 & \text{if } x_t y_t > 0 \\ 0 & \text{else} \end{cases}. \tag{A 5.12}$$

Defining $P_y = \mathsf{Pr}(y_t > 0)$ and $P_x = \mathsf{Pr}(x_t > 0)$ we can assume $P_y < 1$ and $P_x > 0$ (which is likely to be satisfied when dealing with economic data. Now let $\hat{P} = n^{-1} \sum_{t=1}^{n} Z_t$ and under the assumption that x_t has **no** power in predicting y_t, $n\hat{P}$ is binomially distributed with mean

[14] Pesaran and Timmermann (1992)

$$\begin{aligned} nP_\star &= n\,\mathsf{Pr}(Z_t = 1) \\ &= n\,\mathsf{Pr}(y_t x_t > 0) \\ &= n\left(\mathsf{Pr}(y_t > 0, x_t > 0) + \mathsf{Pr}(y_t < 0, x_t < 0)\right) \\ &= n\left(P_y P_x + (1-P_y)(1-P_x)\right) \end{aligned} \tag{A 5.13}$$

If P_y and P_x are known the test can be built upon the standardised binomial variate

$$S_n = \left\{\frac{P_\star(1-P_\star)}{n}\right\}^{-\frac{1}{2}} (\hat{P} - P_\star) \overset{a}{\sim} N(0,1) \tag{A 5.14}$$

Usually, the probabilities P_y and P_x are not known in advance and have to be estimated. An efficient estimator of these two variables is the mean over the observed cases, i.e.

$$\hat{P}_y = \frac{1}{n}\sum_{t=1}^{n} Y_t \tag{A 5.15}$$

$$\hat{P}_x = \frac{1}{n}\sum_{t=1}^{n} X_t \tag{A 5.16}$$

and therefore

$$\hat{P}_\star = \hat{P}_y\hat{P}_x + (1-\hat{P}_y)(1-\hat{P}_x) \tag{A 5.17}$$

Consequently, S_n is no longer binomially distributed and the standardised statistic for the nonparametric test of predictive performance is given by

$$S_n = \frac{\hat{P} - \hat{P}_\star}{\sqrt{\hat{\mathsf{VAR}}[\hat{P}] - \hat{\mathsf{VAR}}[\hat{P}_\star]}} \tag{A 5.18}$$

where

$$\hat{\mathsf{VAR}}[\hat{P}] = \frac{1}{n}\hat{P}_\star(1-\hat{P}_\star) \tag{A 5.19}$$

$$\begin{aligned} \hat{\mathsf{VAR}}[\hat{P}_\star] &= \frac{1}{n}(2\hat{P}_y - 1)^2\hat{P}_x(1-\hat{P}_x) + \frac{1}{n}(2\hat{P}_x - 1)^2\hat{P}_y(1-\hat{P}_y) \\ &\quad + \frac{4}{n^2}\hat{P}_y\hat{P}_x(1-\hat{P}_y)(1-\hat{P}_x). \end{aligned} \tag{A 5.20}$$

The general standardised Pesaran-Timmermann test statistic is asymptotically distributed $N(0,1)$ under the null hypothesis that the signs of the forecasts and the signs of the actual time series values are independent. For a detailed derivation of the statistic please see Pesaran and Timmermann (1992).

A 5.3 Jarque-Bera Test

The Jarque-Bera Test is a frequently used method to check for normal distribution of (stationary) time series. The test is constructed on the basis of skewness and kurtosis coefficients. Usually, the test is used to check for the normal distribution property of the residuals in a linear regression problem. The test statistic can be calculated as

$$JB = (T - n)\left(\frac{\hat{s}^2}{6} + \frac{(\hat{k} - 3)^2}{24}\right) \overset{a}{\sim} \chi^2(2). \tag{A 5.21}$$

In this context T is the number of observations which is reduced by the number of parameters n (constant included) of a regression model. If only an empirical distribution is tested (and **not** the residuals of a linear regression) n can be set to zero. The variable $\hat{k}$ is the empirical kurtosis with

$$\hat{k} = \frac{1}{T}\frac{\sum_{i=1}^{T}(X_i - \hat{\mu})^4}{\hat{\sigma}^4} \overset{a}{\sim} N\left(3, \frac{24}{T}\right) \tag{A 5.22}$$

and $\hat{s}$ is the skewness of the distribution that can be written as

$$\hat{s} = \frac{1}{T}\frac{\sum_{i=1}^{T}(X_i - \hat{\mu})^3}{\hat{\sigma}^3} \overset{a}{\sim} N\left(0, \frac{6}{T}\right). \tag{A 5.23}$$

The Jarque-Bera Test statistic is asymptotically χ^2 distributed with two degrees of freedom and the null hypothesis presumes $H_0 : \hat{s} = 0$ and $\hat{k} = 3$ (normal distribution) and is rejected if $JB > \chi^2_\alpha(2)$.

Chapter 6

Forecasting through Local Modelling by using SOM and Linear Regression

6.1 Introduction

So far, we have only focused on the power of SOM for the identification of certain patterns in time series. Our first approach was to build trading rules on the basis of the discovered chart patterns in order to evaluate the profitability of the method. The further deployment was to form two data sets from the raw price time series and extract empirical probabilities which were used for forecasting the immediate next out-of-sample value of prices. The drawbacks of this method were discussed in section 5.6. In the following chapter we go one step further and try to straighten out the problems of the previous chapter. To this end, we combine the model from chapter 5 with regression procedures and hope to improve forecasting power.

The subject-matter of this chapter is the effort to gain forecasting improvements from building local linear models. Economic and/or econometric theory usually attempts to find an explanatory model that captures an entire time series, but the performance – in many cases – is poor. There is evidence that economists are able to forecast market volatility in financial markets fairly well, but prices are extremely hard to predict with the standard econometric toolbox.[1] It seems to be a critical assumption that there is only one process or parameter constellation, respectively, that is able to explain the overall features of a data generating process. Nonlinearities in the time series, which cannot be modelled by a single linear functional form, are most likely to be the reason for this sort of problems. Therefore, a logical extension of the current econometric standard toolbox is the adoption of a technique that splits the data driving process into many different cases. This method assumes the time series to be a product of a sequence of distinct local models. In this respect, the idea is closely related to the family of Markov Switching Models, but is rather a vector quantising method than a probabilistic approach. The latter depends on certain probabilities which drive the system's changing between different regimes and therefore the model tries to explain the development of a sequence of data by

[1] See for example Cochrane (2006) or Bacchetta et al. (2006)

switching between these scenarios. The SOM method assumes a limited number of geometrical patterns which recur in time and makes use of these observation without depending on regime switching probabilities. The striking assumption underlying this design is the limited number of different models. The models are suspected to recur over time. That is, it might be possible to break down the time series to categories where the constellation of market forces always lead to the same pattern!

The aim of the model presented in this chapter is the determination of recurrent patterns and the estimation of representative parameter sets. Local models (or: *sub-models*) are identified by the application of Self-Organizing Maps to the data set. As already shown in chapter 4 we will stick to the embedding procedure of data following the suggestions of Takens[2], i.e. the local models are constructed as an autoregressive processes of order q.

The remainder of the chapter is organised as follows. After a short description of the data set, a description of the data embedding procedure and the subsequent employment of the SOM is described. Local models are then estimated and prices are forecasted. At the end a short summary and a conclusion is given.

6.2 Preprocessing and Training

If financial time series are sequences of a limited number of recurring consecutive geometrical structures it should be possible to identify these representative structures and make use of them for forecasting purposes. One of the main tasks to be solved in the framework of such a model would then be the identification of adequate model vectors and the subsequent ensuing assignment of patterns extracted from a time series which is to be forecasted. This problem can be solved by employing a SOM. Before we turn to the architecture of the net used here we first have to describe the preprocessing of the raw time series.

We split the data into two sets.[3] The first part of the time series is used to train the net. The second part is the testing set, which is the workhorse to check the power of the proposed method out-of-sample. The time series has to be reshaped into a predefined embedding dimension before we can apply the SOM algorithm and train the net. The first data matrix adopts the dimension $[(T - q + 1) \times q]$ which is achieved by using the sliding window device again. Here, q is the backshift operator and T is the total number of observations included in the time series. The first part of the sample data is grouped by its geometrical characteristics, i.e. the fragments of the time series are clustered by assigning each element to its closest neuron in terms of the Euclidean distance.

[2] Takens (1981)

[3] See section 6.3.1 of this chapter for a further description of the data we used in this chapter.

The embedding of data was first applied in chapter 4 and was then described in detail in section 5.2. The model presented in this part of the thesis will make use of the same technique. We stick to the notation introduced in chapter 5 and denote the matrix obtained by the first embedding $\mathbf{y}$ and the matrix of the second step $\mathbf{y}'$.[4] For the purposes considered here we focus on $\mathbf{y}'$ which can be splitted into submatrix Γ and subvector Υ. If we treat Υ as the vector of dependent variables and Γ as the matrix of independent variables we can use this data for the estimation of local model parameters. This implicitly assumes that there exists a strong autocorrelation in the observations, which can usually be postulated when prices are the research objective.

In our analysis we predetermine the size of the SOM net as a two-dimensional 6×6 layer and the neurons are interconnected on a hexagonial lattice. The clustering of the matrix $\mathbf{y}$ results in a categorisation of 36 local models represented through the weight vectors formed by the SOM net. Once the first set of data is grouped, the patterns from the matrix $\mathbf{y}'$ that correspond to the respective neurons in the codebook of the trained SOM are determined. As mentioned above there are $(T-q)$ cases from which we know the exact $(q+1)$-th outcome of the patterns used for the clustering in the first step. Each of these patterns is now assigned to one of the 36 models developed before. This – again – is achieved by means of the Best-Matching Unit Criterion (see equation (3.2) and also chapter 5). To this end, only the first q columns of matrix $\mathbf{y}'$ are taken for assigning the data pieces to a best matching neuron on the net. The result of this criterion is an index vector of length $(T-q)$ that assigns each row/pattern of matrix $\mathbf{y}$ to its winning neuron. It can also be used to assign the first $(T-q)$ rows of matrix $\mathbf{y}'$ to its closest representative codebook vectors. In order to assure a matching dimension of the two vectors we only take the first q entries of each row in consideration.[5] This procedure gives 36 new data sets which will be analysed separately. The size of the data sets depends on the number of patterns a node on the net represents. We do not lose any information contained in the data by this procedure because we are interested only in the location of the vector in question on the SOM net. The $(q+1)$-th observation of each row will be used as the dependent variable in the later estimation.

Applying the BMU criterion to the data set $\mathbf{y}$ and combining matrix $\mathbf{y}'$ by the help of the indices c results in N matrices (where N is the number of neurons on the SOM net) with a varying number of rows and q columns of independent variables and one column of dependent variables. The number of rows usually differs from each other due to unequal hit-distribution[6] over the net. Not every node attracts the same

[4] The parameter for autoregressive lags was denoted by q and the length of the raw price series is T.

[5] Note, that this is necessary in order to keep up the dimensionality. Otherwise, it would not be possible to calculate the vectorial distance between the input data and the model vectors.

[6] The *hit-distribution* provides information on the number of patterns that is assigned to each of the neurons on the SOM net. Hence, it can be interpreted as the relative importance of a node and,

number of patterns. In the extreme case of a very rare pattern the representative codebook vector might even unify only a very few data vectors. There is a trade-off between reducing the net size and the explicit caption of rare events. Additionally, for each of the N matrices the corresponding vector of dependent variables is given by the $(q+1)$-th column. The size of the vector will, of course, also depend on the hit-distribution of data on the SOM net. According to the notation in chapter 5 the dependent vector is called Υ_i with $i = 1, ..., N$ and the data matrices are called Γ_i in the remainder of this chapter.

The matrices Γ_i are now used as the data basis for estimating model parameters locally. Different estimation techniques can be applied to such a problem. We will now take a closer look at the use of least squares estimation in this context.

6.3 Least Square Estimation of Local Parameters

Once the data sets are determined through the SOM (in the case at hand we have to deal with 36 different data sets) it is possible to obtain the least square estimates by computing the following equation:

$$\hat{\beta}_c = (\Gamma'_c \Gamma_c)^{-1} \Gamma'_c \Upsilon_c, \tag{6.1}$$

where Γ_c is the matrix of independent variables that belong to the respective winning model c and Υ_c is a vector of dependent variables. The result $\hat{\beta}_c$ is a parameter vector of length q that gives a representative estimator constellation in the submodel (local model) with index c. For all submodels the estimated parameter vector is denoted by $\{\hat{\beta}\}_{q=1}^{Q}$. The underlying assumption of the estimation is that the dynamics of a (highly) nonlinear time series are locally smooth, i.e. it should be possible to approximate a financial time series by splitting it into a number of different "smaller" models. The entire process is then given by:

$$f(x) = \bigcup_{i=1,...,N} R_i(\mathbf{y'}_{c^i}), \tag{6.2}$$

where x is the raw time series, N is the total number of neurons on the SOM net, R_i is the model belonging to node i and $\mathbf{y'}_{c^i}$ is the data set that is assigned to the respective winning node c.

Under general conditions ordinary least squares is a consistent estimation procedure for the parameter in question here. The representative estimated parameter vector for the submodel i should be able to cover the local characteristics of the

therefore, also as the relative importance of the group of patterns that is represented by a certain codebook vector.

data structure significantly better than a global estimator if there are distinct local patterns in a series. By using this technique the traditional approach of a global modeling strategy is disavowed and a local modelling procedure is adopted. Such an analysis can only be accomplished under the assumption that the local dynamics of the underlying data generating process are smooth.[7]

6.3.1 Data

We apply this model to the NYSE daily closing prices for the period of 02/01/1980 until 31/12/1999. This set contains 5055 observations. The trajectory of the NYSE index is shown in figure 6.1. Of course, the price index time series is a non-stationary process. For our purposes this feature does not play an important role since we will have to normalise the data anyway. The normalisation is necessary because of the computational reasons explained in chapter 2.4.

We choose the normalisation method as shown in equation (2.37) and apply it to the matrices $\mathbf{y}$ and $\mathbf{y}'$. Each row of these matrices represents a pattern. For each of them the observations are normalised according to the respective minimum and maximum of each row. An adjustment over the entire time series would only have the effect of a shift in the scaling, but the operation would not lead to the elimination of the original problem of a biased-trained SOM net.[8]

First-differencing of the price time series leads to a stationary process $\{r_t\}_{t=1}^{T}$ which exhibits the well-known statistical properties of return series (stylised facts).[9] The returns are non-normally distributed, having narrow shoulders but more probability mass in the tails (*fat tails*). Furthermore, the phenomenon of *volatility clustering* is observable from the time series. Figure 6.1 (b) shows these characteristics graphically.

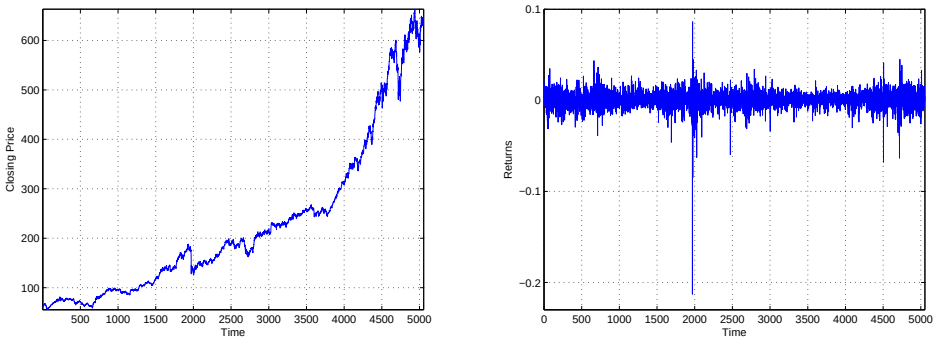
The returns will be of interest later on when we turn to the analysis of the forecasting power of this model. In the first step we have to deal with the estimation of prices. Due to the construction of the model at hand we are only interested in the estimated parameters in a certain situation (local parameters). The idea of breaking down global complex processes into several local submodels stems from the notion that most financial markets have the *long memory property*, i.e. if an event occurs today it is likely that it affects the future for all future points in time.[10] This property is usually defined as the hyperbolic decay of the autocorrelation function. However, our memory-argument is different from that stated before. Under the

[7] Principe et al. (1998)

[8] A shift in scaling would not eliminate the problem of the SOM method to attach a higher weight to patterns with higher values. We are rather interested in the geometrical nature of the patterns than in their absolute numerical values. In order to avoid a bias during the training process it is reasonable to restrict every pattern to fall into the interval $[0, 1]$.

[9] See 4.2 for a short summary.

[10] Peters (1994)



(a) NYSE closing prices (b) Corresponding NYSE return series

Figure 6.1: NYSE data from 02/01/1980 to 31/12/1999

assumption that the market processes information in a certain way it is reasonable to think of the market as an institution that remembers situations from the past. A market consists of a large number of interacting social beings. There is no logical reason why these agents should forget their experiences or why these experiences should cancel each other out.[11] This stands in contrast to the assumptions often imposed on models for the analysis of financial markets.[12] Therefore, the market can be assumed to react on the arrival of new information or the upcoming of certain situations as it has already done in the past. Hence, given a certain trajectory of e.g. the price, the market will adjust its reaction according to its experiences from the past. This is exactly what the proposed SOM model tries to mimic and it can be seen as the theoretical justification of the application of feature-domain models in finance.

It is doubtful whether statistical models that incorporate short term variables only have the ability to adequately explain the long memory component of the market. The estimation by means of the transformed time series into the interval $[0; 1]$ is feasible and directly translates into the non-normalised time series (e.g. for the later use of forecasting).

[11] If the later point held true, this would intrinsically mean that there is no valid learning from experiences. All agents on the market would draw opposite conclusions from the same situation. This does not seem to be a very realistic scenario.

[12] For instance, de Long et al. (1990) use a simple overlapping generation model for the analysis of the risk generated through the existence of noise traders on financial markets. In this framework agents live only for two periods and act according to rational expectations, i.e. the costs of acquiring information are virtually zero and every agent has the same access to information. This is a simplifying, but rather unrealistic assumption which limits the model to a myopic viewpoint and prevents it to account for agents' experiences.

6.3.2 Parameterisation

For models using SOM as an ingredient the parameterisation of the training phase is a crucial factor. It was already pointed out earlier that the grid size, the training phase length, and the initial learning rate (among others) have an impact on the accuracy of the adjustment of the SOM net to the real distribution of the data. Table 6.1 summarises the parameter set-up of the model.

The number of neurons on the map has an important influence on the outcome of the training. The larger the number of neurons the more precise will be the topological reconstruction of the inputs on the map, but the less informative is each of the model vectors. In the extreme case of constructing a net with exactly as much neurons as input vectors we would achieve a perfect fit of the SOM to the data but there would not be any reduction in complexity or filtering of information. By construction of the SOM algorithm this has to be the case because every neuron would adapt exactly to that data vector that is its nearest, no more and no less. The less neurons a researcher allows on the network the more patterns or data vectors, respectively, are concentrated on the respective neuron. This makes the topology of the system a bit more imprecise and the distances between the nodes become larger. If one of the data vectors would be located on the border between two neurons and only marginally closer to one of the two, it is probably not very well represented by the neuron it is assigned to.

From table 6.1 we can see the further parameter set-up of the model. The splitting of the training process into a *rough training* and a subsequent *fine tuning* phase is generally accepted in the literature for reasons explained in chapter 3. The parameter values for the initial radii and the learning rates are set arbitrarily, but are geared to the parameterisation in Chen and Tsao (2003).

6.4 Training Process

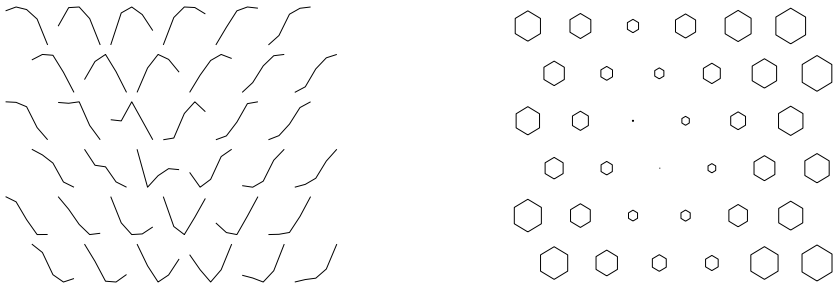
We split our data set into two unequally sized parts. The first part will be used as a training set for the Self-Organizing Map and the second part is taken as the testing data in the subsequent analysis. From the NYSE data we take the first 4000 observations and construct a matrix of size $[3996 \times 5]$, which corresponds to an embedding dimension of five lagged values for each entry ($q = 5$). This data set is sufficiently large for a consistent training of the SOM and we find 36 representative patterns extracted from all 3996 input vectors. These patterns are shown in figure 6.2 (a). Since every vector that contributes to the training of the net is normalised between 0 and 1 it is possible to abandon the axis of the respective graphs. The main focus is on the geometric and symbolic characteristics of the pattern families. Each of the patterns shown here complies with a certain class of events that produce

Parameter	Value
Number of Iterations Rough Training	1000
Number of Iterations Fine Tuning	1000
Number of neurons	36
Backshift Parameter (q)	5
Initial Radius Rough Training	6
Final Radius Rough Training	1
Initial Radius Fine Tuning	1
Final Radius Fine Tuning	0.1
Initial Learning Rate Rough Training	1
Initial Learning Rate Fine Tuning	0.1
Total NOBs	4000
Number of Time Series Patterns	3996

Table 6.1: Parameter set-up

recurrent patterns all over the entire set of training data. That is what we called *long memory* in the previous section. We can identify several well-known patterns from financial markets such as uptrends or downtrends and even hump- or u-shaped gradients are observable.

