Advanced Neural Networks:

Finance, Forecast,

And Other Applications

Von der Wirtschaftswissenschaftlichen Fakultät der
Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor der Wirtschaftswissenschaften
— Doctor rerum politicarum —

genehmigte Dissertation
von
Dipl.-Math. Dipl.-Ök. Hans-Jörg Henri von Mettenheim
geboren am 24. Juli 1981 in Hannover

2010

Erstgutachter: Prof. Dr. Michael H. Breitner
Zweitgutachter: Prof. Dr. Klaus-Peter Wiedmann
Tag der Promotion: 16.12.2009

Prologue

This advanced textbook "Advanced Neural Networks: Finance, Forecast, And Other Applications" by Dr. Hans-Jörg Henri von Mettenheim (Master of Economics 2003, Master of Mathematics 2008, Dr. rer. pol. 2009, all from Leibniz Universität Hannover) is based on his Ph.D. thesis. Dr. von Mettenheim carefully develops both theory and implementation of so-called matrix prognosis models for general time series. All time series must be aligned on a regular time scale. Matrix prognosis stands for simultaneous prognoses of both many time series and multi-time steps. This matrix prognosis must be robust, i.e. the model must be easily trainable and adjustable and it must be — more or less — persistent in time.

Dr. von Mettenheim first builds his new, innovative mathematical theory of artificial neural networks with a shared layer perceptron topology. The essential difference to standard multilayer perceptrons is that only a single weight matrix is used. It shows that the training of the neural network models is much faster and much more robust, especially for challenging and real life problems. All mathematical and algorithmic considerations are well grounded and reproducible.

Secondly, Dr. von Mettenheim's highly efficient implementation is fully incorporated into the FAUN-Neurosimulator suite under development since 1996 at the Technische Universität Clausthal and the Leibniz Universität Hannover (FAUN = Fast Approximation of Universal Neural Networks). Highlights are, e.g., a wake on LAN coarse grained parallelization for large, inhomogeneous low budget computer clusters. These computer clusters enable an extremely fast training of matrix prognosis models even for very large and very difficult data sets.

Finally, Dr. von Mettenheim presents some convincing solutions for challenging and real life problems, e.g., a market value at risk model for a portfolio for the next 10 days based on 25 financial time series over the last 10 years.

This advanced textbook by Dr. von Mettenheim is well written, is well structured and is a must for scientists and practitioners solving challenging and real life problems. Covered fields include time series analyses and forecasts, artificial neural networks and also high performance neurosimulation.

Hannover, December 30, 2009

Prof. Dr. Michael H. Breitner

Dean of the School of Economics and Business Administration

Head of the Institute for Information Systems Research

Gottfried Wilhelm Leibniz Universität Hannover

Acknowledgements

First and foremost I thank my supervisor Prof. Dr. Michael H. Breitner who encouraged me in many ways to write this book. His ideas appear throughout the text and it is always interesting to discuss new topics with him. Especially, I am very grateful to him for giving me the opportunity to continue studying mathematics.

I also thank Prof. Dr. Klaus-Peter Wiedmann, my second referee, for accepting to examine my dissertation.

Many people contributed in different ways to this book. It is impossible to name them all. Without making a claim of completeness I mention some of them. I thank my colleagues Karsten Sohns and Marc Klages. We often had interesting discussions which gave me new insights on how markets work. I thank Cornelius Köpp for numerous technical advice. You three had to endure me all the time... I also thank Prof. Christian Dunis, director of the Centre for International Banking Economics and Finance at Liverpool John Moores University. His studies on neural networks helped me considerably. I had numerous fruitful conversations with Dr. Hans Georg Zimmermann and Dr. Ralph Grothmann from Siemens Corporate Technology. This pushed my imagination of what one can do with neural networks further. Thank you for this.

I dedicate this work to my family. Your continuous support is essential for me. Thank you!

Contents

Abstract

This book enhances the FAUN neurosimulator, FAUN = Fast Approximation with Universal Neural Networks. It implements a grid computing client. With this client spare computing capacity of user workstations can be reused. Tests on heterogeneous networks are presented. The grid computing client achieves a speedup of more than 95 percent. Remote wake up and shutdown saves power costs when the computers are not needed. Measurements show that power consumption is only one third of a comparable always-on scenario.

A novel neural network topology, the shared layer perceptron, is presented and analyzed. It is memory enabled and allows multi asset and multi step forecasts. The shared layer perceptron explicitly allows for uncertainty in the observed world. It incorporates uncertainty using hidden states. Convergence is robust and not sensitive to meta parameters.

Applications include modeling market value at risk, transaction decision support and investment. 25 financial time series spanning 10 years are used. The shared layer perceptron produces good or even very good results on equities, interest and exchange rates, and commodities. The shared layer perceptron forecasts multi asset time series well *by design*. Multi step forecasts enable market timing with high accuracy. The distribution of returns allows to evaluate the probable path of the portfolio within confidence bands. Performance is robust over a time span of 8 years, without retraining. Compared to benchmark models the shared layer perceptron produces *consistent* results.

It can be concluded that advanced neural networks can provide sustainable and economic competitive edge in today's financial markets.