A further finding is that direct neighbours on the map turn out to be most similar. By construction of the SOM this has to be the case because direct neighbours are linked and the topological structure (and its smoothness) strongly depends on the relative equality of the symbols in neighboured neurons. The question arising next is about the distribution of data over the map, i.e. how many input vectors are clustered at one neuron and what are the distances between the neurons on the map. These information can be obtained from figure 6.2 (b) where the relative distances of the nodes are presented. Each hexagon refers to one node and the sizes of the nodes indicate the distances from one representative pattern to its neighbours. This is an important information since it provides an insight into the equality of the underlying model vectors. The closer the neighbours are located to each other, the more equal are the models. If we were confronted with a map that shows a very narrow image it would be worth to reduce the number of neurons in order to concentrate more information on less vectors. The same applies the other way around. If the distances are too large over the entire map it is very unlikely that the model vectors give a good representation of the overall data structure. In such a situation an increase of neurons might be helpful. By visual inspection we can see that in the case at hand the distance map confirms the choice of the number of nodes. The distances on the edges of the map are relatively small, therefore the degree of equality of the patterns discovered by the SOM is very high. In the centre of the map a different picture is drawn. The neurons are separated by a large Euclidean distance, i.e. there is only



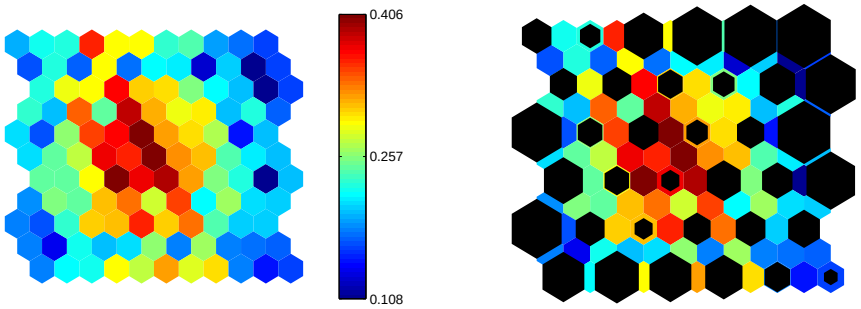
(a) Patterns (b) Distances

Figure 6.2: Patterns and distances discovered by SOM for the NYSE

a little evidence for similarity between the nodes.

Some further interesting information can be obtained by inspecting figure 6.3. In subplot (a) the U-Mat of the SOM is shown (see section 3.3.2 for a detailed description of U-Mats). It is simply an alternative illustration of figure 6.2 (b) and shows us the distances between the nodes in terms of the *connections* rather than by using the nodes. This explains the increased number of depicted hexagons. In subplot (b) of the same figure SOM hit markers are added. The black dots correspond exactly to the magnitude of patterns that is grouped on a certain neuron (relative to the others). The graphic can be interpreted as a distribution plot of the input vectors over the representative (local) models.

If we compare the hit-histogram with the distance plot in figure 6.2 (b) it is easy to see that the smallest distances correspond to the highest response of neurons on the map. This is not necessarily always the case but in the analysis at hand it gives some evidence for a high degree of similarity between the patterns identified on the edges of the map. To some extend this can be traced back to the fact that our backshift parameter, i.e. the embedding dimension, is set to five. The generated input vectors are simply not able to exhibit a highly complex structure because the length of the data piece is too short. In a later analysis we will have a closer look at the performance of the model when applied to higher-order backshifts.



(a) U-Mat (b) Hit-Histogram

Figure 6.3: U-mat and hit-histogram for NYSE

6.5 Forecasting

6.5.1 General Findings

After training the SOM and estimating the parameters as described above by dint of the calibration from table 6.1, by construction of the model we obtain exactly 36 local models. The parameters of these models lay a regression line through the different data clouds generated by the implementation of the BMU criterion. The resulting parameters are representative ones and stand for a whole class of models. We will use these parameters for the forecasting of the respective next future value of a time series (one-step ahead forecasting).

The procedure works in the following way. After embedding of the normalised out-of-sample data taken from the NYSE (a total of 1000 observations) we obtain a $[(T-q+1)\times q]$ matrix. By means of the BMU criterion each of the $(T-q)$ vectors is assigned to one of the model vectors generated by the SOM (the *winning neuron* m_c). We take the pertinent parameter set of the local model m_c and calculate the next value by

$$\hat{y}_{t+1} = \hat{\beta}_c x_t, \tag{6.3}$$

where x_t is the vector containing $(y_{t-4}, y_{t-3}, ..., y_t)$, i.e. the q previous non-normalised observations of the dependent variable y_{t+1}. The parameters used for estimating the models are given in table 6.2. The first line of each model's entry presents the parameters for the estimation in focus here.

In the last column of table 6.2 the hit-distribution or the neuron response, respectively, is given. In figure 6.4 (a) these numbers are presented in terms of a bar diagram. The responses of the neurons range from nine to sixty (with a total number of 996 patterns in the analysis) and at a first glance they are neither exhibiting

any nodes that are unifying too many patterns nor neurons that attract too few patterns. In subfigure 6.4 (b) the three neurons that unify the largest number of patterns (models 31, 33 and 25 - thick solid lines) and the three most infrequent codebook vectors (models 36, 16 and 17 - dashed lines) are indicated. What one can learn from this graph is the dominance of uptrend patterns over the analysis period. This is not a surprising result because under consideration of figure 6.1 (a) there is a definite long-term upward trend observable. After categorisation of the price series into a limited number of subsets it has to be the case that the model vectors which represent the general trend unify the most patterns. The more interesting question is the *shape* and the *persistence* with respect to the lag length of the "extreme" patterns.

Among the 36 discovered representative patterns we find many well known trajectories from literature on technical analysis, e.g. *hump-shaped*, *u-shaped*, *measured move up*, *double top*, or *broadening bottom* trends. For model 1 in subplot 6.4 (b) the *bump-and-run reversal tops* apply.[13] From Bulkowski's viewpoint the *bump-and-run reversal tops* are characterised by a sequence of rising prices along a trendline, then round over and decline afterwards through a trendline. Bulkowski considers relatively long time intervals for his analysis and argues that these sort of chart patterns is mainly observable in the "short-term" period (Bulkowski considers a short-term period to be of length up to three months). Here, a significant shorter period is considered ($q = 5$ corresponds to five trading days in this framework) and the question arising from this analysis is if results in terms of the shape of pattern change if a longer lag-length is used to train and estimate the SOM.

To this end, the results corresponding with lag-length $q = 30$ are shown in figure 6.4 (c) and (d). If the four subplots are compared visually several interesting findings become clear:

- due to the higher number of data points in the case $q = 30$ the curves appear smoother,
- most of the patterns identified with the first SOM ($q = 5$) are recovered by the second SOM (longer time horizons) as well,
- the hit distribution of the second SOM ranges on the interval $[7; 85]$ which indicates an adequate size of the SOM net because the extremes are not overloaded with patterns.

The second item is a very important one. It is easy to see from graphic 6.4 (b) and (d) that the alignment of the submodels on the SOM nets are distinct but still many of the chart patterns generated by the SOM with the shorter backshift recur

[13] See Bulkowski (2000) for a detailed interpretation of typical chart patterns and their use in technical analysis.

with almost the same trajectories in the $q = 30$ case. That brings up the question whether the patterns we observe are lag-invariant.

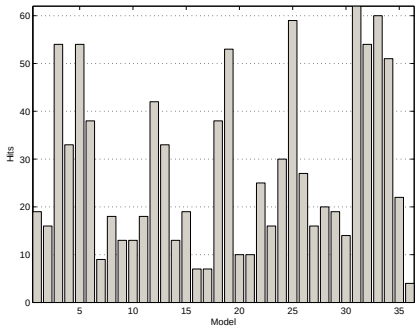
One of the hyperparameters that play an important role in the framework of the SOM method is the number of neurons. The user of the method is confronted with a permanent trade-off between the accuracy of the trained net with respect to the topology of the input data and the degree of information reduction, which is the ultimate goal of the SOM algorithm. As it was pointed out in section 3.2.6, it is extremely difficult to prove convergence to an ordered state and researchers have only succeeded to do so in the relatively simple case of a one-dimensional SOM (see section 3.2.6). The determination of the optimal number of neurons is closely connected with the convergence issue because convergence speed and the resulting shape of the trained net have to be taken into account. So far, there is no rule to set the optimal number of neurons. The user of the method has to predetermine this parameter under consideration of the respective problem. In case of pattern recognition in financial time series it is reasonable to restrict the number of representative patterns to an interpretable maximum (e.g. allowing for 500 different model vectors would not compress the information contained in the time series and the resulting map would gain additional insights).

One possibility to recheck the net size is the training of the map with a higher number of neurons. In case the input vectors are not more dispersed than in the case of the original design, this may be assessed as an indication for an adequate grid size. Of course, this is not an exact method, but it allows for reasonable approximations. Generally, the map size should be as small as possible under consideration of the problem at hand.

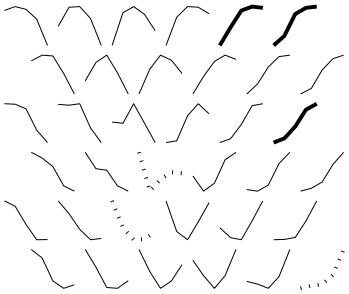
6.5.2 Prediction

All the features described in the previous subsection are by-products that can provide very useful information on the underlying structure of the data but do not lead to the ultimate aim: prediction. So far, the test data patterns were assigned to the best-matching codebook vectors of the trained SOM. From these assignments it is possible to infer the model parameters that are used for forecasting (see table 6.2). Treating the first q elements of one test data pattern as the vectors of independent variable forecasting the $q + 1$ value is accomplished by a simple weighted summing as given in equation (6.3).

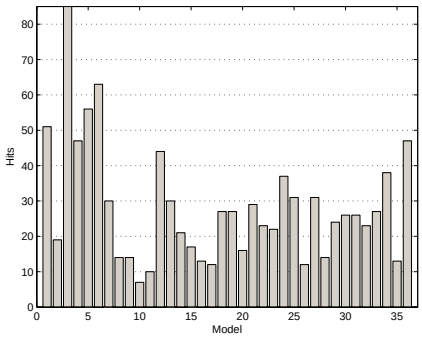
The result of this forecasting technique is plotted in figure 6.5. The dashed blue line indicates the true devolution of the time series and the red one shows the course of the one-step ahead prediction. Because the two lines are so close together the second graph in the same figure plots the residuals of the two time series. At a first glance the forecasting performance seems to be very good. The trajectory of the time series is emulated in a relative realistic way. In subplot 6.5 (a) a snapshot of 50



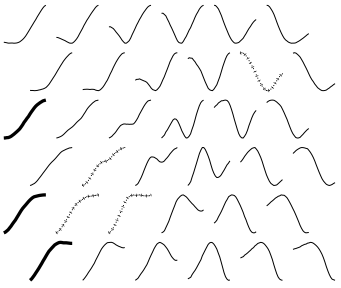
(a) Bar diagram ($q = 5$)



(b) Patterns and "extreme" patterns ($q = 5$)



(c) Bar diagram ($q = 30$)



(d) Patterns and "extreme" patterns ($q = 30$)

Figure 6.4: Hit-distribution/neuron response of the model for the test data set with $q = 5$ and $q = 30$

observations is cut-off and zoomed in. Even in this magnification the deviations of the forecasted time series from the real observations appear to be extremely small. It is worth noting that for this snapshot the period with the largest prediction errors was chosen.

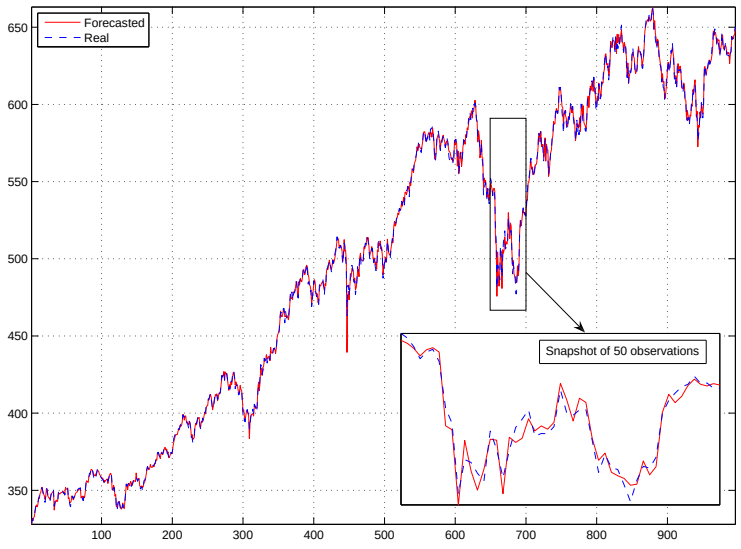
The snapshot makes clear that also volatile periods are covered nicely by the model. Nevertheless, it is also obvious that the local model approach tends to overshoot the true evolution of the price series. This is due to the fact that in periods of large swings the proposed algorithm chooses a parameter set that follows the initial (sometimes extreme) up- or downward impulse. In the subsequent period the time series suddenly jumps up and the algorithm has to adjust the parameter set by searching a new submodel that explains the current situation better. This "braking" effect (the immediate switching between different submodels) leads to the staccato image of the forecasting time series and explains to some extent the overshooting of the smoother true observations.

Employing the CDC statistic (see 5.4.3) in case of the NYSE data we are using here 81.53% of the forecasted data have the correct directional change. Taking the simple random walk as a benchmark a rate of $\approx 50\%$ of correct prediction is expected. In comparison with this naïve model the SOM technique presented in this section performs relatively good. But this is a very simple test that is not suitable for hypothesis testing. Later on, other – more sophisticated – testing procedures are introduced in order to check for the significance of the results.

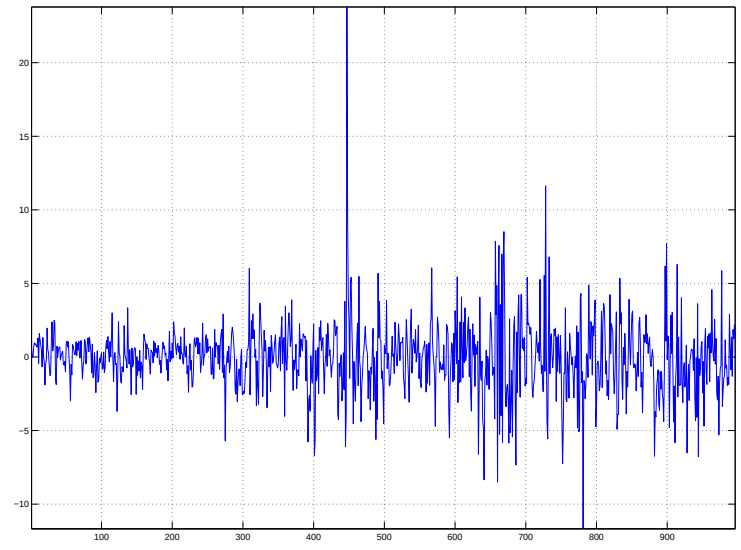
Using the RMSE as a criterion of the evaluation of prediction power for the case at hand the SOM performs better than the random walk model ($RMSE_{SOM} = 2.2624$ and $RMSE_{RW} = 5.1725$). This result is confirmed by the MAPE statistic ($MAPE_{SOM} = 0.2948$ and $MAPE_{RW} \approx 0.74$), i.e. the percentage of absolute errors made by the random walk is higher.

Table 6.3 summarises the tests applied to the model. Several tests well-known from neural network evaluation have been applied to the forecasts generated by the SOM model. Most of the test statistics exhibit promising results in favour of the new method.

The Diebold-Mariano test statistic is clearly below the critical value of -1.69 and therefore we can conclude that the forecasting errors of the SOM model and those of the benchmark model are not equally distributed (5% significance level). In the appendix A 5.1 it is mentioned that the Diebold-Mariano Test in its original form tends to produce too large test statistics for small samples. An adequate modification can be found in the appendix to chapter 5, but it is not used here. The sample size is large enough to result in a correct and unbiased statistic. The same applies for the Pesaran-Timmermann Test, where the null can be rejected at a 5% significance level, i.e. the distributions of the signs of the actual realisations and the signs of the forecasts are **not** distributed independently.



(a) Comparison forecasted series and real values



(b) One-step forecasting errors

Figure 6.5: Real vs. forecasted values

Moreover, the analysis of the structure of forecasting errors is important to learn more about the accuracy of the model in question. In figure 6.5 (b) the deviations of the forecasted values and the real time series is shown graphically. The errors are calculated simply by taking

$$e_t = y_t - \hat{y}_t. \tag{6.4}$$

The forecasting is optimal if the errors are unbiased, i.e. it is necessary to test for zero mean in the forecast errors. Usually, this can be done by a simple t-test. The errors are regressed on a constant and the resulting t-statistic can be used to assess the null hypothesis of zero mean in the series. Applying this to the forecasting errors of the model here, the null cannot be rejected (p-value ≈ 0.009, $\mu \approx -0.183$). Since the errors are computed as shown in equation (6.4) the result of the test is a sign of a systematic overestimation of the proposed forecasting method. In other words, the SOM method tends to be more optimistic in the prediction of prices. The bias of the distribution of forecast errors can also be seen from graphic 6.6 (a). This histogram[14] supports the test results from above and plots the distribution of the sample with the highest mass around the mean 0.3 and has more narrow shoulders and more observations in the tails than the Gaussian normal distribution (for convenience, the Gaussian normal distribution is plotted, too). This is caused by the high fluctuations in the last third of the forecasting horizon, specifically the large swings between period 650 and 800.

Most of the tests considered to check the optimality of forecasting errors are built upon the **unforecastability principle**.[15] This key concept states that if we deal with optimal forecast errors, they should be unpredictable on the basis of the information set available at the point in time when the forecast is made. If there is a way to forecast these errors that means not all the information contained in the data has been used efficiently to predict the time series in question. A frequently applied approach to this problem is the *Mincer-Zarnowitz Regression*. The regression has the form

$$\mathbf{y} = \beta_0 + \beta_1 \hat{\mathbf{y}} + \epsilon \tag{6.5}$$

where $\mathbf{y}$ is the vector of true values, $\hat{\mathbf{y}}$ is the vector of the one-step ahead forecasts and ϵ is the disturbance term. In case of optimal forecast with respect to the information used to construct the prediction, the expected parameters are $\beta_0 = 0$ and $\beta_1 = 1$, i.e. subtracting $\hat{\mathbf{y}}$ from both sides of equation (6.5) leads to

$$\mathbf{e} = \alpha_0 + \alpha_1 \hat{\mathbf{y}} + u_t \tag{6.6}$$

[14] The data points are categorised into 80 bins.
[15] Diebold (2001) pp. 289

with expected parameter values $(\alpha_0, \alpha_1) = (0, 0)$ when $(\beta_0, \beta_1) = (0, 1)$. For this test the null hypothesis that the constant is zero and the slope coefficient β_1 is one is tested. Applying the Mincer-Zarnowitz Regression in form of equation (6.5) to the data set at hand leads to the parameter set $(\beta_0, \beta_1) = (1.1837, 0.99726)$ and with $p \approx 0$ for both parameters (highly significant), i.e. the null can clearly be rejected at a 5% level. The forecast of the model with respect to the information is not optimal. One explanation could be that the categorisation into 36 models might be too rough for the complexity of the time series. The lack of optimality is probably a result of the neglect of combination of model classes identified by the SOM.

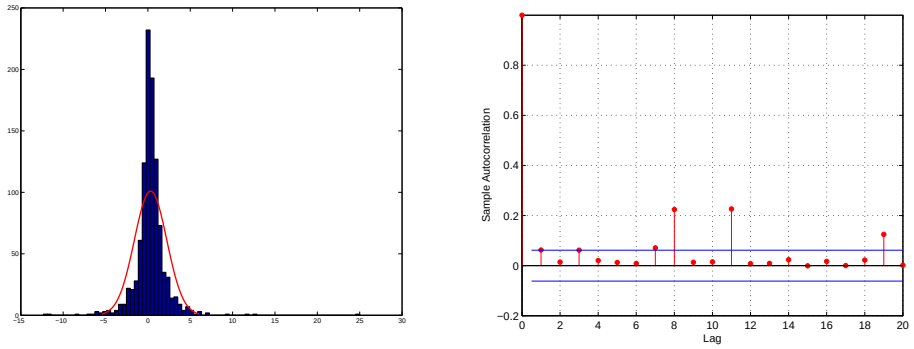
A further way to check for the goodness of forecasting is to analyse the *White Noise Property* of one-step ahead errors. Specifically, this can be tested by means of first-order autocorrelation tests or more general approaches, such as the Box-Pierce or the Ljung-Box statistics. The Durbin-Watson test rejects the null hypothesis of $\rho = 0$ because $d_0 < d_l$, where d_0 is the corresponding value of the test for our case and d_l is the lower bound of the Durbin-Watson test statistic. That means first-order autocorrelation in the forecasting errors cannot be rejected at the 5% significance level. Hence, the Ljung-Box Test for randomness is applied. The Ljung-Box Test is based on the so-called Ljung-Box Q-statistic, which is given by

$$Q = N(N+2) \sum_{k=1}^{L} \frac{r_k^2}{N-k} \tag{6.7}$$

where r_k^2 is the squared sample autocorrelation at lag k, N is the sample size, and L is the number of lags included in the sample autocorrelation function. Under the null hypothesis that there is no serial correlation in the sample (the model fit is adequate) the Q-statistic is χ^2 distributed. Following the results that are given in table 6.3 the null is clearly rejected. There seems to exist some serial correlation in the forecast errors. This result is supported by the autocorrelation plot of the absolute values of forecasting errors shown in figure 6.6 (b). The blue line is the confidence bound which is fixed at 0.061692.

Moreover, the Jarque-Bera Test statistic clearly rejects the null hypothesis of $\hat{s} = 0$ and $\hat{k} = 3$ (skewness and kurtosis coefficient, respectively) in the forecasting errors.[16] Hence, the distribution of the errors deviates from normality. The impact of this conclusion is hard to interpret, but it stands in contrast to the desired white-noise property of forecast errors.

[16] For details on the test, please see appendix A 5.3.



(a) Histogram of forecast errors (red line is Gaussian normal distribution)

(b) Sample autocorrelation of squared forecast errors

Figure 6.6: Forecast error analysis of NYSE data

6.6 Weighted Least Square (WLS)

After training of the neural grid each of the trained nodes represents a whole group of data that stands behind the pattern. The Best Matching Units are found by searching the model vectors that minimise the distances to an input data vector that is presented to the net in the Euclidean space. Thus, a model vector of a trained Self Organizing Map describes only the center of a cluster of patterns that unifies similar geometric characteristics. Some of the grouped patterns within one class are represented with a high degree of accuracy, others lie further away from the center. In the extreme case an input data vector is located exactly on the border between two nodes. This would be the case when the distance to the input is exactly the same for two codebook vectors.

It is now assumed that the estimation will be more accurate if those input patterns that are further away from the center of a cluster have attached a smaller weight when computing the representative parameters. This hypothesis will be checked in the following. From an econometrical point of view a natural approach to this problem is the definition of a weighting matrix $\mathbf{W}$. The weights in this context are usually defined as a diagonal matrix which is incorporated into the regression

$$\hat{\beta}_c = (\Gamma_c' \mathbf{W} \Gamma_c)^{-1} \Gamma_c' \mathbf{W} \Upsilon_c, \tag{6.8}$$

where

$$\mathbf{W} = \begin{pmatrix} \omega_{1,1} & 0 & \dots & 0 \\ 0 & \omega_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \omega_{N,N} \end{pmatrix}. \tag{6.9}$$

Generally, the respective weights are calculated by means of a weight function $\mathbf{W}(\delta)$ that fulfils the following properties:

1. $\mathbf{W}(\delta) > 0$,
2. $\mathbf{W}(-\delta) = \mathbf{W}(\delta)$ and
3. $\mathbf{W}' \leq 0$ for $\delta \geq 0$.

In our application the variable δ denotes the Euclidean distance between the centre of a node and the input data pattern at hand. The first condition ensures that we exclusively have to deal with positively defined weights. The second item makes sure that the weight function does not make any differences between the direction of measurement, i.e. the function should disregard whether the input data pattern e.g. lies to the west or to the east of the codebook vector in the two-dimensional Euclidean space. Only the distance itself is of interest for the correct loading of a pattern. Since distances cannot become negative the second item is redundant and only mentioned for completeness. The third property demands the weighting function to be nonincreasing for $\delta \geq 0$. This elementary property ensures the weights to decrease as the distance to the centre increases.

The *bicube* and *tricube* functions satisfy these properties. Formally, they can be written down as

$$\mathbf{W}(\delta) = \left(1 - \left(\frac{\delta_i}{\max_i \delta}\right)^2\right)^2 \tag{6.10}$$

or

$$\mathbf{W}(\delta) = \left(1 - \left(\frac{|\delta_i|}{\max_i \delta}\right)^3\right)^3, \tag{6.11}$$

respectively. Of course, the weighting function could acquire an alternative form but the bicube and tricube functions are easy to implement and describe the relation of the input data to the cluster node in a sufficient manner.

Our guess is that one of the main advantages of the weighted least squares method is the ensurance of the continuity between the submodels (or: local models). Unweighted input patterns may lead to a bias in the estimation of the local models and restrain the continuity of the models in the boundary regions. Through the integration of the weight matrix into the estimation scheme the procedure accounts for the relative importance of each of the subgroup members unified in one neuron. But even though we allow for a discretised model selection the transition between the neurons is smoother than in the case without weighting of inputs.

The estimated parameters obtained from the weighted least squares estimation are given in table 6.2. Comparing the WLS estimators with the ones calculated

Model	β_1	β_2	β_3	β_4	β_5	Hits
1	-0.2885	-1.2593	0.1961	1.2829	1.0651	19
	-0.3830	-1.2705	0.1413	1.4004	1.1083	
2	-0.3745	-0.6100	0.0394	1.5016	0.4447	16
	-0.4800	-0.4540	0.0207	1.3506	0.5630	
3	-0.4852	-0.1895	0.5107	0.5419	0.6241	54
	-0.5587	-0.0788	0.5091	0.5024	0.6275	
4	-1.0801	0.0984	1.8870	0.3281	-0.2355	33
	-1.0868	0.1394	1.8785	0.3315	-0.2654	
5	0.0501	-0.2815	0.7215	-0.0417	0.5526	54
	-0.0085	-0.2640	0.8632	-0.2014	0.6118	
6	-0.0179	0.1530	0.0776	0.0549	0.7336	38
	0.0133	0.1373	0.0640	0.0963	0.6901	
7	1.8504	0.1587	-0.3814	-0.3995	-0.2345	9
	1.5493	0.4420	-0.5690	-0.2274	-0.2036	
8	-0.0351	0.1720	-0.4631	0.5059	0.8222	18
	-0.3992	0.1376	-0.4631	0.6805	1.0475	
9	-0.4354	-0.1527	0.5103	0.2646	0.8141	13
	-0.6756	-0.2336	0.4931	0.4918	0.9276	
10	-1.3747	0.4059	2.0346	0.4827	-0.5454	13
	-1.5326	0.5202	2.0769	0.6603	-0.7214	
11	-0.6611	0.3057	1.3876	0.4659	-0.4962	18
	-0.6184	**-0.1707**	1.4570	0.5146	-0.1769	
12	-0.1519	0.0536	0.2078	-0.0659	0.9603	42
	-0.1712	**-0.0455**	0.0569	**0.0175**	1.1475	
13	0.6235	1.2107	0.0928	-0.4811	-0.4497	33
	0.5414	1.1787	0.1428	-0.4290	-0.4393	
14	-1.0421	2.1189	0.2010	-0.2795	-0.0029	13
	-1.1202	2.0335	0.0574	-0.3554	**0.3805**	
15	-1.3220	1.0421	-0.2471	0.9869	0.5438	19
	-1.8328	0.9684	-0.1211	1.2218	0.7670	
16	-0.8634	-0.3981	1.0152	0.4489	0.7968	7
	-0.7556	-0.0182	1.5641	0.1657	0.0440	
17	-0.3250	-0.8184	1.6598	0.1749	0.3129	7
	-0.4415	-0.9921	1.9936	0.2699	0.1757	
18	0.5060	0.4924	0.1594	-0.2598	0.1017	38
	0.7211	0.4901	0.2954	-0.4131	**-0.0942**	

Table 6.2: Parameters for each of the N models identified by the SOM. The first line of each model's entry is the parameter vector for the least squares estimation; the second line shows the parameters when weighted least squares estimation is applied.

Model	β_1	β_2	β_3	β_4	β_5	Hits
19	-0.0936	0.5072	0.0350	0.0392	0.5105	53
	-0.1765	0.5928	0.1432	0.0369	0.4000	
20	-1.2915	1.6424	1.2692	-0.5916	-0.0305	10
	-1.3781	1.7699	1.4231	-0.6815	-0.1363	
21	-1.1293	1.0031	1.4797	-0.6639	0.3105	10
	-0.8803	1.1304	1.3421	-0.7318	0.1408	
22	-0.2185	-0.4145	0.4879	0.6862	0.4584	25
	-0.0791	-0.4649	0.3824	0.6325	0.5267	
23	0.3011	0.2523	-0.0457	-0.5777	1.0712	16
	0.4101	0.3941	-0.1275	-0.7711	1.0942	
24	1.4734	0.5726	-0.4593	-0.6384	0.0522	30
	1.6599	0.6809	-0.5950	-0.7443	**-0.0016**	
25	0.2216	0.1090	-0.8054	0.4957	0.9801	59
	0.2384	0.0782	-0.9689	0.6317	1.0211	
26	-1.2855	1.1181	1.4247	0.1011	-0.3591	27
	-1.4232	1.2396	1.5199	0.0494	-0.3879	
27	-0.9784	0.0504	1.4783	-0.0249	0.4750	16
	-0.8744	0.0905	1.4647	-0.0287	0.3497	
28	-0.7881	-0.3193	0.6516	0.7910	0.6616	20
	-1.0272	-0.2094	0.7249	0.9559	0.5532	
29	0.7518	-0.1956	-0.7137	1.0329	0.1247	19
	0.5844	-0.2085	-0.6362	1.1158	0.1452	
30	1.2265	0.6248	-1.1961	-0.2362	0.5836	14
	1.3590	0.6334	-1.1384	-0.4140	0.5633	
31	-0.0875	0.1726	0.1198	0.1645	0.6295	62
	-0.0519	0.0832	0.1823	0.1190	0.6650	
32	-0.4921	0.1185	0.7343	0.2753	0.3617	54
	-0.4812	0.0606	0.8134	0.3160	0.2884	
33	-0.6844	-0.0955	1.0090	0.5599	0.2090	60
	-0.6773	-0.1234	1.0373	0.6265	0.1352	
34	-0.8594	-0.0260	0.7290	0.8134	0.3429	51
	-0.9471	**0.0226**	0.7690	0.8119	0.3429	
35	-0.8011	-0.3316	0.1381	1.3960	0.5974	22
	-0.7484	-0.4282	0.1909	1.3640	0.6205	
36	-0.1064	-0.1197	-0.1523	0.5624	0.8247	4
	-0.0908	-0.0994	-0.2161	0.6507	0.7666	

– Table continued –

with the simple least squares method we find that the sign of the parameters is almost always coinciding within each submodel. For the rare cases when the sign differs between the alternative models (in table 6.2 indicated by bold numbers) the values are close to zero which explains the switching sign to a certain extent. The only exceptions are the differences between parameters β_2 in model 11 and β_5 in model 14. The estimated parameter values differ significantly from each other when different methods are used.

Table 6.3 summarises the test statistics of this chapter.[17] We compare the two different approaches (least-squares and weighted-least squares estimation) with a simple random walk as a benchmark model. Taking the RMSE as a figure to measure forecast accuracy the SOM-based methods turn out to perform better than the simple random walk. We were able to beat the random walk with both, the normal least-squares estimation and the weighted least-squares technique. The overall picture shows that the less sophisticated normal least squares method outperforms the WLS.[18] Comparing the two approaches with each other the MAPE is lower for the LS case and the CDC is slightly higher. The Diebold-Mariano Test leads to ambiguous results. In the LS case the null of equal forecasting errors of the SOM-based method and the benchmark model (random walk) can clearly be rejected. For the WLS estimation the same test indicates that the random walk and the local modelling technique result in equal forecasting errrors.

The Jarque-Bera Test gives for both methods qualitatively similar results. The Pesaran-Timmermann Test clearly rejects the null of independently distributed signs of the true realisations of the time series and the signs forecasted by the proposed methods. This emphasises the relatively good CDC value. For the encompassing test (see subsection 5.4.3) we find highly significant parameters for the respective SOM model, but insignificant ones in the random walk case. Therefore, neither of the two models outperforms the other one in any case (the null can only be accepted if **both** models exhibit significant **or** insignificant parameters).

6.7 Sensitivity Analysis

One of the dissatisfactory elements of the previous analyses was the ad hoc predetermination of the embedding dimension. We tried to approach this problem with Taken's Embedding Theorem and the False-Nearest-Neighbour method (see sections 4.3 and 4.4) but found that it is not possible to simply apply this procedure to the

[17] Most of these tests were already introduced in chapter 5 (see section 5.4.3).

[18] We also tested the forecasting errors generated by the WLS method for unbiasedness by means of the procedure explained in equation 6.4 and thereafter. The regression resulted in a mean $\mu = -0.042$ and was insignificant ($p \approx 0.576$). From this test it remains unclear whether the forecasting errors are biased because the null hypothesis cannot be rejected.

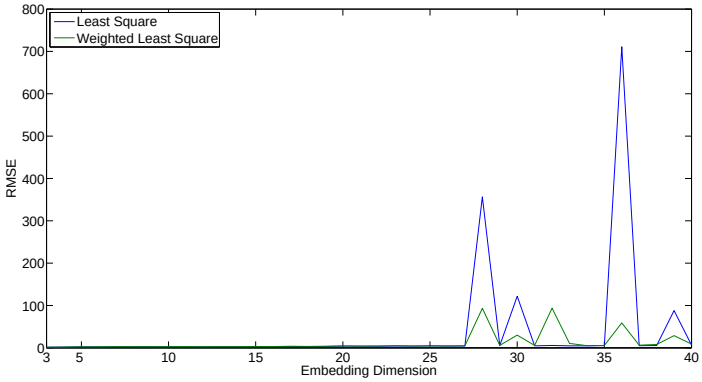
Test	LS	WLS
RMSE	2.2624	2.4763
RMSE Random-Walk	5.1725	
MAPE	0.2948	0.3165
MAPE Random-Walk	0.7388745	
Correct Directional Change (CDC)	0.8153	0.8102
Pesaran-Timmermann Test	19.9111	19.5831
Diebold-Mariano Test	-2.1880	-1.3931
Jarque-Bera Test	82.5624	82.5931
Ljung-Box Test (25 lags)	18771.11	18770.08
(critical value)	(31.41)	
Durbin-Watson (forecast errors)	1.5032	1.4817
Encompassing Test (p-value SOM)	0.03535	0.11859
Encompassing Test (p-value RW)	0.7253	0.6985

Table 6.3: Tests for forecasting capability of the SOM model. Least squares (LS) and weighted least squares (WLS) are compared in the last two columns (values rounded).

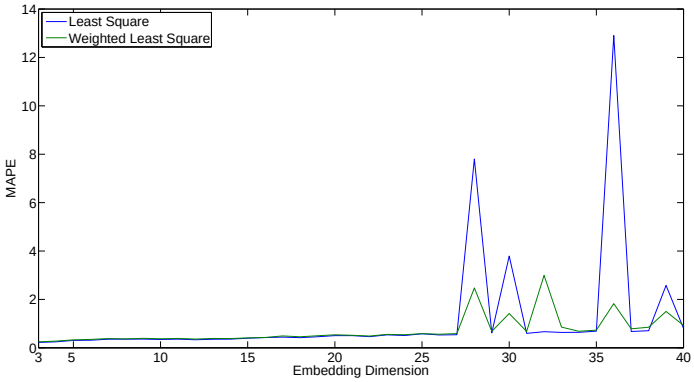
time series we are interested in. So far, we justified the selection of the embedding dimension with economic arguments, such as the length of one trading month. Of course, it might be disputed whether the choice of one month is correct and why prices should in particular be affected by their historical values within this timeframe and not rather a longer or shorter one. As long as a reasonable quantitative measure to determine the optimal embedding dimension is missing economic reasoning appears to be the most reliable procedure. However, in the framework of the two approaches analysed in this chapter it is interesting to take a closer look at the models' performances under the implementation of changing backshift parameters.

The goal is to check for differences in the forecasting performance which are directly connected with the number of backshift parameters. To this end, we take the two approaches presented in this chapter (LS and WLS) and estimate each of them for $q = \{3, 4, \ldots, 40\}$, i.e. 37 different embedding dimensions. We abandoned backshift parameters 1 and 2 because we basically look for recurrent patterns and the SOM needs an appropriate minimum of observations for working properly. The analysis was accomplished under the *ceteris paribus* condition, therefore the parameter set-up as given in table 6.1 is valid for every estimation. The time series used here is again the NYSE index from 02/01/1980 until 31/12/1999 (daily closing prices). For each of the different q's the RMSE, the MAPE and the CDC is calculated. The results are summarised in figure 6.7.

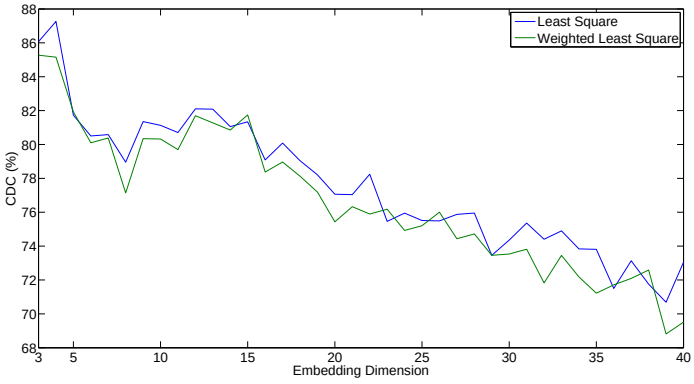
Blue lines correspond to the least squares version and green lines belong to the weighted least squares model. In subfigure 6.7 (a) the RMSE of the two methods is plotted. For $q = \{3, \ldots, 27\}$ the RMSE is low and relatively stable. The values for



(a) RMSE



(b) MAPE



(c) CDC

Figure 6.7: Goodness criteria for varying embedding dimensions

the LS and the WLS are very close together. For the case $q = 28$ there is a sudden jump of the RMSE in both cases, but it is more pronounced for the LS. In the further developing of the RMSE both techniques exhibit qualitatively almost always identical trajectories. The only exception is $q = 32$, where the WLS forecasting goodness decreases dramatically and the LS does not show any significant reaction. The same behaviour can be seen in the second figure 6.7 (b). The spikes generated by the LS approach are more extreme than those given by the WLS. The overall picture obtained from plot (a) is confirmed. Up to embedding dimension $q = 27$ the MAPE is quite low and stable. Thereafter it becomes difficult to choose a "good" backshift parameter. The last criterion we consult for this analysis is the CDC statistic. The picture leads to a somewhat different result than subplots 6.7 (a) and (b). The embedding dimensions from $q = 5$ to $q = 12$ show a temporary decrease in the CDC values. The series is not monotonically decreasing as we would have expected under consideration of the results obtained from figures (a) and (b). However, the general message of the plot is the same: the higher the number of backshift parameters the worse becomes the forecasting capability.

A further striking feature of the third subplot is the tendency of the LS method to systematically outperform the WLS application. In the other two graphs the contrary was indicated. It seems that the weighting of the parameters leads to a smoothing in forecasting. This is likely to be the reason for the lower MAPE and RMSE of the WLS method. On the other hand, this smoothing impeded the system to reproduce bigger jumps in the time series which, eventually, resulted in a worse overall performance than in the LS case.[19]

We also experimented with embedding dimensions $40 < q \leq 100$, but the SOM models started to become uncontrollable. In the framework of this enhanced experiment the RMSE and the MAPE did never permanently return to the low values we obtained for $q = \{3, \ldots, 27\}$. Moreover, in some cases we experienced problems to estimate a reasonable parameter vector (some of them were very close to zero). Therefore, it suffices to present the analysis up to $q = 40$ backshifts here since no improvement in the criteria was found for larger embedding dimensions.

6.8 Frequencies and Cycles

6.8.1 Analysis of Regularities

A central question to be answered in this part of the chapter is whether the frequency of the appearance of patterns contains any useful information and if the patterns appear in cycles. When the patterns occur in a certain organised arrangement it

[19] Because of scaling problems it is not visible in figures 6.7 (a) and (b) that for the RMSE and the MAPE the LS method leads to superior results for embedding dimensions $q = \{3, \ldots, 27\}$.

will be of further interest whether there is anything to learn from the periods of appearance and the length of the breaks patterns take before recurring.

To approach these considerations we resort to a graphical representation. This is very convenient since it helps to organise a possible structure in the data and makes it recognisable by visual inspection. In order to give a better understanding of the graphical analysis used here, we refer to figure A 6.1 (see appendix to this chapter), where the 36 subplots together show the last 1000 observations of the New York Stock Exchange price series. This corresponds exactly to the predetermined out-of-sample testing data set. The (red) bars right above the x-axes indicate the location of model i ($i = 1, ..., 36$) when assigned to an observed pattern, i.e. the vertical lines (*frequency bars*) visualise the beginning of the employment of a certain codebook model for forecasting the one-step ahead future price. This display format was chosen for the identification of potential structures in the representative models' occurences. First of all, the graphic confirms the distribution that was shown in figure 6.4. Since every plot corresponds to one model and not every codebook vector represents the same number of patterns it is clear that the frequency bars generate a different image for each of the 36 subplots. Again, from figure A 6.1 the relative importance of each model vector identified becomes obvious.

At a first glance, there is no identifiable regularity in the respective subplots. The employment of representative codebook models seems to be completely random. Some patterns exhibit strong influence over outstanding periods of the time series, such as the "crash" in between $t = 550$ and $t = 650$ in the second half of the series. Models 32 and 36 are obviously the models that are used most frequently, but models 3, 4, and 6 are the codebook vectors that are used most often to explain the large downswing in the time series. Note that figure A 6.1 departs from the order as given in subplot 6.2 (a). The order of the models in the graph at hand are in transposed order compared with the trained map, i.e. the first horizontal row corresponds to the first column of the net drawn in figure 6.2 (a). Model 1 with 60 hits and model 36 with 52 hits are the two patterns which explain most of the forecasting. This is an interesting finding because precisely these two models are the most opposed ones as well. Model 1 is geometrically characterised by an initial upswing and a subsequent stagnation, whereas model 36 has a slight downward movement that is followed by a sharp fall and an adjacent upward trend. These two different types are almost evenly distributed over the forecasting horizon but from a close visual inspection of this graph no definitive conclusion about the existence or non-existence of any kind of cycle is possible.

In order to facilitate the graphical analysis the time differences (Δt) of the appearance of codebook models is plotted in figure A 6.2. On the x-axes the n-th appearance of model i is plotted; the y-axes show the time difference Δt between the starting points of employment of a certain model. The plots can be interpreted

as follows. Suppose a model is used to forecast the value of a time series in the next point of time. The method used here then stops the time until the previously used model is employed again. The difference between these two employment times is plotted of every model in figure A 6.2. If we consider the case of a perfect synchronous recurrence of patterns the plots were simply a horizontal straight line. A completely different image is generated by the graphical method. Following the plots in figure A 6.2 any evidence for the hypothesis of evenly distributed patterns can be rejected. Unfortunately, the graphic does not provide any evidence for the existence of observable regularities over time. Furthermore, it seems to be impossible to give any expedient statement on the correlation between the frequency of appearance and Δt.

6.8.2 Parameter Variation

How much does this result depend on the number of neurons on the net? The answer on this question is hard to determine analytically. In fact, to the best of the author's knowledge there is no quantitative testing procedure applicable to this problem. Therefore, the graphical method is consulted again in order to approach this question. Figure 6.8 illustrates the time difference Δt of the recurrence of a certain model i. As in the previous analysis of figure A 6.2 it is obvious that there are no systematic cycles in the appearance of any of the models. The selectivity between the four models becomes more significant, i.e. models 1 and 4 unify the overwhelming majority of data vectors, whereas the codebook patterns 2 and 3 are only of marginal relevance. This confirms the assumption that the employment of a model does not follow a stable frequency or anything near that.

A further extremely influential parameter is the lag-length q of the AR(q)-process the model assumes. So far, the lag (or: backshift parameter) was set to 5. This value was chosen arbitrarily and could adopt any other integer that makes sense for the objective of the research. Indeed, in section 4.4 it was tried to identify the optimal lag length by means of the False-Nearest-Neighbour method, but it is arguable whether this technique is applicable to the problem at hand. The class of models proposed here strongly depends on the symbolic characteristic of patterns. The presumption is that information about the data generating process is hidden in the geometrical trajectories of the data set. It remains an open question what data frequency fits best with which number of backshift parameter. This is mainly subject to trial and error and the selection of the optimal lags is more an art than a science.

Nevertheless, in the following a lag length of $q = 22$ backshifts is chosen. This value is also chosen arbitrarily but can be justified economically. The data set used here consists of daily data. The average month has roughly 30 days and approximatily eight non-trading days. Therefore, we define the 22 day horizon as the average

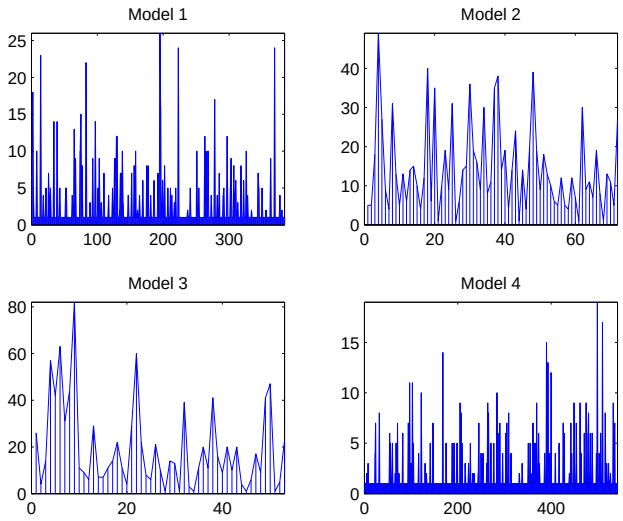


Figure 6.8: Δt of frequencies of the appearance of four codebook models

trading month of financial markets. The figures A 6.3 and A 6.4 in the appendix to this chapter provide the same analysis that was already accomplished in the previous part of this section. Obviously, the distribution of the use of the respective codebook models is even more ambiguous than in the case of $q = 5$. From the second graph where Δt is plotted there is still no identifiable systematic recurrence.

6.9 Are Certain Patterns Interconnected with Volatility Clusters?

We will now turn to the question whether certain models identified by the SOM are interconnected with periods of very high or very low volatility. The importance of this issue is obvious: if there are really evidences for positive (or even negative) correlation between the application of a certain representative model and phases of outstanding volatility, this fact could be exploited and used for prediction. Furthermore, a regular appearance of volatility clusters within one class of models would lead to an insight of the market forces and crucial patterns would exhibit the geometrical characteristics of such a situation.

To approach this problem we propose a method which is developed in the style of the graphical representation of high and low points of stocks within a trading day. In order to summarise the major facts of the stock's intraday trajectory a *candlestick graph* is used. Candlestick graphs are a comprehensive aggregation of the opening price, the closing price, the highest and the lowest value of a stock within a trading day. Due to the efficient display of this method it is convenient for our purposes to

lend some of its main features and apply them to our problem.

Before creating a candlestick graph it is necessary to preprocess the data in question. Since our aim is not the depiction of stock prices (but the volatility within the different model classes) we first have to extract data on the volatility from the model. To this end, returns are – as usual – calculated from the raw time series as

$$r_t = \ln(P_t) - \ln(P_{t-1}), \tag{6.12}$$

where P is the price in the respective point in time and r is the corresponding return. In the case at hand, volatility is defined as absolute returns, i.e. $\sigma_t = \|r_t\|$.[20] We are interested in the intra-class volatilities of the different submodels. Therefore, it is necessary to further differentiate the data. First, the absolute returns for the time windows which were used for the training of the SOM are calculated. Each of these sequences is assigned to the corresponding model vector by employing the best matching unit criterion again.[21] The patterns of absolute returns are then separated and ordered following the information obtained from the preceding procedure. This approach leads to a matrix of intra-window volatilities for the different families of patterns. Taking the mean of all the cases belonging to a subgroup i leads to an average volatility sequence $\bar{\sigma}_i$:

$$\bar{\sigma}_i = \frac{1}{M} \sum_{k=1}^{M} \|r_k\|, \tag{6.13}$$

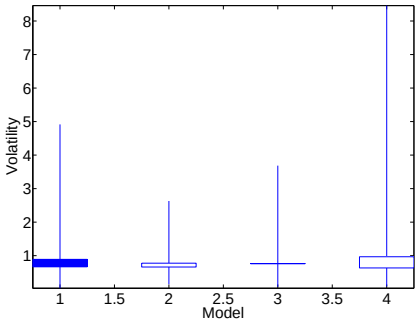
where M is the total number of volatility sequences assigned to model class i. The resulting vector consists of q elements from which the highest and the lowest values are determined and additionally the first (entrance) and the last (exit) absolute returns are extracted. These are the data necessary of the candlestick diagram.

In figure 6.9 the candlestick diagrams for three different map sizes are shown. The number of models is chosen arbitrarily and give an overview of the change of volatility coverage for a varying number of available model types. The bars represent the volatility range within a class of models and the boxes indicate the entrance and the exit average absolute returns. A blank box describes an entrance volatility that is lower than the exit value and the coloured boxes represent the contrary. At a first glance, each of the variants shown here exhibit a concentration of high volatility ranges around the first third of all possible models. If we compare the distribution of the high absolute returns with model vectors in figure 6.4 (b), we can observe that these patterns are associated with downward sloping trajectories.[22] From a technical

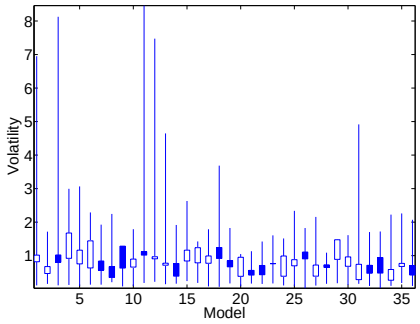
[20] Note, that this is one of diverse possibilities to describe volatility; another usual form is the squared returns.

[21] Of course, this task has to be accomplished in two steps. First, we search the BMUS for the normalised price patterns and in the second step the BMUS are translated into the volatility data.

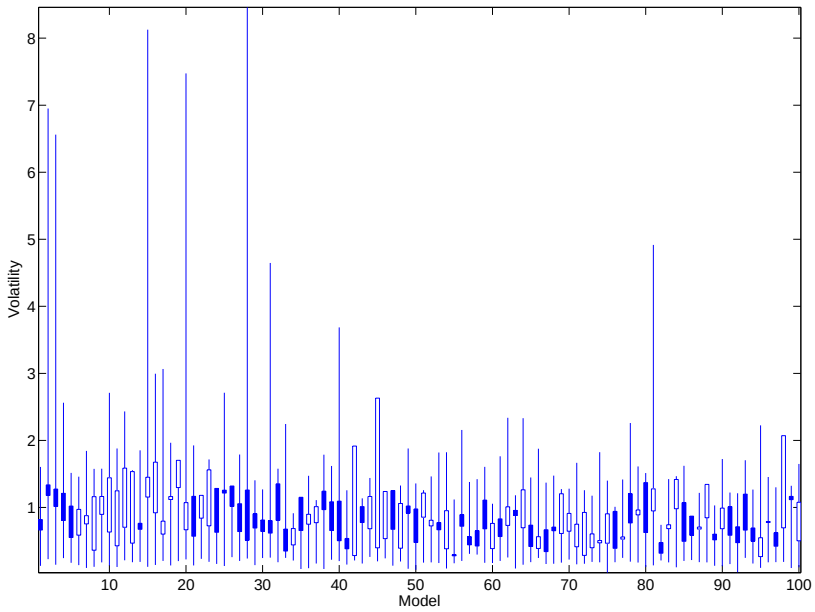
[22] Note, that in e.g. figure 6.4 (b) the models are ordered column-wise, i.e. model 1 is the the first in



(a) 2×2 SOM Map



(b) 6×6 SOM Map



(c) 10×10 SOM Map

Figure 6.9: Candle stick diagrams for different grid sizes ($q = 5$)

point of view this is an interesting feature. The proposed method seems to work better in the case of upward sloping trends. The reason for this is the appearance of crashes. Obviously, the SOM method on one hand attempts to follow the time series' trajectory, but in the same moment it is reliant on its previous training. The algorithm forces the model to choose the model vector that minimises the Euclidean distance to the input vector. Hence, even if the algorithm takes a decision for a certain model the Euclidean may still be large in relative terms. The best matching unit is not automatically a perfect mapping for an extreme market situation.

One reason for the model's difficulties to capture strong downward swings is the general positive trend in the time series; the model does simply not expect a sudden decrease in future prices as much as it anticipates the inherent upward trend. Hence, most of the fluctuation in these situations are not covered very well by means of the proposed method. An important weakness of the model becomes clear: extreme swings in the data are not covered well by the SOM method and therefore, it is not very suitable as a crash predictor.

However, this conclusion has to be treated carefully. It is essential to interpret the numbers under the consideration of the distribution of input data. Taking the hit distribution into account, we see that 757 input vectors are attached to the outstanding model vectors 1, 3, 11 and 12 from figure 6.9 (b) (the models with the largest volatility range in the case of a 6×6 SOM map). This quantitity corresponds only to approximately 16% of all the data vectors that were presented to the map for training. Hence, even though some of the model vectors carry a heavy load of volatility range, the majority of the codebook vectors is able to capture the fluctuations of the volatility.

The picture changes a bit when the lag-length is increased to e.g. $q = 30$. The absolute average volatility decreases significantly from above eight to below three, but it is consistent with the finding in the previous case with less backshifts. The reason for the lower average absolute returns is the averaging over all return inpute vectors within one model class. For the model parameterised with window width $q = 30$ the extreme volatility sequences are more equally distributed over all 36 models. Thus, the average volatility of these classes are smaller than the ones in the previous case. However, the $q = 5$ variant assigns many of the extreme absolute return patterns to two or three codebook vectors. This leads to significantly higher σ's in real terms. By comparing these two cases the increased lag length seems to function as a smoothing element (please see figures 6.9 (b) and 6.10).

From the drawing of the entrance and exit volatility values it is not possible to deduct any regularity. Indeed, on the average it seems to be the case that the bigger part of models has lower absolute returns in the beginning than in the end, but the examination of patterns underlying the coloured boxes does not lead to any

the upper-left corner of the map, model 2 is the second pattern in the first column and so on.

disclosures of the process' nature.

6.10 Chapter Summary

In this chapter the SOM method was used to model financial markets with the help of local models. The idea was to split a complex global sequence of data into smaller patterns and then find a number of small submodels which represent the time series in a piece-wise manner. The submodels were given by the model vectors of a Self-Organizing Map after training. Each of the patterns was assigned to a certain submodel which, in turn, was assumed to be an adequate proxy for this pattern. Once the submodels were identified we used the attached patterns to estimate representative parameters and took them for one-step-ahead forecasting. This technique was called *local modelling*. The approach was based on the assumption that the time series is "locally smooth", i.e. the employment of tailor-made situational models should lead to a good approximation of the overall process. The task was to discriminate between the different situations and to estimate the respective parameter vector. Furthermore, after estimation an algorithm was needed that categorised the data into their model families in order to assign the right parameter set to forecast out-of-sample. Both exercises were accomplished by a Self-Organizing Map and the Best-Matching-Unit criterion. In order to estimate the parameters two different approaches were used: the least squares estimation and its weighted variant. It was found that the weighted least square method was not able to outperform the – simpler – least squares estimation. We expected to obtain improved results from the WLS compared with the simple LS method. This expectation was based on the assumption of a boundary smoothing effect. Obviously, the smoothing effect outweighed the geometrical features of the model vectors which led to a deterioration of performance of the overall method.

The second part of this chapter's analysis aimed at a different question. We suggested to check for a connection between the observation of certain patterns (or: geometrical symbols) and their corresponding volatility. To this end, we looked at the intra-class volatility levels of the respective patterns and tried to find some regularity by comparing the distribution of volatility over the patterns. It was not possible to find reliable evidence for a stable relationship between the level of volatility and the underlying geometrical symbol.

A last exploration was designed to check for cycles in the data, i.e. we tried to find an indication for the assumption that patterns recur in equally spaced time intervals. Obviously, if there were such a regularity one could use this information for a variety of different applications. We chose a simple graphical approach to answer this question and found that there is no obvious argument in favour of our

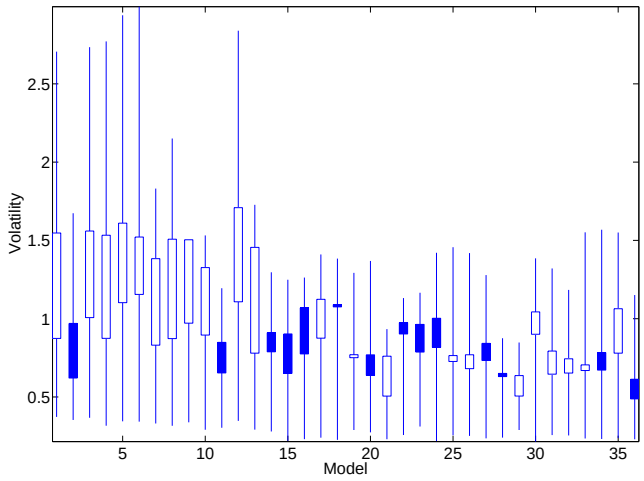


Figure 6.10: Candlestick graph for $M = 36$ and $q = 30$

assumption. The chapter closed with an analysis of the models identified by the SOM and their respective volatility patterns. We found that the largest volatility ranges are covered by downtrend models which was attributed to the general positive trend in the non-stationary price index we used for the analysis.

The SOM based local model estimation method presented in this section is a technique that can be extended in several ways. An interesting extension is the connection of SOM and sophisticated CI techniques in order to improve the accuracy of the method. This was done several times in literature but works only for certain data sets.[23]

With this analysis we end our examination of SOM-based models for forecasting purposes. In the next chapter a SOM is used to analyse macroeconomic data and to offer a new perspective for the interpretation of certain macroeconomic coherencies.

[23] For instance, see Fontenla-Romero et al. (2002).

Appendix to Chapter 6 - Figures and Tables

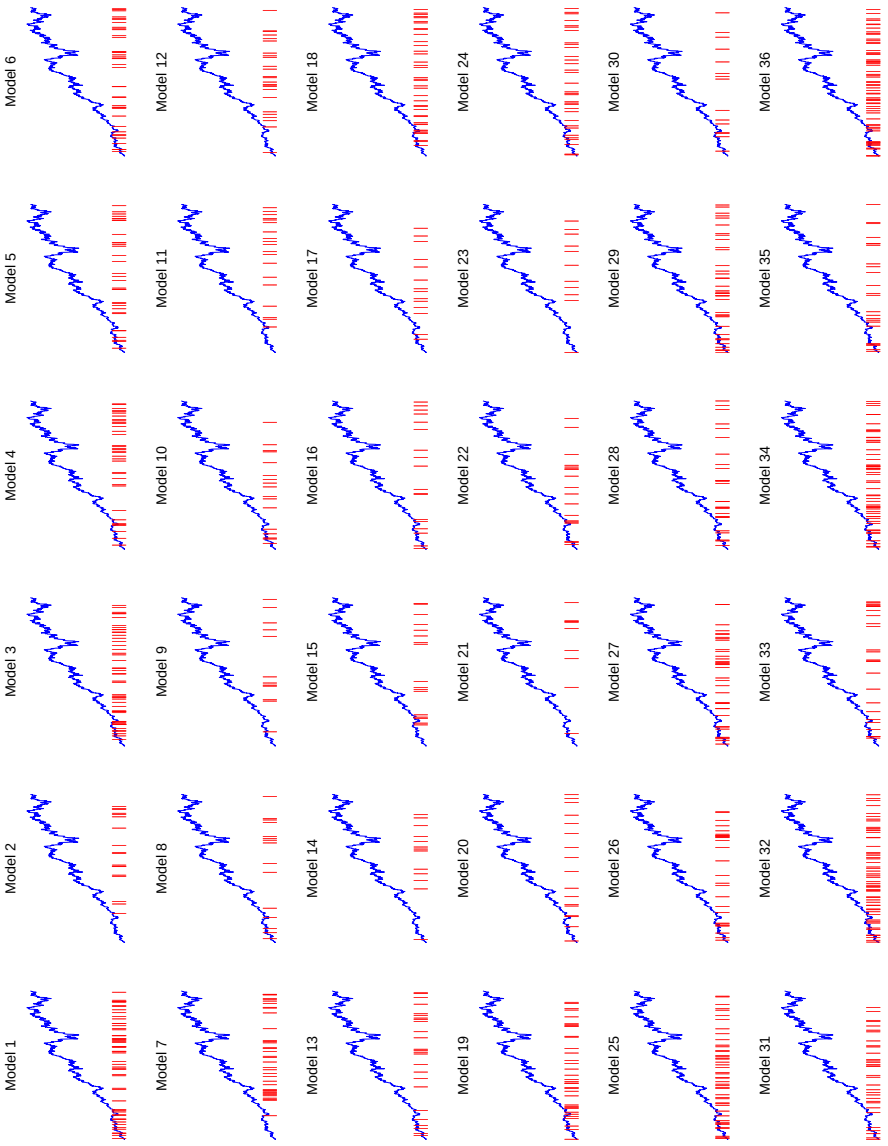


Figure A 6.1: Frequencies of the appearance of codebook models ($q = 5$)

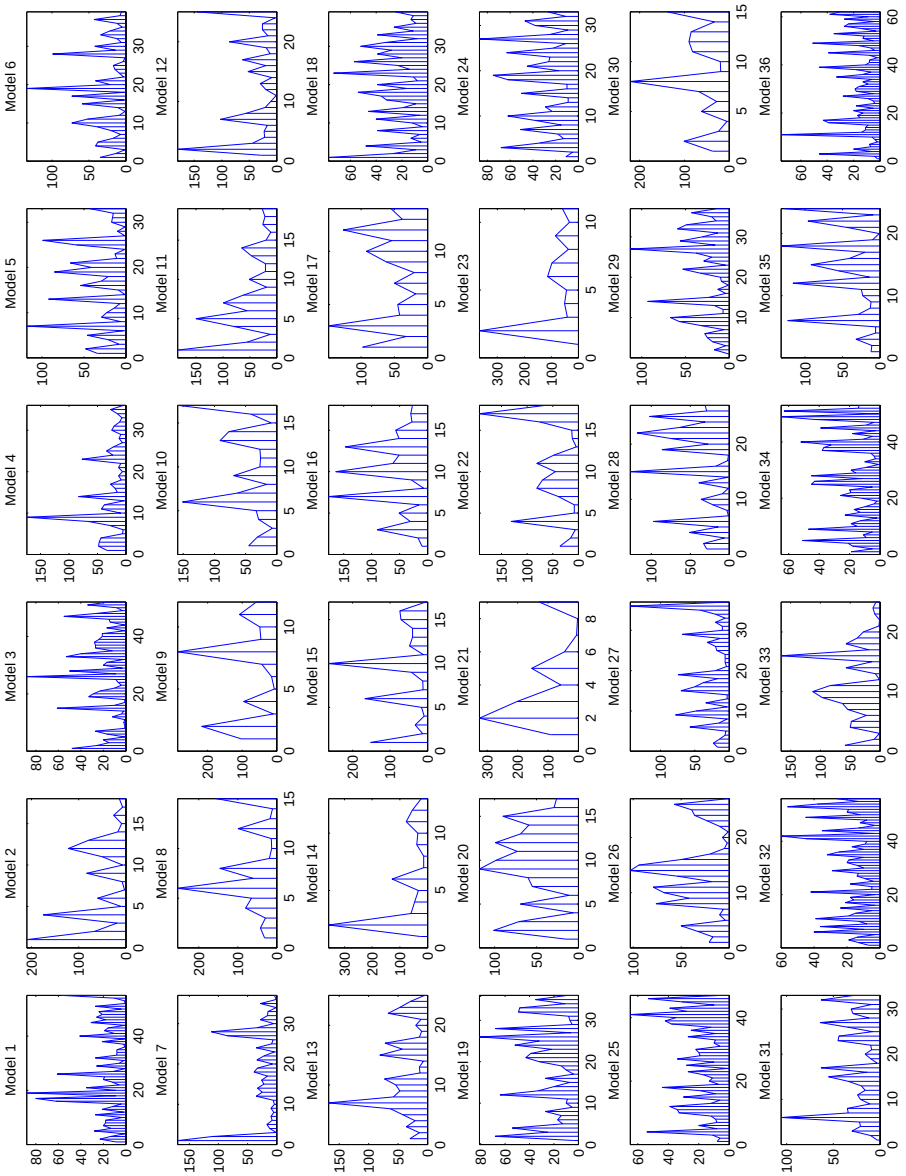


Figure A 6.2: Δt of frequencies of the appearance of 36 codebook models ($q = 5$)

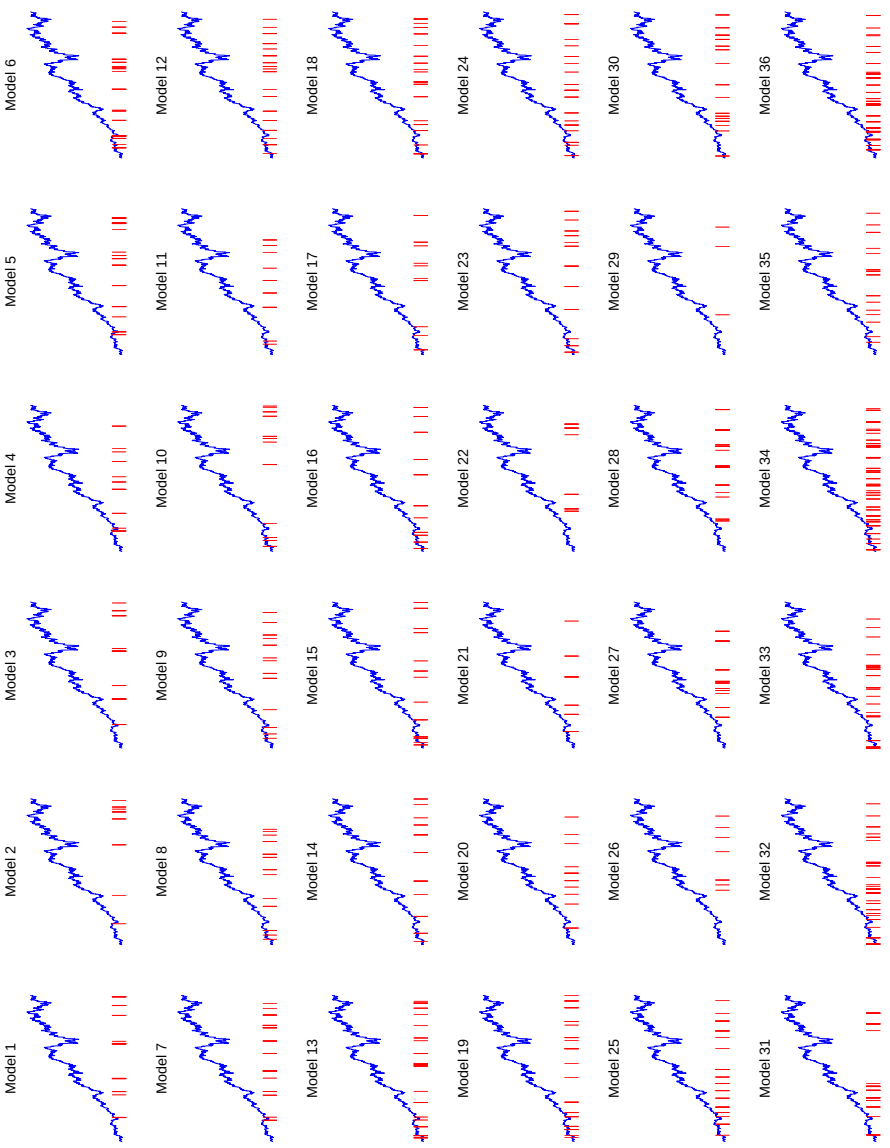


Figure A 6.3: Frequencies of the appearance of codebook models ($q = 22$)

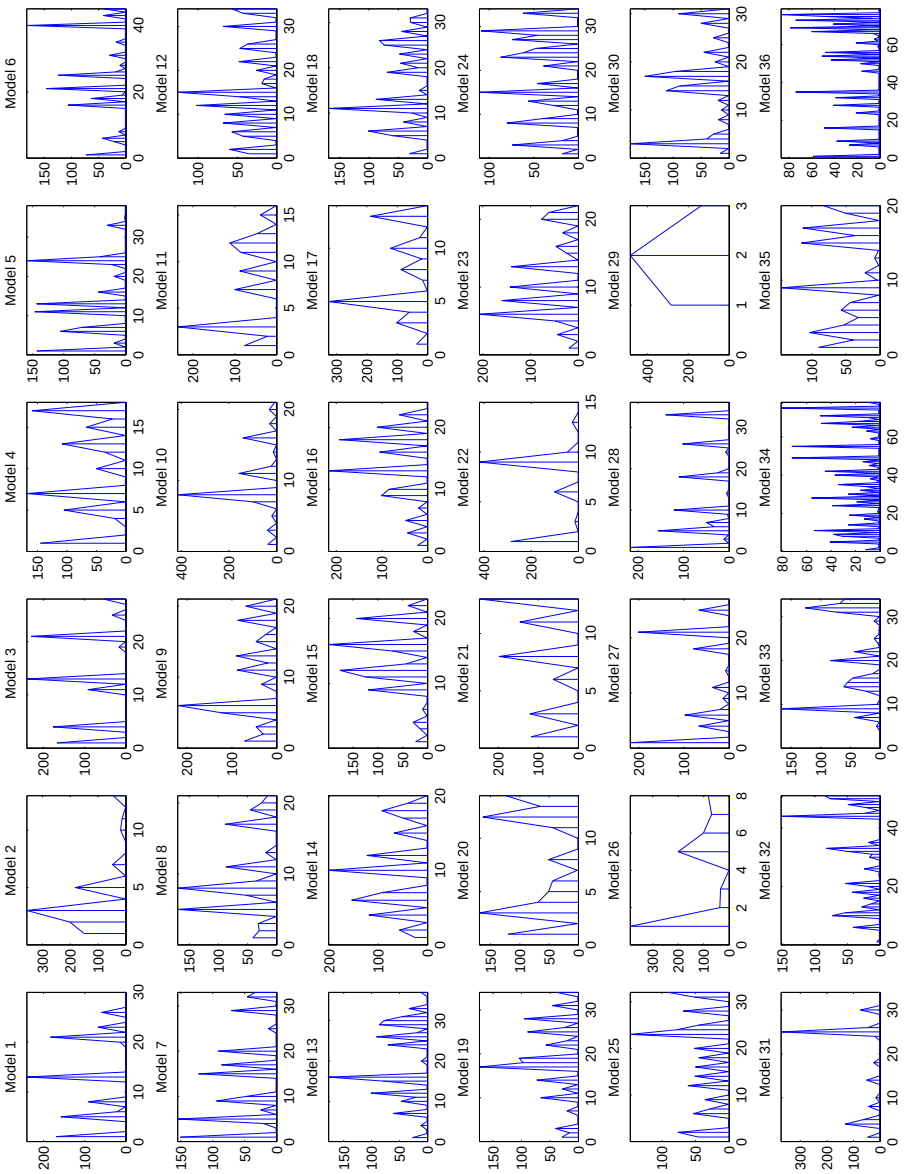


Figure A 6.4: Δt of frequencies of the appearance of 36 codebook models ($q = 22$)

Chapter 7

Exploratory Data Analysis with SOM

7.1 Introduction

In the previous chapters we have shown the application of the Self-Organizing Map algorithm mainly to prediction problems. All of these applications have in common the preprocessing of data which was accomplished by means of the sliding window device. So far, we have formed patterns from a time series that could then be used to feed the algorithm with the necessary data. However, in practice this approach seems to be insufficient because in real life one is often confronted with multi-dimensional data which may all play an important role in a decision making process. For instance, to take an investment decision whether one should buy a certain stock depends on a number of hard and soft factors such as average returns for different maturities or the experience (measured as tenure) of the management etc.. It is often a very sophisticated task to process all the data coinstantaneously and to come up with a correct decision on the basis of this information set.

SOMs provide a convenient and powerful way to translate high-dimensional data sets into relatively simple one-, two- or three-dimensional maps. This makes it easier for the user of the method to quickly cope with complex market variables and to take a decision on the basis of visual identification of similarities or particularities in the data structure. The method provides a different viewpoint and captures features in the data that are hardly identified by other techniques. It will be shown in this chapter that SOM cluster homogenous groups of data surprisingly well without any prior knowledge of their actual affiliation. This is in line with the works by Serrano-Cinca[1] and Deboeck[2].

In the following a Kohonen Map is trained with macroeconomic data from 55 different countries. The resulting net is analysed extensively by various methods and the results are interpreted. In the subsequent step the time path of a certain country is investigated, i.e. we look at the movement of a country on the map and how its position on the net changes over time. This is an interesting question because we

[1] Serrano-Cinca (1998) p. 3
[2] Deboeck and Kohonen (1998) p. 39

will try to infer from that movement whether the country is situated in dangerous zones ("crises zones").

7.2 Financial Crises

One of the aims of this chapter is to extract "crises zones" from the macroeconomic data. This knowledge should be useful in order to evaluate a country's state in terms of economic vulnerability. Once the map is trained it will be possible to identify certain areas, each with a different degree of economic stability. Assigning a country to the trained map will help to infer of the candidate's situation by means of its neighbourhood. In order to better interpret the later analysis we give a short summary on the most influential financial crises models at this point of the thesis.

In the theoretical literature there are usually six different types of currency crises models described. A general precondition of most of these models is the presence of a fixed exchange rate system (*Peg*). The classes of approaches to explain currency crises can be subdivided into first and second generation models, banking-crisis models, twin-crisis models, herding behaviour models and models based on the moral hazard assumption. This section is devoted to give a short and intuitive overview of the current status of the literature in this area of research.

7.2.1 First Generation Models

First generation models try to reproduce the situation of some Latin-American countries in the 1970s and 1980s. These models go back to the work of Krugman (1979) and Flood and Garber (1984). In these approaches a small economy suffers from a trade deficit which – in a fixed exchange rate regime – is financed either through the depletion of foreign exchange reserves or the printing of domestic money. Depending on the level of the central bank's foreign exchange reserves sooner or later the peg cannot be defended by the central bank anymore and the exchange rate is allowed to float. A depreciation of the domestic currency will occur which is accompanied by a sudden increase in the domestic price level. Under the validity of uncovered interest parity, perfectly mobile capital moves abroad and interest rates in the domestic economy rise. The same model can be used to analyse a situation with *perfect foresight* of the investors. Here, the agents anticipate the imminent abandonment of the peg and shift their portfolios accordingly. Then the exchange rate adjusts smoothly to its floating equilibrium value.

Critics of this kind of models argue that the government in this set-up is too passive, whereas investors optimise their portfolios actively. It was also criticised that deficit financing by depleting foreign exchange reserves exclusively is not very likely to happen. Furthermore, Krugman's and Flood & Garber's models do not

explain satisfactorily the extreme jumps in the exchange rate in crises periods.[3]

7.2.2 Second Generation Models

Obstfeld (1994) exploits the first generation models by adding certain reaction of the government to private investors behaviour. In other words, if a central bank assumes a currency attack it readjusts its policy even though the previous policy was consistent with the fixed exchange rate regime. Obstfeld concludes from the experiences of the severe attacks in the early 1990s on many European currencies in the Exchange Rate Mechanism that the costs of defending a fixed exchange rate system rise when people expect the peg to be given up. This is because the public anticipated in the past a devaluation of the exchange that has not yet taken effect. The demand for higher interest rates and wages goes along this anticipation, making the domestic firms uncompetitive and the debts too high at the non-devaluated level of the exchange rate. Therefore, the expectations of agents seem to be a very important factor in the development of a crisis and can put pressure on the economy (*self-fulfilling crises*). The timing of a crisis in these sort of models depends on the formation of expectations among agents. An attack on the exchange rate system is not likely to occur if there is only a big number of small investors on the market. This group is not able to coordinate their actions and take a strong position against a pegged currency.[4] Thus, second generation models open the opportunity for two different equilibria: the attack and the no-attack equilibrium.

7.2.3 Twin Crises Models

The term *twin* refers to the perception of the financial sector's importance for the development of currency crisis. This class of models states that a currency crisis is often triggered by a previous banking crisis, which can be observed best in the case of Mexico in 1994 (*Peso Crisis*). There is disaccord in academic literature on whether a banking crisis premises a balance of payment crisis or the other way around. In Mexico the banking crisis appeared first and the central bank tried to solve the excessive indebtness of financial institutions by printing money. This finally resulted in an abandonment of the peg and a depreciation of the Peso, which put even more pressure on the banks because their loans were mainly denominated in US-Dollar. One way or the other, both types of crises appear to be correlated with each other and reinforce themselves.[5]

[3] Saxena (2004)

[4] However, big investors sometimes gain the financial power to take massive position against weak currencies. A famous example is George Soros when he was speculating against the Hong Kong Dollar in 1998.

[5] See e.g. Kaminsky and Reinhart (1999).

7.2.4 Herd Behaviour

Apart from expectations formed on basis of the analysis of fundamental data there exists a further element in financial markets that presents an explanation for certain phenomena: *herd behaviour* of investors. The underlying idea is that individuals have a tendency to chose an action which is similiar to that of the majority of other individuals, even though they have come to a differing result by conducting their own analysis. This oddity can have several reasons; the most striking argument in favor of the existence of herding was made by Froot et al. (1992). They assume that a fund manager's penalty for missing a bull market is significantly higher than taking the loss during a bear market, but speculating on a bull market instead. This is due to the fact that in the latter case the entire "herd" of fund managers will be punished by betting on the wrong horse and the individual failure does not become obvious.

At a first glance, herding seems to lead to major distortions and inefficiencies on financial markets, but this behaviour can be completely rational for an individual. For instance, if the (institutional) investor is able to hide in the herd of small credit-restrained agents his actions and evaluations of the market remain unreaveled. This class of explanation is also known under the term *principal-agent problem*.

Critics of herd behaviour as an explanation for crises on foreign exchange markets argue that herding can only be one out of many factors pushing a currency system into a crisis. Agents are continuously adjusting their strategies to the arrival of new information and will not disregard them completely. Furthermore, agents sometimes have an incentive to commit strategical interactions. This important element of foreign exchange markets is suppressed by herding.[6]

7.2.5 Contagion

Contagion is a famous approach to explain crises that broke out not only in one country, but in an entire region. The "Tequila Crisis" (Mexico 1994), the "Asian Crisis" (seemingly starting off in Hong Kong in 1997) and the crisis of the European Currency System (starting with Soro's speculation against the British Pound) are prominent examples for contagion. Of course, every crisis had different roots but the consequences were mostly similar.

Contagion can be attributed to two sources. The first one is the existence of direct economic linkages between (regionally) interdependent countries. This was the case in the European crises where the attack on the British Pound led to a crisis contagion to most of the member countries of the European Exchange Rate Mechanism. The sharp depreciation of the Pound increased the competitiveness of the UK and put pressure on the French labour market and the trade balance.

[6] Saxena (2004)

Therefore, France was forced to give up the peg with the Deutschmark.

The second reason for contagion can be found in sociocultural considerations. The Latin American countries neither did exhibit strong trade or competition links with Mexico at the time of the crisis nor were they interconnected by some sort of monetary union. There is basically no real economic reason for Mexico to have triggered the crisis. Drazen (1999) offers the explanation of *political contagion*, i.e. the contagion of currency attacks arises only because of political objectives of the country. Another assumption is the homogeneity of culture in countries within a region (Saxena (2004) takes as an example the "latin temperament" which led to the contagion of the Tequila Crisis). These influences are very hard to measure quantitatively or to model theoretically. Hence, the real effect and the ultimate impact of these "soft" explanations is unclear.

7.2.6 Moral Hazard

The final explanation for the ocurrence of currency crises is *moral hazard*. Krugman (1998) introduced this idea in his 1998 mimeo "What happened to Asia?". In his model Krugman shows that a combination of an under-regulated banking sector and public debt guarantees in the banking system result in a moral hazard that leads to a massive attraction of capital and therefore to overinvestments.

Since credit loans are hedged against default through governmental bails, banks have an incentive to invest into high-risk (and often unprofitable) projects. Under these circumstances institutions do not risk any of their own capital and a possible bankruptcy will be absorbed by the government. This leads to excessive investment and through perfectly mobile capital the financial sector has instantaneous access to the world capital markets.[7] Cash shortages and re-financing of risky-projects are bridged by foreign lenders as long as the government bails for the credits, but once the foreign creditors are not disposed to re-finance the debt anymore the government has to stand in for the external liabilities. If the government then tries to bail out the deficit by printing money, the inflationary effect of this solution translates into an expectation of depreciation of the domestic currency and via the uncovered interest parity to a sharp rise of interest rates. Finally, the currency collapses.

7.3 The IMF International Financial Statistics (IFS) Database

The data used for the analysis in this chapter is obtained from the *International Financial Statistics* database provided by the International Monetary Fund (IMF).

[7] If this was not the case (closed economy), excessive investment would be reflected in extremely high interest rates, which, in turn, would regulate the lending. In the case of an open economy moral hazard in the domestic financial sector converts into real capital accumulation.

This database contains every country's most important macroeconomic data, such as time series that summarise balance of payments, trade and reserves, but also data on monetary expansion and contraction, government surpluses and deficits, production, prices and interest rates. The frequency of the data is monthly and quarterly since 1957; the annual records exist since 1948.

The following subjects are included[8]:

- exchange rates; international liquidity; monetary authorities
- commercial banks – reserves, foreign assets, government deposits; monetary survey – foreign assets, domestic credit, money, quasi-money
- interest, prices, production – discount rate, treasury bill rate, government bond yield, share prices, wholesale prices, consumer prices, wages, industrial production, manufacturing employment
- international transactions – exports, imports, balance of payments, goods services and transfers, long term capital, short term capital
- government finance – deficit or surplus, financing, total debt, intragovernmental debt national accounts
- exports, government consumption, gross fixed capital formation, increase in stocks, private consumption, gross domestic product, gross national expenditure, national income market prices, gross national product.

The IFS is a comprehensive database that has a pronounced advantage compared with the products offered by other data providers: the data are mutually consistent, i.e. it is not necessary to account or adjust, respectively, for country-specific statistical methods or the actual derivation of the data. This is especially important for the further use in the framework of the proposed SOM method in this chapter.

The analysis of this sort of data is usually subject to many obstacles resulting from the acquisition of the data. If many different countries are included in the exploration the researcher has to deal with values that are often denominated in national currency and/or that are calculated in dissimilar compositions.[9] Therefore, it is essential to preprocess the data in question very carefully in order to determine a common basis on which the analysis can be conducted.

The data we used for the analysis at hand consist of 55 countries and the year 1996 is utilised to train the map. The decision for the respective countries was taken

[8] Cf. `http://data.library.ubc.ca/datalib/finance/imf/ifs/ifswhat.htm`.

[9] For example, this is the case for money supply which is often given as M1 but some countries use their own aggregates.

on the basis of the availability and completeness of the data.[10] Furthermore, the consistency of computation of the economy-wide aggregates was taken into account. The selection of countries reflects a comprehensive mixture of different types of economies, disregarding the ruling political systems or their respective classification into first, second or third world categories. From this point of view, the data set we used for training is a good snapshot of the world economic situation in 1996. That year was chosen for training because it embraces both, very secure and extremely vulnerable economies and is therefore suitable for one of the goals of our analysis: crisis identification. The data used for the analysis are given in table 7.1. Note, that these values are raw data that have to be preprocessed before they can be used with SOM. The crosses behind column entries indicate a denomination of the listed values in national currency. In order to assure comparability we recalculated these values for the later computation of the SOM in terms of US-$ by means of the exchange rate given in the third column of table 7.1. The exchange rate is national currency per US-$ and the interest rate as well as the unemployment rate are percentages. The consumer price index (CPI) uses the year 2000 as the basis year (2000=100) for all countries and the population of each country is given in millions. The foreign reserves, assets and liabilities, the money supply M1 and the current and capital accounts are denoted in millions. We excluded the variables "exchange rate" and "population" from the training because they are not expected to contribute to the training or the later analysis in any reasonable way. However, since we used the exchange rate to compute dollar-nominated values from the domestic currency it is also given in table 7.1. The last column ("population") is given for completeness of the data set.

There are missing values in the table. When dealing with this sort of data this is often the case because the possibility of macroeconomic data extraction is limited due to the lack of statistical data ascertainment within the countries and/or the political reluctance to publish certain economic key data. However, these missing values do not significantly affect the outcome of the trained net. It is a particular advantage of the SOM that the method is able to deal with missing values without any problems. As it was said several times before, the SOM uses Euclidean distances to weight the net adjustment in the next updating step and to determine the reallocation of the nodes in the evolution of the system. Since the technique is an heuristic (and iterative) process, missing values in the data set do usually not take any significant effect on the trained net after a sufficiently high number of iterations. For a better clarity of the graphs every country is denoted by a two or three letter

[10] Earlier points in time exhibited a severe lack of key variables for many countries. This is due to the fact that many of the countries we deliberately included in our analysis are ex-communistic countries which had no interest in the publication of solid economic data. Moreover, it was necessary to choose a period **before** the European statistical data collection was merged, i.e. every year after 1997 was not convenient for SOM training.

code. This will facilitate the interpretation of the map's graphical representations.

7.4 Mapping Countries by Macro Data

Why is the Self-Organizing Map algorithm an adequate method to approach this problem? First, Kohonen's Maps are able to discover surprising structures in the data through unsupervised learning. This feature leaves the door open for an unprejudiced and unorthodox look at the data. Supervised learning narrows machine learning and may only discover features in a predetermined range. An akin characteristic of the method is the absence of any a-priori assumptions on the distribution of the data. Since SOM is a non-parametric method it has the advantage to freely adjust to the topology of training data until the net approximates the input up to a degree that is determined by the selection of the number of map units. Furthermore, SOM is a numerical data mining method and not a symbolic one. The method is capable to reduce the multidimensionality of data to a one-, two- or three-dimensional set which can be displayed and analysed graphically. The graphical analysis by means of Self-Organizing Maps can be seen as the first step to approach a complex data set. In many cases a rigourous statistical treatment of the data has to be accomplished as a second step in order to fully answer the research objectives. On the other hand, there are numerous cases for which traditional statistics does not offer the required tools, and hence, there is a need for methods such as Kohonen's Maps that fill this gap. For many statisticians and/or econometricians unsupervised learning methods or even neural network techniques seem to be too ad-hoc and without any substantial reasoning. But the opposite is true: SOM enhances the existing econometrics and statistics toolboxes and offer a flexible way to find formerly unknown data features. Thus, practitioners appreciate the ability of gaining a quick but comprehensive overview of complex data structures.

Keeping the above considerations in mind the SOM method appears to be a natural approach to analyse macroeconomic data of different countries. In the application area of SOM presented in this chapter of the thesis we focus on crises of economies. By the very nature of these events a large variety of different factors plays an important role for the crises to occur. Disregarding the crisis' character – currency or debt crisis – we hypothesise that the constellation of macroeconomic fundamentals indicate well before the crisis' actual occurrence the upcoming events. This assumption rests on the observation that crises within the last 30 years had some common features which can be perceived as *crisis indicators* (e.g. the development of foreign debts, money supply or the interest rate trajectories). SOMs form the shape of their nets by virtue of learning from past observations. The net adjusts to the data and represents an approximation of the distribution in the input

Country	Code	Ex. Rate	For. Reserves	For. Assets†	For. Liabilities†	M1†	Int. Rate	CPI	Unempl.	CA	Cap. Acc.	Pop.
Argentina	ARG	1,00	18103,90	19745,10	6292,57	19041,60	6,23	100,67	16,56	-6769,98	50,80	35,27
Australia	AUS	0,80	14484,70	22581,40	67,81	33043,10	7,20	93,30	8,23	-15648,70	964,33	18,16
Austria	AT	10,95	22864,80	268,31	0,09	431,15	3,19	95,03	7,03	-4890,37	78,35	8,08
Azerbaijan	AZU	0,82	211,28	176,68	144,50	234,74	20,00	19,90	0,90	-931,18	0,00	7,88
Belarus	BEL	15,50	469,15	7,41	6,49	n.a.	8,30	52,71	3,80	-515,90	101,10	10,22
Brazil	BRA	1,04	58322,90	69829,00	3434,74	29807,50	27,45	80,74	7,00	-23248,00	494,00	163,82
Canada	CAN	1,37	20422,40	7,46	0,28	77,92	4,32	93,23	9,70	3378,29	5833,41	29,59
Chile	CHI	424,97	14972,60	7230,88	3,40	2544,40	n.a.	83,53	5,40	-3082,65	0,00	14,62
China	CHN	8,30	107039,00	956,22	0,00	2756,38	9,00	8,32	3,00	7243,00	0,00	1230,98
Colombia	COL	1005,33	9844,97	10402,80	356,07	8992,76	28,37	58,73	11,90	-4641,24	0,00	39,26
Cyprus	CYP	2,13	1541,94	805,20	26,52	653,54	6,85	89,21	3,10	-467,71	0,00	0,74
Czech Republic	CZE	27,33	12351,80	373,08	17,98	n.a.	12,67	78,47	3,98	-4127,55	0,57	10,32
Denmark	DEN	5,94	14140,40	86,25	1,63	322,74	3,98	91,08	8,80	3089,67	n.a.	5,25
Ecuador	ECU	3635,00	1858,45	1992,18	323,37	1445,08	46,38	18,84	10,40	-54,84	14,36	11,59
Egypt	EGY	3,39	17398,30	65189,10	41807,20	44521,20	13,00	86,93	n.a.	-192,00	0,00	62,38
Esthonia	EST	12,44	636,82	7954,16	1572,31	11289,70	3,53	77,77	9,90	-398,24	-0,65	1,42
Finland	FIN	4,64	6916,26	36461,50	934,40	n.a.	3,63	93,20	17,94	5002,58	55,65	5,13
France	FRA	5,24	26796,00	308,79	n.a.	1814,71	3,73	96,04	12,06	20,56	1,23	58,44
Germany	GER	1,55	83177,90	132,18	15,60	804,99	3,27	95,29	11,50	-13,98	-2,18	81,92
Greece	GRE	247,02	17501,40	5253,88	196,50	n.a.	13,80	85,41	10,30	-4554,00	0,00	10,74
Hong Kong	HKG	7,74	63808,00	463,89	0,32	174,38	5,13	99,41	2,80	n.a.	n.a.	6,28
Hungary	HUN	164,93	9720,16	1614,92	57,75	1374,84	23,00	61,28	9,88	-1766,09	155,83	10,32
Ireland	IRE	1,68	8205,15	4958,52	0,00	n.a.	5,74	89,74	11,80	2048,62	784,82	3,64
Italy	ITA	1530,57	45948,20	108,65	1,91	863,29	8,82	92,21	11,18	39999,00	66,08	57,40
Japan	JAP	116,00	216648,00	17,49	0,00	188,15	0,47	98,64	3,35	65,79	-3,29	125,81
Kazakhstan	KAZ	73,30	1294,06	129801,00	46001,00	n.a.	35,00	64,85	13,00	-751,00	-315,50	15,68
Korea	KOR	844,20	34037,10	28172,70	321,70	39542,10	12,44	86,39	2,00	-23209,80	-597,60	45,41
Latvia	LAT	0,56	654,07	407,63	72,55	425,65	13,08	83,86	7,20	-279,78	n.a.	2,46
Lithuania	LIT	4,00	772,25	3345,32	1098,68	3610,90	20,26	85,89	6,20	-722,61	5,50	3,60
Luxembourg	LUX	32,01	73,68	7,52	0,91	111,40	3,29	93,79	3,28	2219,48	n.a.	0,41
Malaysia	MAL	2,53	27009,40	70737,30	15,90	60585,30	6,92	88,70	2,50	-4461,95	0,00	20,89
Mexico	MEX	3,07	17725,50	56333,00	21018,90	113634,00	23,58	22,77	2,20	-14888,00	0,00	85,95
Netherlands	NET	1,74	26767,00	62,72	0,14	514,99	2,89	91,14	6,60	21502,30	-2024,00	15,55
New Zealand	NZL	0,71	5953,47	8856,46	424,00	10565,50	9,38	95,21	6,10	-3890,82	1335,78	3,70
Nigeria	NIG	21,89	4075,72	180799,00	47322,50	235577,00	13,50	73,50	n.a.	3506,87	0,00	106,64
Norway	NOR	5,97	13232,00	78,75	n.a.	255,80	10,00	82,08	5,48	5031,90	17,25	4,26
Paraguay	PAR	2109,67	1049,29	2263,63	161,28	1570,67	16,35	72,05	8,20	-352,90	14,20	4,95
Peru	PER	2,60	10578,30	25933,40	3687,71	5408,53	18,16	80,00	7,00	-3646,49	21,56	24,26
Philippines	PHI	26,29	10058,20	308,76	76,10	652,81	12,77	78,70	8,60	-3953,00	0,00	69,87
Poland	POL	2,88	17844,00	52497,60	708,46	52330,90	20,63	65,87	13,20	-3264	94,00	38,63
Portugal	POR	156,39	15917,70	3250,46	6,90	4302,23	7,38	90,57	7,25	-5023,19	2238,34	10,06
Russia	RUS	5,56	11276,40	102861,00	71272,30	192373,00	47,65	30,42	9,90	10847,00	-463,00	147,95
Singapore	SIN	1,40	76846,80	107751,00	346,00	27040,00	2,93	96,96	3,00	13853,70	-138,72	3,59
Slovak Republic	SLR	31,90	3418,86	115665,00	53259,50	173350,00	n.a.	71,30	11,33	-2090,42	30,30	5,38
Slovenia	SLO	141,48	2297,36	329,81	0,24	495,84	13,98	73,99	13,90	55,44	-1,86	1,97
South Africa	SAF	4,68	941,56	11369,10	4586,11	147580,00	15,54	77,76	5,10	-1678,19	-47,13	42,79
Spain	SPA	131,28	57926,90	7959,90	129,10	19116,10	7,65	91,00	22,23	-2233,97	6636,59	40,02
Sri Lanka	SRL	56,71	1961,55	104899,00	34787,20	78202,40	24,33	75,07	11,30	-682,66	95,88	19,07
Sweden	SWE	6,87	19107,20	140,21	4,14	n.a.	6,28	98,27	8,04	5892,26	8,94	8,85
Thailand	THA	25,61	37731,20	988,75	0,11	423,69	9,23	86,01	1,10	-14691,50	n.a.	59,00
Turkey	TUR	0,11	16435,80	2028,70	1403,76	882,29	50,00	11,42	5,80	-2437,00	0,00	63,74
United Kingdom	UK	1,70	39896,20	27,40	28,36	26,15	5,96	89,70	7,10	-10,33	1,97	57,87
USA	USA	n.a.	64040,80	56,06	0,17	1104,50	5,00	91,0	5,42	-124,73	-0,74	272,51
Uruguay	URU	8,71	1250,96	18854,90	8275,08	10712,50	28,47	68,04	11,90	-233,40	0,00	3,24
Venezuela	VEN	476,50	11788,20	7242,40	1935,92	2777,65	16,70	34,18	11,80	8914,00	0,00	22,55

Table 7.1: Macroeconomics data of 55 countries in 1996

space. Once a network is trained by means of different macroeconomic variables from different countries it should be possible to use the trained map with a hitherto unknown country data sample and to map it onto the trained net by dint of the best-matching unit criterion. Depending on the neighbourhood of the newly assigned sample it should be possible to diagnose the current situation of the country at hand.

Moreover, depending on the economic history of the countries in the neighbourhood of the mapped item we can infer on the future development of the country of interest, or at least we are able to give propositions on the degree of hazard that the respective country drifts off into a crisis. In the following first the net is trained with preprocessed macro data and different zones are identified.

7.4.1 Training

Here, the batch training algorithm was employed (see section 3.2.5). This choice makes the training much faster without any loss of efficiency. For the neighbourhood function a gaussian kernel in the style of equation (3.11) is used and the initial radii are 2 and 1, respectively. Due to the construction of the batch training algorithm no definition of the learning rate function is necessary.[11] The original map was initialised linearly. As in previous applications of the SOM in this thesis the training was splitted into a rough and a fine tuning phase. Each of which encompassed 68 and 260 iterations, respectively.

7.4.2 Initial Visual Inspection

Figure 7.1 shows the image of the trained map. In figure 7.1 (a) the distribution of the countries and the concentration on certain neurons (or: *nodes*) is shown. For the clustering we used a 10×10 map, which leads to hundred neurons in total. Of course, there is not an input vector attached to every node because our data set includes only 55 countries. At a first glance, it is interesting that the distribution is not uniform over the map. A number of nodes does not exhibit any response, whereas others unify several countries (indicated by filling of the hexagons). This first picture is only a diagrammatical one, however, it provides a very good visual illustration of how the countries are grouped and if there are striking zones of unification.

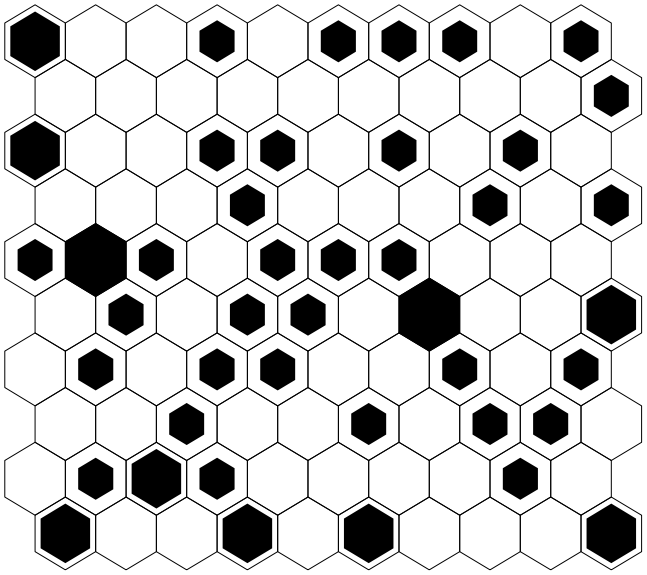
Turning to figure 7.1 (b) the picture becomes clearer. The cells are labelled with the country codes that are assigned to the respective neurons. Countries that are located close to each other are experiencing the same constellation of macroeconomic data. This does not necessarily mean that they are at the same absolute level of the key data, but their relative situation can be considered as being likewise. The plot provides a first, rough overview of the data at hand. It helps the user to get an idea

[11] Kohonen (2001) pp. 138

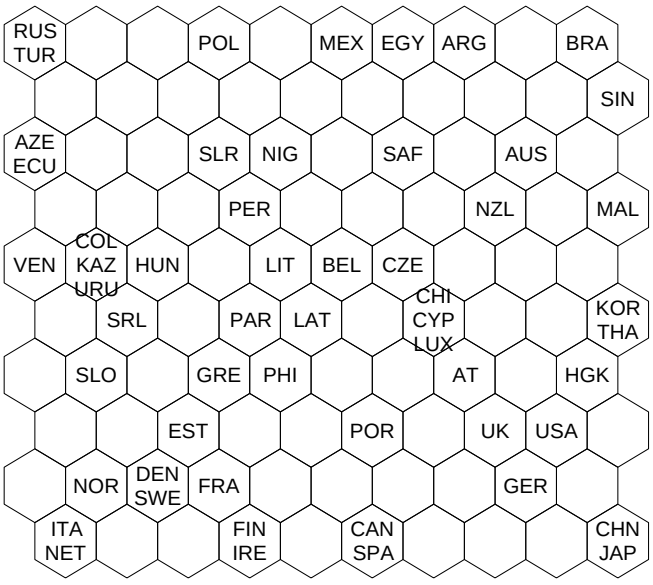
of the state of economies and in which neighbourhood they fit best. The relative location of the countries on the map appears to be reasonable from an economical point of view. For instance, in the lower right edge of the map Germany, the United Kindgom, Japan and the USA are located. These countries are neighbours on the SOM, which is a realistic classification under consideration of the historical context of the survey-year 1996. All of these countries belong to the group of so-called *first-world* countries and are members of the G-8 group. China is also attached to the neighbourhood of these countries. Even though not a member of the G-8 countries it is one of the most important and fastest growing economies in the world. Therefore, the group of countries can be assumed to underlie akin economic set-ups. Some of the countries are located in the same region of the map like the "big" economies mentioned before are certainly not comparable with e.g. the US-American economy in any absolute measure. But this does not necessarily mean that they are misplaced. In this application of the SOM we are interested in the relative constellation of macroeconomic data in order to infer on the economic health of a certain country. It is important to keep in mind that the data pre-processing squeezed the observations into the interval $[0; 1]$ and, therefore, absolute differences between the countries in question have been eliminated anyway. We will come back to the interpretation of the clusters later on.

It is noteworthy how the SOM was able to capture the similarity of different countries only by looking at the macroeconomic data without taking e.g. geographical information on the country into account. We did not feed the system with any sort of socio-cultural data, but to a large extend it resembles a geographical map quite well. In the lower part of the map the three Scandinavian countries Norway, Sweden and Denmark are direct neighbours. Also Finland is close to this Nordic cluster. Most of the EU countries are located in the lower half of figure 7.1 (b) and the relative distance between "old" member of the EU and candidate countries reflects the 1996 situation quite well. A further interesting finding is the location of Finland on the map. The SOM algorithm assigned Finland to a neuron a little apart from its geographic neighbours. Finland does not officially belong to the group of Scandinavian countries (even though, it is often added on to them) and the method seems to be capable to capture this special feature, i.e. Finland's exceptional position in the club of the Nordic countries is emphasised by this analysis.

Assuming the analysis works properly it is interesting to have a closer look at countries such as Poland or the Slovak Republic on the map in figure 7.1 (b). Both countries are relatively new members of the EU and originate from former Community of Independent States (CIS) countries. Since the map was trained with data from 1996 the large distance between the core countries of the EU and these two candidates is striking and reflects the necessity of convergence at that time. Other EU candidate countries (e.g. the group of the Baltic countries Lithuania, Latvia and Estonia) are already in 1996 much closer to the European club, with Estonia as



(a) Distribution of countries over a 10×10 map



(b) Location of countries on the map

Figure 7.1: Distribution and location of 55 countries on a 10×10 SOM

model candidate.

A further plausible attribution accomplished by the SOM algorithm are the two South and Central American clusters. The first one consists of Mexico, Argentina and Brazil and the second one is formed by Venezuela, Ecuador, Colombia, Uruguay and Peru. The main difference between the two clusters is the country size, which seems to have an important influence on the constellation of macroeconomic data. The distance between the big South and Central American countries and the "first world" countries may be a result from the aftermath of the 1994 Mexican "Tequila Crisis" that affected the entire subcontinent and left economical instability behind. Furthermore, in figure 7.1 the Korea and Thailand share one single neuron and Hong Kong is very close by. This is very interesting because the SOM map is obviously also able to cover the characteristics of at that time faster growing Asian economies. Moreover, most of the countries from the southern hemisphere are assigned to the upper part of the map, whereas in the lower half most of the countries located in the northern hemisphere can be found. For the northern countries this feature is not as noticeable as in the southern case, so this interpretation should not be overemphasised.

So far, we have discovered the general distribution of the countries on the map which gave us the opportunity to discuss the assignment of countries to the neighbourhood of other countries. However, an important feature of Self-Organizing Maps is the possibility to reveal the distances within a certain neighbourhood. This can be done either by computing the Euclidean distances for each node pair or it can be visualised by means of the U-Mat (see sections 3.3 and 3.4). Distances between neurons provide useful and interpretable information and lead to a comprehensive and more detailed description of the coherences of the net. Figure 7.2 shows an extension of the two graphs we discussed above. This plot is augmented by the application of a colour scheme which represents the absolute Euclidean distance over the entire map and can be used to better discriminate between different clusters.

The colour code is taken from the so-called *thermo map*. Blue translates into small and red into big heights. These are the two extremes on the colour-coded scale and figure 7.2 can be interpreted as the bird's eye two-dimensional view of a hilly landscape that approximates the density of the input data. At a first glance, the map can be divided into four parts. There are two homogenous triangles (upper right and lower right) which are subdivided by a heterogeneous trench (the blue offshoot). Furthermore, the upper left corner represents a further partition. The biggest part of the map is occupied by a homogenous area, which is coloured in blue. The neighbourhood of e.g. Paraguay is Latvia, Lithuania, Greece and the Philippines (among others) and the distance on the SOM is very small. Therefore, all of these countries exhibit a very high degree of similarity in terms of the constellation of their respective macroeconomic data. On the other hand Brazil and Singapore are

direct neighbours of Argentina, but contrary to the former case the neighbours are separated by a relatively large difference in height. Hence, even though the algorithm assigns the countries to their closest neighbours on the map, it is not possible to only interpret the locations without taking the length of their interconnections into account.

Figure 7.3 provides a good elaboration of details concerning the distances on the U-Mat. Additionally to the information given in figure 7.2 this graph is augmented with the z-axis which adds the third dimension to the plot and makes the structure more conceivable. The white lines on the net represent the links between the different neurons. Taking into account the plots discussed so far (i.e. figures 7.1, 7.2 and 7.3) the overall picture rounds off. From visual inspection we find that – apart from some mavericks – the SOM delivers a quite realistic order of the countries **without** using socio-cultural or geographic data. The grouping of different countries was only accomplished by only looking at macroeconomic data. Furthermore, the high-dimensional data set was compressed in such a way that it is possible to plot it in a three-dimensional space. The main characteristcs of the data are preserved and previously unknown features are discovered.

Apart from the analysis given above some of the assignments achieved by the Self-Organizing Map seem odd. For instance, there is no hard explanation why Australia and New Zealand are located in the direct neighbourhood of countries such as Malaysia, Singapore or South Africa. The economic situation of these countries in 1996 leads intrinsically to the assumption that they should have been assigned closer to Western European countries. The "miss-mapping" of Australia and New Zealand gives a hint for a suboptimal choice of variables. The construction of alternative data sets is beyond the scope of the analysis of this chapter, but might be subject of future research.

In the next subsection we will identify cluster by means of a more sophisticated cluster evaluation index: the *Davies-Bouldin Index*. This will give us the opportunity to conduct a more substantiated analysis.

7.4.3 Cluster Evaluation Index Analysis

The cluster validation procedures belong to the class of methods that measures the outcome of a clustering in an objective (and quantitative) function. These *validity indices* are usually used to evaluate the adequacy of the ordering accomplished by some clustering algorithm and if the structure of the data is reflected well by the outcome of the cluster analysis. The main goal of almost every validation method is to determine the best number of clusters in order to assure a reliable interpretation of the result.

The basic idea of cluster validation methods we introduce here consists of two parts: firstly the *intercluster* distance should be maximised and secondly the *in-*

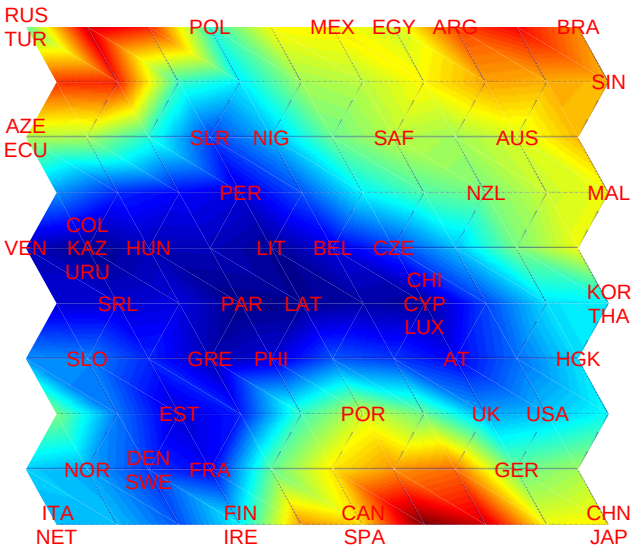


Figure 7.2: U-Mat of the 10×10 SOM with the respective country labels

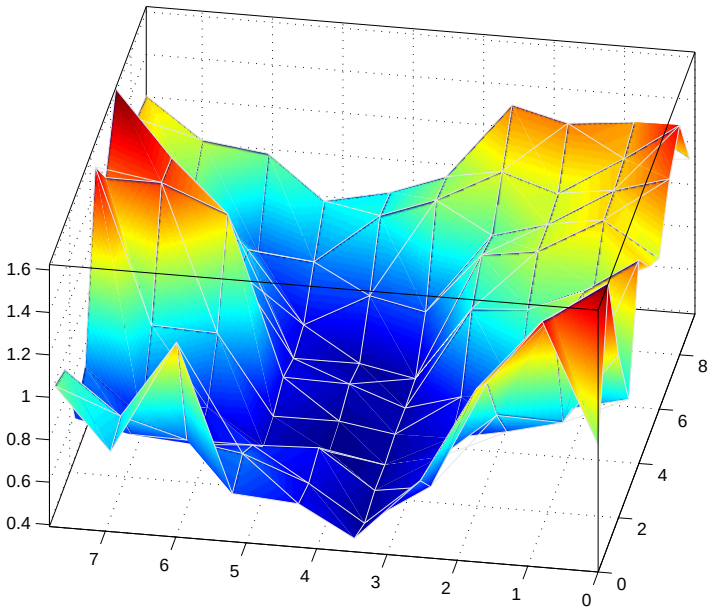


Figure 7.3: Distances on the net visualised in a 3D representation

tracluster distance should be minimised. There are several methods known in the literature, but we will mainly focus on the *Davies-Bouldin Index* (DBI).[12] A closely related measure is *Dunn's Index* (DI) which was developed long before DBI was published.[13] Therefore, we will start with the definition of DI and will then turn to DBI which is eventually used in our analysis.

Dunn's Index

DI's goal is to identify well separated and compact sets of clusters. Let U be a partition of a set X, i.e. $U \leftrightarrow X : X_1 \cup \ldots X_i \cup \ldots X_c$ with X_i as the i-th cluster of the partition. Furthermore, a distance metric $\delta(X_i, X_j)$ has to be defined (intercluster distance). This distance is usually calculated in terms of the Euclidean distance and has to satisfy the minimal necessary conditions as given in appendix A 3.1[14].

The second building block for DI is the intracluster distance $\Delta(X_k)$. This value is also known as the *dispersion measure*, which accounts for the heterogeneity of the members of one cluster. The idea is to maximise intercluster distances and to minimise intracluster distances. Therefore, a high DI value translates into good

[12] Davies and Bouldin (1979)
[13] Dunn (1974)
[14] Davies and Bouldin (1979)

clusters. Formally, this relation can be written down as:

$$DI(U) = \min_{1\leq i\leq c}\left\{\min_{\substack{1\leq j\leq c\\ j\neq i}}\left\{\frac{\delta(X_i,X_j)}{\max_{1\leq k\leq c}\{\Delta(X_k)\}}\right\}\right\}. \tag{7.1}$$

In this equation c is the number of clusters. Following the above considerations it is optimal to choose the number of clusters such that $DI(U)$ is maximised.

Davies-Bouldin Index

The DBI is based on the same ideas as the DI: the aim is to identify compact and well separated clusters. Although pursuing a similar strategy the DBI approaches the problem from a different angle. The Davies-Bouldin validation index aims at the minimisation of the value $DB(U)$. Taking the definitions as given in the previous section it can be formalised as

$$DB(U) = \frac{1}{c}\sum_{i=1}^{c}\max_{i\neq j}\left\{\frac{\Delta(X_i)+\Delta(X_j)}{\delta(X_i,X_j)}\right\}. \tag{7.2}$$

The smaller $DB(U)$ becomes, the further away are the respective centres of the clusters located from each other.

One big advantage of these two techniques is the absence of human subjectiveness. Older methods for the determination of clusters rely on human interpretation and subjective analysis of changes in the optimality parameters. The user of these methods has to decide on whether such a change is "large" or not. The DI and the DBI offer a solution to this subjectivity problem.[15] Due to the minimisation or maximisation, respectively, the determination becomes a simple optimisation problem which does not leave any space for interpretation errors.

We apply the DBI to the problem at hand and restrict the algorithm to find at most seven clusters. This restriction is imposed in order to assure interpretable results. Without limiting the number of allowed clusters the optimal BDI could result in a big number of clusters and this would make a meaningful interpretation of the SOM outcome impossible. The restriction of clusters not to be greater than seven was chosen arbitrarily, but seems to be a reasonable choice under consideration of the 55 countries in question. Figure 7.4 shows the result of the DBI computation. The maximum number of seven main clusters were not reached. Only five clusters are found by means of the index which we denoted by Cluster I-V. The clusters are not equally sized. Cluster III covers most of the trained SOM which is consistent with the colour-coded representation of the Kohonen Map in figure 7.2. The blue zone of the map corresponds to a very close neighbourhood between the neurons,

[15] Davies and Bouldin (1979)

i.e. the countries attached to these nodes exhibit a relatively similar constellation of the macroeconomic indicators we used in this analysis.

Most member countries of the European Community are located in Cluster III, Cluster IV and Cluster V, but especially in Cluster IV there are also countries such as China, Hong Kong, and the USA. The categorisation into five clusters is a very rough one by nature. Most of the countries in Cluster IV are rich industrial countries. Cluster III unifies mainly smaller first world countries. Cluster I corresponds to weaker economies, often coupled with a recent politic change and a relatively unstable or unexperienced government, respectively. In Cluster II the group consist of "the big South and Central Americans" plus Malaysia, Singapore, Australia and New Zealand. These countries do not belong to the group of the richest industrial economies, most of them experienced severe structural problems and had to enforce painful economic and political reforms. As mentioned before, an oddity in this respect is the presence of Australia and New Zealand in this cluster. Cluster V unifies only Canada and Spain. It is interesting that the DBI formed a particular cluster embracing these two countries only. Consulting figure 7.2 again one can easily see the dark-red colouring around Spain and Canada. Hence, the distances to their neighbours must be large. This, in turn, explains the DBI's decision to form an extra cluster in that area.

Furthermore, it is interesting to observe that the Kohonen Map is organised in such a way that a number of former members of the Eastern Bloc are not attached to the direct neighbourhood of Russia. This fact could be a hint that the Eastern European countries had already cut the (economical) cord to their former patriarch Russia in 1996. For instance, the three Baltic Republics are still direct neighbours, but their location on the map emphasis their exceptional position and their desire to uncouple from Russia. It is important to keep in mind that Cluster III is a very dispersed one, i.e. the countries unified there are not necessarily of a high degree of similarity. They only belong to this cluster in *relative* terms because they form a more compact and well separated group that minimises the DBI. In other words, the group members of Cluster III are more homogenous relative to each other than to elements outside their cluster. The DBI does not describe an absolute measure of quality, but a relative one. This emphasises a further important point when interpreting graphical representations of SOMs: not only the height (drawn in terms of colours) are crucial, but also the distance between two nodes on the two-dimensional plane.

7.4.4 Component Planes and Variable Correlation

In this section we will concentrate on the analysis of the specific variables contributing to the training. The selection of the variables is to be taken by the user of the method and depends on the research objective. As in any other quantitative

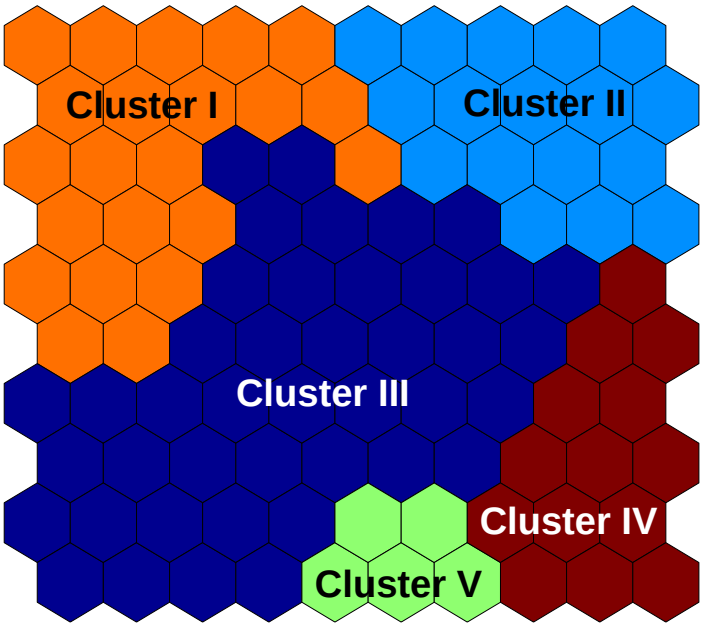


Figure 7.4: Clusters calculated by means of the Davies-Bouldin clustering evaluation index

analysis the right balance between the number of variables and the desired outcome is a difficult task which has to be treated carefully and depends on the experience of the researcher. Selecting too few variables will lead to a bad resemblance of the input data field and allowing for too many variables often ends up in overfitting and no reasonable interpretation of the result is possible. The data set used in the exemplary application in this chapter does claim to be the correct or complete choice of macroeconomic variables, but is serves to illustrate potential fields of application for Kohonen's Maps.

In order to obtain a deeper insight into the influence of the specific variables (henceforth *components*) we avail ourselves of a graphical method that is closely related to the U-Mat technique we utilised before. Speaking figuratively, we take the U-Mat of the trained map as given in figure 7.2 and slice it into different layers, each of which representing one component. That is why Guido Deboeck describes the *component planes* as a "sliced version of the SOM"[16] in which – again – the colour code serves to simplify the cognition of the information contained in the graph. The component plane representation opens up the opportunity to make explicit statements on the relative distribution of a specific data vector component. The colour coding is similar to the coding in the previous figures: the darker the colour the smaller is the respective value and vice versa. It is possible to find correlation between the components by simple visual inspection. If a component exhibits similar colour patterns like another one (i.e. the same part of two component planes look alike) these two components can be interpreted as being correlated with each other. Note that correlation in this context is not defined in terms of a stable positive or negative link between two variables. It is rather the case that two components in the analysis have a rectified influence on the Euclidean distance between the neurons on the trained net. Furthermore, picking the same neuron of the map on each component plane the relative values of an input data vector would become visible.

In figure 7.5 the *slices* for the case at hand are illustrated. Figure 7.2 is an aggregation of these nine graphs. The component planes of the specific variables are drawn in figure 7.5. It is possible to observe several similarities among some of the plots. The main findings are:

- There is an intriguing correlation between foreign assets and foreign liabilities.
- Foreign assets, foreign liabilities and money exhibit almost identical patterns.
- Reserves and Money are at least partly correlated with foreign assets and foreign liabilities.
- The unemployment rate and the CPI are the most heterogeneous elements of the U-Mat.

[16] Deboeck and Kohonen (1998) p. 54

- Money and interest rate exhibit at least in parts similar characteristics.

The correlation between money market interest rate and the money supply is evident and was also expected beforehand. Almost every macroeconomic model known in the literature presumes a strong link between money supply and the cost of lending on the interbank market. From this point of view it is not surprising that the SOM discovered a certain correlation between these two factors, but it is important to keep in mind that we did not postulate any functional form or coherency for the components. In other words, the Kohonen algorithm was able to discover a relation between these two variables without any previous knowledge. The connection was "learned" by the system.

The plane offers a nice insight into the constellation of the data belonging to certain countries. For instance, comparing the red coloured section of the CPI component plane with figure 7.2 we find that especially the Canada, Spain, China, Japan and also Germany exhibit a high relative value in terms of this variable. From the component plane of money supply we see a separated image with the boundary mark running diagonally. This appears to be an important element of the final map: the cluster analysis we accomplished in the previous subsection identified Cluster II as one of two South American partitions and money supply appears to play an important role for the location of Mexico, Egypt, Argentina and Brazil on the final map. Since we trained the SOM with data from 1996 this fact can be partly be justified through the aftermath effects of the 1994 Mexican "Tequila Crisis" which affected the entire subcontinent.

7.5 Trajectory of a Country: The Case of Japan

A trajectory is simply the sequential development of the BMUs over a specific period of time. A country's data of a certain year is presented to the codebook of the trained map and the BMU is determined by employing equation (4.12). The neuron is marked on the map. Then the data of the following year is used in order to compute the location of the next best-matching unit and so on. The marked neurons are then connected through lines so that a trajectory of the country on the map is obtained. The resulting image discloses the path of economic development a country has covered over a period of time relating to the training data in 1996. In order to assure the correct calculation of the BMUs we first have to de-normalise the trained map before identifying the map unit closest to the data from in question. In this way the actual BMUs are obtained and can be drawn on the original map.

Up to now, we have only accomplished a static analysis of the year 1996. As it was previously said, 1996 was suitable for the analysis due to the availability of data and because of the world economic situation, which consisted of both, very healthy

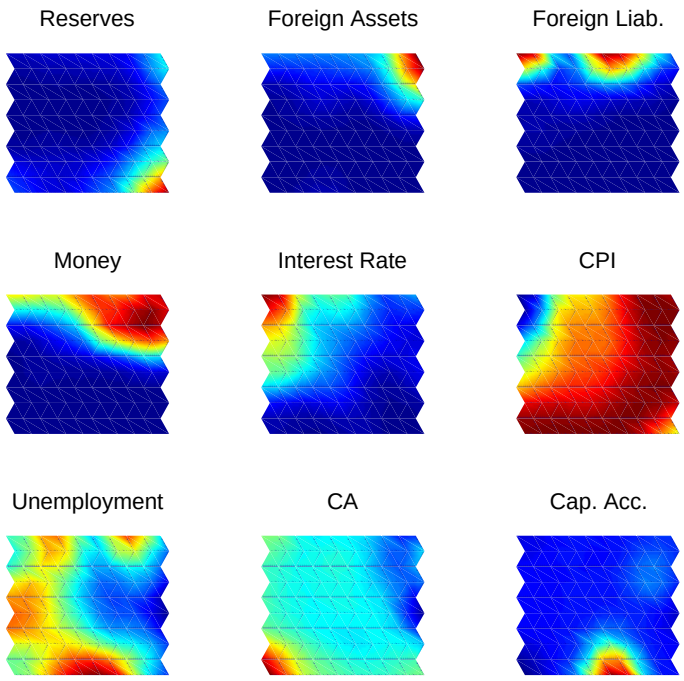


Figure 7.5: Component plane representation of the macrodata trained SOM

regions and also vulnerable areas, such as the Latin American subcontinent. In this part of the chapter we pursue a different purpose. In the following we will use the trained net of 1996 as a map to follow a country's development over time. To this end, we select a country from the list in table 7.1 and project its time series onto the trained SOM.

As a candidate country for our analysis we chose Japan. Japan has the advantage of representing a country that suffered from a highly regulated economy in the 1970s and experienced far-reaching reforms in the subsequent years. Furthermore, the Japanese economy was booming during the 1980s, experienced a crisis of the financial sector in 1989 and struggled with a banking crisis in 1997. One of the results of the first crisis was a lingering period of very low interest policy. Japan's economic history of the past three decades makes the country an interesting subject matter of this case study because it embraces healthy and vulnerable periods.

Figure 7.6 shows Japan's trajectory on the 1996 SOM for the time period 1975-2004. For a better readability of the graph we abandoned the colour coding and allowed only for labelling. The neurons are drawn diagrammatically but correspond exactly to the map as shown in figure 7.2. It is important to point out that the red line in plot 7.6 represents the *relative* trajectory, i.e. a time series is plotted onto the statically trained SOM based on the data of 1996. The interpretation of the progression has to be made against the background of the economical situation in 1996.

Starting in 1975 Japan's BMU is located in the middle of the lower part of the map. This is the region where through the training of the map countries such as Estonia, Greece, the Philippines, Denmark, Sweden and France are situated as well. According to the cluster analysis accomplished in the previous section of this chapter in the period from 1975-1983 Japan belongs to Cluster III and Cluster V, respectively, i.e. from the beginning of the sample the country is assigned to the regions of western economies. The starting point of the Japan on the map appears plausible because it is located between transition countries on one hand side and strong economies (e.g. Sweden and France) on the other side. After 1976 there is a sudden jump from the very left of the map in the direction of the bigger economies. This indicates Japan's transition to an industrialised, high-technology economy which took place during the late 1970s and the beginning 1980s. After this period the country remains in the lower right corner, but fluctuates between Cluster III, Cluster IV and Cluster V before settling down on a relatively stable node on the most-right lower corner of the map.

From the above considerations it is extremely difficult to infer on the state of Japan concerning an economic crisis. Japan's path on the map gives an indication for a certain change through the years included in the observation sample, but it is not clear if the fluctuations in the underlying variable can be explained through a crisis or by a normal structural change of the economy. In order to check for the

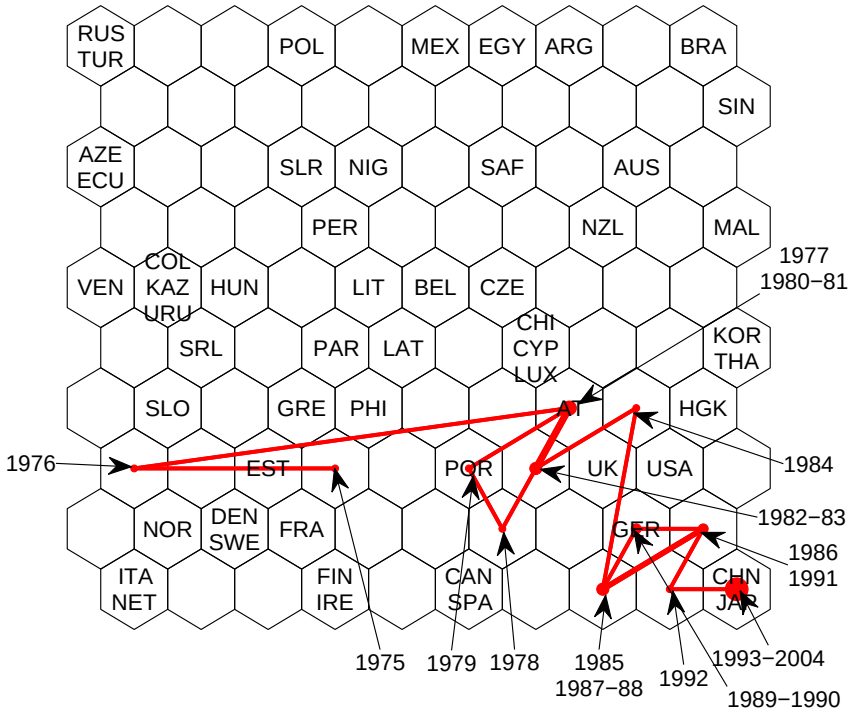


Figure 7.6: Trajectory of Japan on the trained map for the years 1975-2004

possibility of crisis patterns we apply the same method to other countries, namely the Philippines and Russia. Figure 7.7 shows the results graphically.

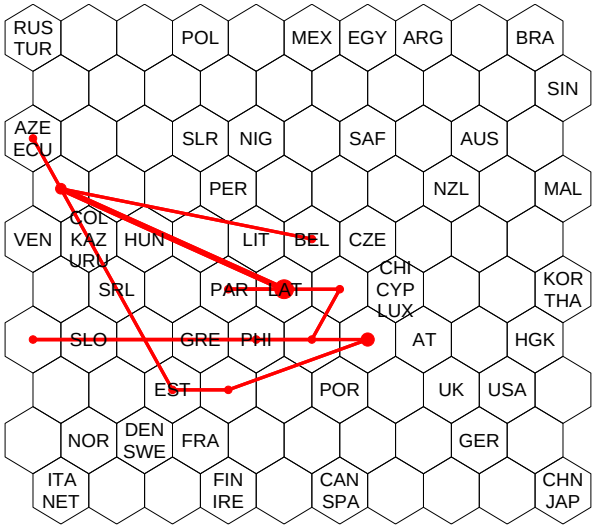
The two trajectories exhibit very different characteristics. In the first case (the Philippines) the movement on the map is limited to a relative narrow area (figure 7.7 (a)). The country's BMUs stay mostly in Cluster III which can be interpreted as a constant economic course. There are some tendencies in the direction of Cluster IV where most of the big economies are located, but eventually the trajectory goes back to the centre of Cluster III. This "outlier" belongs to the years 1999-2001. At that time the Philippines were counted to the *Asian Tigers*, a group of countries that experienced very high annual growth rates. After the burst of the bubble in 2000 it was these countries who suffered most from the worldwide economic slowdown. The second country we consider here is Russia (figure 7.7 (b)). For obvious reasons it was not possible to obtain any useful data before 1993 which leads to a shorter, but nevertheless very interesting path on the map. Russia features extreme jumps during the period in question. For most of the time the country stays in Cluster I and Cluster II. There is one maverick in the year 2000 and afterwards the trajectory heads back for Cluster II. In this respect the picture given by Russian data is less meaningful than it appears at a first glance.

As we already suspected from the analysis of Japan's trajectory it turns out to be extremely equivocal whether it is possible to identify crises zones on the a SOM-trained net with certainty. The SOM is able to group countries from the macroeconomic variables in a reasonable way but as an application for crises identification there is simply too much space for interpretation.

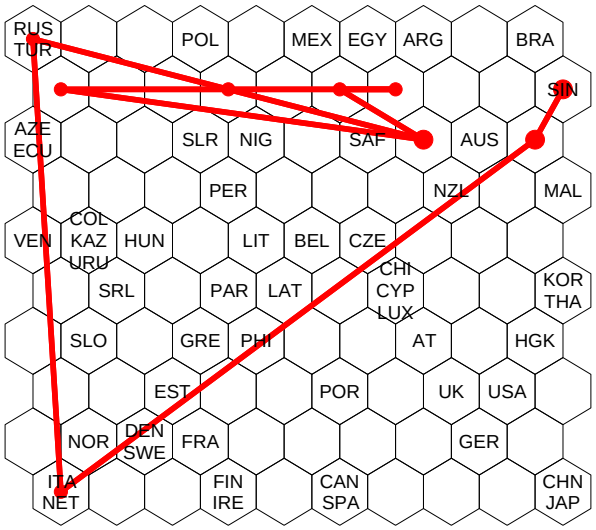
7.6 Out-of-Sample Mapping

The last question we pursue in this chapter is the assignment of a country that did not participate in the training of the map. That is, we are interested in the location of a country's BMUs when the data of the respective candidate is presented to the trained SOM. In the best case, the procedure will give us some insight into the power of generalisation. Obviously, if a country does not contribute to the convergence of the neural net it does not have any direct connections to the final outcome of the training. It will be interesting to see whether the final shape of the SOM is consistent with "external" data. Again, the procedure is the same as before. A test subject's data are taken and when necessary, the components are converted into US-$ in order to assure comparability. The BMUs is found by applying equation (4.12) again. We keep the description of the routine rather short because it was explained several times before.

As a test subject we chose Bulgaria. During the 1990s Bulgaria was struggling



(a) Philippines (1985-2004)



(b) Russia (1993-2004)

Figure 7.7: Trajectories of the Philippines and Russia on the 1996 map

to pick up the pieces of the economic effects of the breakdown of the Soviet Union. Furthermore, Bulgaria tried hard to catch up with its neighbours in order to fulfil the requirements for joining the European Community. Generally, Bulgaria was a very fragile economy during the 1990s. It is one of the of the candidate countries that exhibited one of the weakest economic situation and radical reforms were necessary to fight corruption and misgovernment. Therefore, it will be interesting to take a closer look at Bulgaria's trajectory for the period 1991 until 2004.

In fact, Bulgaria is mainly attached to Cluster III. The group of neighbouring countries in that area consists of Peru, Nigeria and the Slovak Republic. Moreover, it is located in the direct neighbourhood of Latvia, Paraguay and Belgium. Belgium does not fit in the overall picture very well, which – again – raises doubts about the accuracy of the method. Even though Cluster III is a more heterogeneous one the assignment of Bulgaria appears to fit adequately into the overall geographic picture. Its neighbourhood is in large parts consistent with what one would have expected before the analysis. All the countries (apart from Belgium) within the Bulgaria's neighbourhood exhibit analogue characteristics in the (relative) composition of macroeconomic indices. Furthermore, the BMU only attributes a country to the closest unit on the map and does not say anything about the distance in quantitative terms. Therefore, the result has to be treated carefully because even though the data are attributed to a certain neuron, it may be the case that it is only even further away from all other model vectors in terms of the Euclidean distance.

However, it is very interesting to see that Self-Organizing Maps are capable to generalise for larger data fields up to a certain extent. There is no need to offer the algorithm a complete data set in order to obtain results which can be used beyond the scope of the actual analysis. This is a very convenient feature of the SOM method because it offers the opportunity to analyse new and previously unknown data sets. However, the training input data have to be sufficiently large to allow for a smooth convergence of the neural net and should therefore include many different cases (higher heterogeneity assures higher adequacy of subsequent out-of-sample mapping) and should also be big enough in order to enable some reasonable interpretation of the constellation within neighbourhoods.

7.7 Chapter Summary

This chapter adopts an exceptional position in the context of this thesis. It is the only part that did not deal with any sort of time series forecasting or extracting geometrical symbols from price or return series, respectively. The core of the research accomplished here was the clustering of countries that exhibit similar constellation of macroeconomic data.

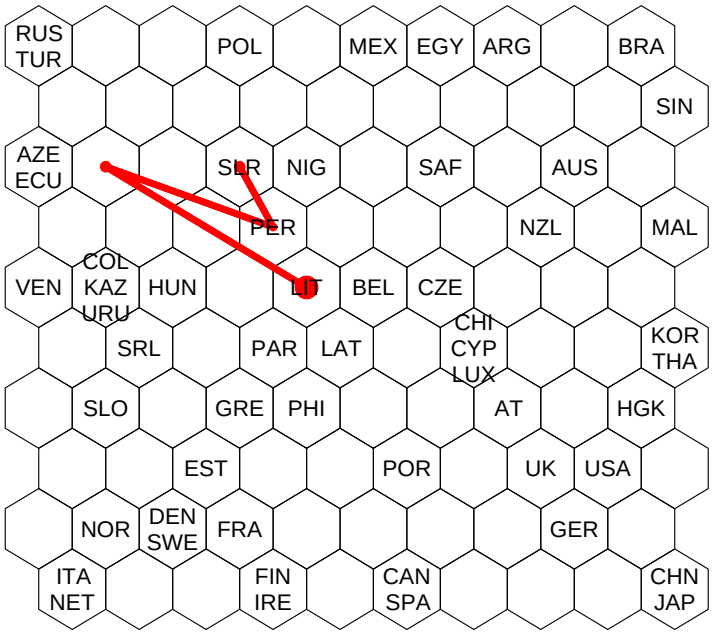


Figure 7.8: Trajectory of Bulgaria (1991-2004)

A map with 100 neurons was employed to find an ordering of those countries that reflected the true similarity as realistic as possible. Even though, we used only 55 candidate countries it was important to choose the map size bigger than the sample size. This leads to an even more striking effect of the unification of several countries on one single neuron (a map size larger than the sample size offers the possibility to disperse the countries over a larger area; if some countries are still clustered on only one neuron it emphasises the close similarity between these observations). The resulting net mostly reflected the expected results, but was also able to disclose some new information. Some of the countries seemed to be in the "wrong" neighbourhood and could not be interpreted easily. However, our guess is that the map generally clusters correctly and that the *relative* relationship to the neighbours was reproduced successfully. That is, not the overall economic state was evaluated through the SOM method, but the constellation of economic fundamentals.

Furthermore, we pursued to project a time series on the trained map and expected to recover a country's economic history from its trajectory – and at best its future, too. As an example we chose Bulgaria and could observe its (slow) convergence process towards a stable growth path after the economic breakdown in the early 1990s. The next step was to present an unknown country to the trained map in order to assess the accuracy of the net. We found that our testing country (Bulgaria) was located in quite a realistic neighbourhood and concluded that the training of the SOM seems to incorporate at least some degree of generalisation with respect to macroeconomic analysis.

Even though the results in this chapter are promising, the drawbacks of the method are obvious. First, as every neural net method the training of Kohonen's Maps is data intensive. This is in particular a problem for macroeconomic research because the ascertainment of data happens often in very large time intervals (annual or quarterly data). Moreover, due to the normalisation our guess is that a part of important information contained in the data is ignored by the method. The relative distances of the countries after normalisation do not reflect the true (and absolute) constellation to each other. This became obvious in some weird assignments on the trained map.

It is necessary to interpret the results very carefully and to perceive the SOM method as an excellent kick-off for groundbreaking theoretical modelling. This can be seen as the actual strength of Kohonen's algorithm: to uncover formerly unknown data structures that can subsequently be used to enhance theoretical models by adding these new coherency.

Summary and Outlook

October. This is one of the peculiarly dangerous months to speculate in stocks in. The others are July, January, September, April, November, May, March, June, December, August and February.

Mark Twain (1894)

The focus of the research in this thesis was the question whether it is possible to identify informative geometric patterns hidden in financial time series. Such regularities or recurrences, respectively, could be used to develop profitable trading schemes and/or to obtain a better insight into the mechanics of the data generating process. For economic research this is an extremely appealing idea because through the increasing adoption of modern information technology and ever-improving capabilities of software data mining has become much easier nowadays. The instant availability of vast amounts of data rises new challenges and requirements which have to be met by the augmentation of existing toolboxes for the technical analysis of economic data. Commercial data base providers offer not only a growing diversity of different data sets, but also quote a large bandwidth of frequencies. Especially high-frequency data (such as tick-by-tick data) often exhibit characteristics different from those which were only recorded on a daily bases and have therefore be treated in a particular way. This extensive availability of information opens a door to break new grounds in economics and econometrics. Hence, the task for future work in the fields related to quantitative economics will be the development of adequate data mining and data analysis tools.

In this thesis we concentrated on one approach to deal with the new challenges described above. Kohonen's Self-Organizing Maps are a powerful tool for information extraction from complex data sets. Developed in the 1980s, Kohonen's Maps were originally used to mimic the procedures in the human brain (brain maps), but soon this technique turned out to be very convenient for a variety of other applications as well (e.g. imaging, speech recognition, data base structuring, among others). Kohonen's goal was to create an algorithm that was able to work flexible but accurate, and additionally this algorithm was required to cope with the reduction of complex (non-linear) data sets without shading relevant information contained therein. This

specification sounds contradicting but Kohonen succeeded in inventing an algorithm which allows a flexible net to adjust to a set of input data. Such a net approximates and preserves the data set's topology without suppressing important information from the data. Moreover, via the application of the SOM method it is possible to depict even multidimensional data in graphical representations of at most three dimensions. In that respect, the SOM can also be interpreted as a dimension reduction technique that projects high-dimensional data into the space $\Re^n$ (with $n \in \{1, 2, 3\}$). A further advantage of Self-Organizing Maps is the absence of a given functional form (*unsupervised learning*). This feature increases the flexibility of the method to a large extent and makes an unprejudiced convergence process possible. Even though the SOM is a special form of neural networks, it differs in a certain point from methods such as the well-known Elman-Network or others: Kohonen's Maps are not a *black-box-approach*. There exists no hidden layer which is trained and that hampers the user to monitor the training process.

All of these SOM's inherent features make the technique an attractive and powerful tool for data-intensive research tasks. The possibilities of graphical representation are the strongest arguments in favour of the method and allow the user to easily grasp even with complicated coherences of multidimensional data sets. SOMs are in particular well suited for clustering and can be modified in order to discover at first invisible features in the time series. In the framework of this thesis we made use of this capacity and tried to construct a forecasting method by means of SOMs. The method's underlying assumption is that there exist certain geometric characteristics in the time series which recur within undetermined intervals of time and which could be used for explanation of phenomena on financial markets and/or for prediction of the future development of a time series.

In the first part, the thesis started with an overview of influential basic concepts in finance. Different types of traders were introduced and we took a closer look at the definition of the Efficient Market Hypothesis. The rest of the first chapter was devoted to shape an idea of financial pattern's information content and to give a general overview of data mining in finance. The second chapter provided an introduction to influential methods in econometrics and computational intelligence which prepared the ground for the application of SOMs. Since Kohonen's Maps are a special class of neural networks we dedicated several pages to the description of this class of models. The second chapter closed with some notes on data preprocessing and pattern recognition. Subsequently, the Kohonen Self-Organizing Maps Algorithm itself was described in great detail in the third chapter. The method is the core of this work and therefore an entire chapter served as the foundation of all the subsequent applications.

The second part of this dissertation presents applications of the concepts described in the previous chapters. The basis of all of the applications (apart from

that in chapter 7) is the construction of different patterns from past observations of a financial time series. These patterns were then used to identify recurrent trajectories within the historical values. The main idea is to use these recurrent patterns for prediction of the future trajectory of a time series.

In a first attempt we developed the concept of *Mean Return Curves* which can be interpreted as the average of all the return patterns within one cluster identified by the SOM. We then used these curves for applying a simple trading rule and compared the profitability of this rule with a plain buy-and-hold strategy. We found an ambiguous result: in certain situations the SOM-based trading rule outperformed the buy-and-hold rule, but this result did not hold systematically. In the subsequent chapter we extended the model with a probabilistic background. For this purpose two SOMs were used. The first one grouped the patterns exactly as in chapter 4. The second one was used to find groups *within* the respective families found during the first clustering. The respective family members were augmented by one future observation each and the empirical probability distribution was computed from the hit distribution of the second SOM. The empirical distribution were used in order to determine the one-step prediction in a out-of-sample setting.

The last application in this framework was the incorporation of regression methods in order to obtain local parameters for forecasting. This approach was divided into a least squares and a weighted least squares estimation. Surprisingly, the WLS approach did not lead to superior forecasting than the LS method. Moreover, we concerned ourselves with the question whether certain patterns discovered by the SOM are interconnected with eye-catching volatility phenomena. We were able to find some empirical evidence in favour of this assumption. When using price time series the SOM-based models are able to beat the simple random walk in forecasting. Moreover, the method exhibited promising results in the forecasting of signs. Future research as well as practical applications in this field should focus on this particular advantage rather than concentrating on the reduction of the RMSE.

The last chapter of this work employed the SOM algorithm in a somewhat different manner. In this case it was not used to filter geometric patterns for later prediction of future values. The aim was rather to cluster countries by means of their macroeconomic characteristics and to use the information from the trained map for financial crises diagnosis. With the variables which were chosen to train the net the self-organising process turned out to be quite capable of capturing the most important features. Under consideration of the fact that no a priori knowledge was offered to the algorithm the map exhibited a well-interpretable image of the economic coherences. We tried to exploit the information for interpreting the trained map in terms of crisis diagnosis. This field of application is extremely interesting to be used with economic data and the results of our analysis are promising.

Kohonen's Self-Organizing Maps are certainly just one item from the ever-

growing new toolbox of computational approaches in economics. Aim of this thesis was to propose some fields of application for the method and to introduce SOMs as a relatively new technique to economic analysis. Not much work has been done so far in this respect and there is still a long way ahead before the algorithm becomes a standard tool in the future. Apart from the pioneering work by Cottrell et al. (1998), Blayo et al. (2003) or Resta (2002) (among a few others) there is still a formidable amount of research to do which will ultimately result in the mining of the huge capability Kohonen's algorithm comprises. In order to integrate the SOM into standard software packages requires more exact methods to determine the hyperparameters, such as optimal number of neurons or the length of patterns. The main obstacle is the dependence of these hyperparameters on the respective problem. It seems to be unlikely to develop any rules-of-thumb to approach this problem and so far responsible calibration by experienced users appears to be the best alternative. For the future of the SOM in economic research it will be critical whether it succeeds to connect the method with the right "partner tools" in order to improve its capabilities. This can evolve into a tricky task, but it might also be a profitable one.

As mentioned above, SOM is especially a powerful tool for graphical representation of complicated data sets. Therefore, it can be used in particular for a first evaluation of the data's characteristics. Of course, this first assessment can never replace a rigourous statistical and/or econometrical treatment of the data, but it can often be helpful to identify the most striking facts by visual inspection, rather than by (often) time-consuming analytical methods. This can be a serious advantage in practice, where large transactions on financial markets are often a matter of seconds. Thus, when embedded into the right analysis environment Kohonen's algorithm can be helpful as a supporting method that complements the existing analytical toolbox.

As one of the motivations for this work we posed in the beginning the question whether it is possible to gain excess returns via the use of purely chartist methods. To answer this question to the full extent there are far too many different techniques that would have to be checked separately. Economists tend to favour decisions that are taken on the basis of economic models and fundamental values and there is no hard reason to depart from this viewpoint. On the other hand, many practitioners rely on the capability of chartist technique to allocate their investments. This is a gap between academia and industry which will hopefully be closed in the future. The relatively good performance of chartists on financial markets give reason to research their behaviour in greater detail.

Bibliography

Bacchetta, P., Mertens, E., and van Wincoop, E. (2006). Predictability in financial markets: What do survey expectations tell us? *Swiss Finance Institute Research Paper Series*, 06(15). URL: `http://ssm.com/abstract=935215`.

Blayo, F., Dablemont, S., Lendasse, A., Ruttiens, A., Simon, G., and Verleysen, M. (2003). Time series forecasting with som and local non-linear models - application to the dax30 index prediction. *WSOM'2003 Proceedings*, pages 340–345.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327.

Bollerslev, T. (1987). A conditionally heteroskedastic time series model for speculative prices an rates of return. *Review of Economics and Statistics*, 69:542–547.

Bollerslev, T., Engle, R., and Nelson, D. (1994). Arch models. In Engle, R. and McFadden, D., editors, *Handbook of Econometrics*, volume IV, pages 2959–3038. North Holland, Amsterdam.

Brock, W. A. and Hommes, C. H. (1997). Models of complexity in economics and finance. *Wisconsin Madison - Social Systems Working Papers*, (9706). URL: `http://www.ssc.wisc.edu/econ/archive/wp9706.pdf`.

Brown, S. J., Goetzmann, W. N., and Kumar, A. (1998). The dow theory: William peter hamilton's track record reconsidered. *Journal of Finance*, LIII(4):1311–1333.

Bulkowski, T. N. (2000). *Encyclopedia of Chart Patterns*. John Wiley & Sons, Inc., New York.

Campbell, J. Y., Lo, W., and MacKinley, A. C. (1997). *The Econometrics of Financial Markets*. Princeton University Press, Princeton, NJ.

Chen, S.-H. and Tsao, C.-Y. (2003). Information content of the trajectory-domain models. unpublished manuscript.

Cheng, S.-H. and Wang, P. P., editors (2002). *Computational Intelligence in Economics and Finance*. Springer, Berlin.

Cheng, Y. (1997). Convergence and ordering of kohonen's batch map. *Neural Computation*, 9:1667–1676.

Cochrane, J. H. (2006). The dog that did not bark: A defense of return predictability. *NBER Working Paper Series*. URL: `http://www.nber.org/papers/s12026`.

Cootner, P. H. (1964). *The Random Character of Stock Market Prices*. MIT Press, Cambridge, MA.

Cottrell, M. and Fort, J. C. (1987). Etude d'un processus d'auto-organization. *Annales de l'Institut Henri Poincaré (B) Probabilités et Statistiques*, 23(1):1–20.

Cottrell, M., Girard, B., and Rousset, P. (1998). Forecasting of curves using a kohonen classification. *Journal of Forecasting*, 17:429–439.

Darrat, A. F. and Zhong, M. (2000). On testing the random-walk hypothesis: A model-comparison approach. *The Financial Review*, 35:105–124.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1(2):224–227.

Dayhoff, J. E. and DeLeo, J. M. (2001). Artificial neural networks: Opening the black box. *Cancer*, 91:1615–1635.

de Long, J. B., Shleifer, A., Summers, L. H., and Waldmann, R. J. (1990). Noise trader risk in financial markets. *Journal of Political Economy*, 98(4):703–738.

Deboeck, G. and Kohonen, T. (1998). *Visual Explorations in Finance*. Springer, London.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74:427–431.

Diebold, F. X. (2001). *Elements of Forecasting*. South-Western, Cincinnati, Ohio, 2. edition.

Diebold, F. X. and Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3):253–263.

Diebold, F. X. and Nason, J. A. (1990). Nonparametric exchange rate prediction? *Journal of International Economics*, 28:315–332.

Ding, Z., Granger, C. W. J., and Engle, R. F. (1993). A long memory property of stock market returns and a new model. *Journal of Empirical Finance*, 1:83–106.

Donaldson, R. G. and Kamstra, M. (1996). Forecast combining with neural networks. *Journal of Forecasting*, 15:49–61.

Dornbusch, R. (1976). Expectations and exchange rate dynamics. *Journal of Political Economy*, 84:1161–1176.

Drazen, A. (1999). Political contagion in currency crises. *NBER Working Paper*, 7211.

Dunn, J. (1974). Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of the united kingdom inflation. *Econometrica*, 50(4):987–1008.

Erwin, E., Obermayer, K., and Schulten, K. (1991). Convergence properties of self-organizing maps. In Kohonen, T., Mäkisara, K., and Kangas, J., editors, *Artificial Neural Networks*, pages 409–414. Elsevier Science Publishers B.V., North-Holland.

Erwin, E., Obermayer, K., and Schulten, K. (1992a). Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55.

Erwin, E., Obermayer, K., and Schulten, K. (1992b). Self-organizing maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, 67:35–45.

Fama, E. F. (1965). The behaviour of stock prices. *Journal of Business*, 38:34–105.

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

Flood, R. P. and Garber, R. M. (1984). Collapsing exchange-rate regimes - some linear examples. *Journal of International Economics*, 17:1–13.

Fontenla-Romero, O., Alonsa-Btanzos, A., Castillo, E., Principe, J. C., and Guijarro-Berdiñas, B. (2002). Local modeling using self-organizing maps and single layer neural networks. In Dorronsoro, J., editor, *ICANN 2002*, LNCS 2415, pages 945–950. Springer, Berlin.

Frankel, J. A. and Froot, K. A. (1986). The dollar as an irrational speculative bubble: A tale of fundamentalists and chartists. *Marcus Wallenberg Papers in International Finance*, 1:27–55.

Frankel, J. A. and Froot, K. A. (1990). Chartists, fundamentalists, and trading in the foreign exchange market. *The American Economics Review*, 80(2):181–185.

Froot, K., Scharfstein, D., and Stein, J. (1992). Herd on the street: Informational inefficiencies in a market with short-term speculation. *Journal of Finance*, 47(4):1461–1484.

Gallant, A. R., Rossi, P. E., and Tauchen, G. (1992). Stock prices and volume. *Review of Financial Studies*, 5(2):199–242.

Granger, C. W. J. and Newbold, P. (1986). *Forecasting Economics Time Series*. Academic Press, San Diego, 2. edition.

Grauwe, P. D., Dewachter, H., and Embrechts, M. (1993). *Exchange Rate Theory: Chaotic Models of Foreign Exchange Markets*. Blackwell, Oxford.

Greene, W. H. (2003). *Econometric Analysis*. Prentice Hall, New Jersey, 5. edition.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press, Princeton, NJ.

Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13:281–291.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Saddle River, NJ.

Kaminsky, G. and Reinhart, C. M. (1999). The twin crises: The causes of banking and balnace of payments problems. *American Economics Review*, 89(3):473–500.

Kennel, M. B., Brown, R., and Abarbanel, H. D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45(6):3403–3411.

Kirman, A. (1993). Ants, rationality, and recruitment. *The Quarterly Journal of Economics*, 108(1):137–156.

Kohonen, T. (1982). Self-organized foundation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Kohonen, T. (1983). *Self-Organiziation and Associative Memory*. Springer, New York.

Kohonen, T. (2001). *Self-Organizing Maps*. Springer, Berlin, 3 edition.

Kovalerchuk, B. and Vityaev, E. (2000). *Data Mining in Finance – Advances in Relational and Hybrid Methods*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Krishnaiah, P. R. and Kanal, L. N., editors (1982). *Handbook of Statistics*, volume 2. North-Holland Publishing Company, Amsterdam.

Krugman, P. (1979). A model of balance-of-payments crises. *Journal of Money, Credit and Banking*, 11:311–325.

Krugman, P. (1998). What happened to asia? Mimeo. MIT.

Lux, T. and Marchesi, M. (1999). Scaling and criticality in a stochastic multi-agent-model of a financial market. *Nature*, 397:498–500.

Lux, T. and Marchesi, M. (2000). Volatiliy clustering in financial markets: A micro-simulation of interacting agents. *International Journal of Theoretical and Applied Finance*, 3:675–702.

Malkiel, B. (1992). The efficient market hypothesis. In Newman, P., Milgate, M., and Eatwell, J., editors, *New Palgrave Dictionary of Money and Finance*. MacMillan, London.

McNelis, P. D. (2005). *Neural Networks in Finance: Gaining Predictive Edge in the Market*. Elsevier Academic Press, Burlington, MA.

Meese, R. and Rogoff, K. (1983). Empirical exchange rate models of the seventies - do they fit out the sample? *Journal of International Economics*, 14:3–24.

Obstfeld, M. (1994). The logic of currency crises. *Banque de France, Cahier Economiques et Monetaires*, 43:189–213.

Ossen, A. (1993). Learning topology-preserving maps using self-supervised back-propagation. In Gielen, S. and Kappen, B., editors, *Proc. ICANN'93, International Conference on Artificial Neural Networks*, pages 586–591, London, UK. Springer.

Ozdemir, K. and Erkmen, A. M. (1993). A modified kohonen's neural network algorithm. In *Proc. WCNN'93, World Congress on Neural Networks*, pages 513–516, Hillsdale, NJ. Lawrence Erlbaum.

Pedrycz, W. and Gomide, F. (1999). *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, Cambridge, MA.

Pérez-Rodríguez, J. V., Torra, S., and Andrada-Félix, J. (2005). Are spanish ibex35 stock future index returns forecasted with non-linear models? *Applied Financial Economics*, 15:963–975.

Pesaran, M. H. and Timmermann, A. G. (1992). A simple nonparametric test of predictive performance. *Journal of Business and Economic Statistics*, 10(4):461–465.

Peters, E. (1994). *Fractal Market Analysis. Applying Chaos Theory to Investment & Economics*. John Willey & Sons, Inc., New York.

Principe, J. C., Wang, L., and Potter, M. A. (1998). Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. *Proceedings of the IEEE*, 86(11):2240–2258.

Resta, M. (2002). Self-organizing maps and financial forecasting: An application. In Seiffert, U. and Jain, L. C., editors, *Self-Organizing Neural Networks: Recent Advances and Applications*, pages 185–216. Physica-Verlag, Heidelberg.

Rhodes, C. and Morari, M. (1997). False-nearest-neighbors algorithm and noise-corrupted time series. *Physical Review E*, 55(5):6162–6170.

Ritter, H. (1993). Parametrized self-organizing maps. In Gielen, S. and Kappen, B., editors, *ICANN 93-Proceedings*, pages 568–577, Berlin. Springer.

Ritter, H. (1999). Self-organizing maps on non-euclidean spaces. In Oja, E. and Kaski, S., editors, *Kohonen Maps*, pages 97–110. Elsevier, Amsterdam.

Roberts, H. (1967). Statistical versus clinical prediction of the stock market. unpublished manuscript.

Rustichini, A., Dickhaut, J., Ghirardato, P., Smith, K., and Pardo, J. V. (2002). A brain imaging study of procedural choice. *Working Paper*. URL: `http://www.econ.umn.edu/ arust/ProcCh3.pdf`.

Saxena, S. C. (2004). The changing nature of currency crises. *Journal of Economic Surveys*, 18(3):321–350.

Serrano-Cinca, C. (1998). Let financial data speak for themselves. In Deboeck, G. and Kohonen, T., editors, *Visual Explorations in Finance*, pages 3–23. Springer, London.

Sirosh, J. and Miikkulainen, R. (1994). Self-organizing feature maps with lateral connections: Modeling ocular dominance. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proc. 1993 Connectionist Models Summer School*, pages 31–38, Hillsdale, NJ. Lawrence Erlbaum.

Sullivan, R., Timmermann, A., and White, H. (1998). Dangers of data-driven inference: The case of calendar effects in stock returns. *UCSD Working Papers*. URL: `ftp://weber.ucsd.edu/pub/econlib/dpapers/ucsd9816.pdf`.

Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, A. D. and Young, L. S., editors, *Dynamical Systems and Turbulence*. Springer, Berlin.

Taylor, M. P. and Allen, H. (1992). The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, 11:304–314.

Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition*. Elsevier Academic Press, Amsterdam, 3. edition.

Volterra, V. (1959). *Theory of Functionals and of Integro-Differential Equations*. Dover Publications, New York.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks. *International Journal of Forecasting*, 14:35–62.

Zschischang, E. (2005). *Genetisches Programmieren als neues Instrumentarium zur Prognose makroökonomischer Grössen: Anwendung auf Inflationsraten und Wechselkurse*. PhD thesis, Christian-Albrechts-Universität zu Kiel, Kiel.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich meine Doktorarbeit "Analysing Economic Data with Self-Organizing Maps – A Geometric Neural Network Approach" selbständig und ohne fremde Hilfe angefertigt habe und dass ich alle von anderen Autoren wörtlich übernommenen Stellen, wie auch die sich an die Gedanken anderer Autoren eng anlehnenden Ausführungen meiner Arbeit, besonders gekennzeichnet und die Quellen nach den mir angegebenen Richtlinien zitiert habe.

Kiel, 27. Juli 2007

(Lars Edler)