Keywords: Neural networks, grid computing, parallelization, high-dimensional optimization, quantitative investment, decision support.

JEL classification: C45, C53

Abstract in deutscher Sprache

Im vorliegenden Buch werden verschiedene Erweiterungen des FAUN Neurosimulators eingeführt, FAUN = Fast Approximation with Universal Neural Networks. Es wird ein Grid Computing Client vorgestellt, der es ermöglicht, überschüssige Rechenkapazität in inhomogenen Clustern zu nutzen. Ein Speedup von mehr als 95 Prozent wird erreicht. Zur Einsparung von Energie hat der Nutzer die Möglichkeit, die Rechner ferngesteuert an- und abzuschalten.

Eine neuartige neuronale Netz Topologie wird entwickelt, das Shared Layer Perceptron. Diese Topologie verfügt über ein Gedächtnis und erlaubt Ensemble- und Mehrschritt-Prognosen. Das Shared Layer Perceptron erlaubt es, die Unsicherheit in der beobachtbaren Welt, explizit zu modellieren. Dies wird durch die Einführung verdeckter Zustände ermöglicht. Die Konvergenz ist robust.

Zu den vorgestellten Anwendungen gehören die Modellierung des Value At Risk, Transaktions-Entscheidungsunterstützung und Unterstützung bei Investment-Entscheidungen. 25 Zeitreihen über einen Zeitraum von 10 Jahren werden verwendet. Das Shared Layer Perceptron führt zu guten or sogar sehr guten Ergebnissen bei Aktien, Zinsraten, Wechselkursen und Rohstoffen. Das Shared Layer Perceptron prognostiziert durch seine Topologie Ensembles gut. Mehrschritt-Prognosen ermöglichen Market Timing Anwendungen mit hoher Genauigkeit. Die Verteilung der Experten Topologie liefert eine Abschätzung des wahrscheinlichen Portfolio Pfades. Die Prognose Leistung bleibt über einen Zeitraum von acht Jahren auch ohne Neutraining robust. Verglichen mit Benchmarks liefert das Shared Layer Perceptron konsistente Ergebnisse.

Schlagworte: Neuronale Netze, Grid Computing, Parallelisierung, hoch-dimensionale Optimierung, quantitatives Investment, Entscheidungsunterstützung.

JEL Klassifikation: C45, C53

Executive Summary

The content summarized for the rushing executive

This book enhances the FAUN neurosimulator. It implements a grid computing client. With this client spare computing capacity of user workstations can be reused. Remote wake up and shutdown saves power costs when the computers are not needed. A novel neural network topology, the shared layer perceptron, is presented and analyzed. It is memory enabled and allows multi asset and multi step forecasts. Convergence is robust and not sensitive to meta parameters. Applications include modeling market value at risk, transaction decision support and investment. 25 financial time series spanning 10 years are used. The shared layer perceptron produces good or even very good results on equities, interest and exchange rates, and commodities. Multi step forecasts especially enable market timing with high accuracy. The distribution of returns allows to evaluate the probable path of the portfolio within confidence bands. Performance is robust over a time span of 8 years, without retraining.

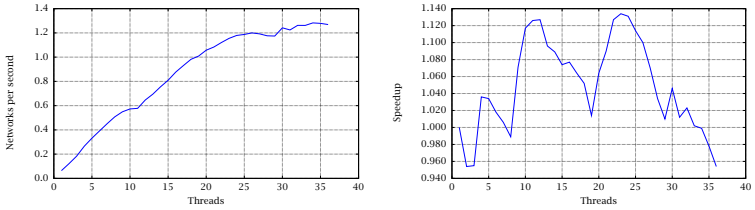
The content summarized on 8 pages

This book answers the research question «Can advanced neural networks provide sustainable and economic competitive edge in today's financial markets?» The author shows that neural networks are indeed capable of adding value to financial applications. To achieve this requires several components working together. Figure 0.1 on the following page provides an overview.

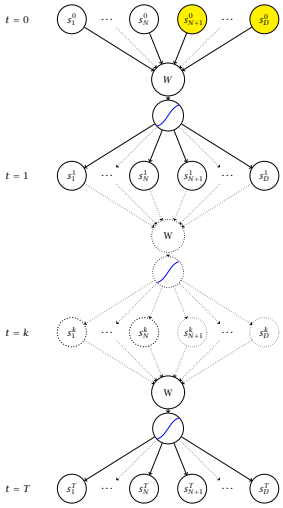
The research question considers several important aspects:

- *Advanced neural networks* are investigated. This is not a standard multi layer perceptron but a quite new topology, the shared layer perceptron, that allows easy modeling of multi dimensional financial time series.
- The models should be *sustainable*, i.e., more than just a statistical fluke, more than just a lucky hit. They should be robust over time.

Chapter 2 The FAUN grid computing client offers speedups of more than 95 %.



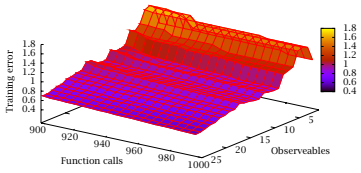
Chapter 3 The shared layer perceptron: a memory enabled neural network topology for multi asset multi time step forecasts.



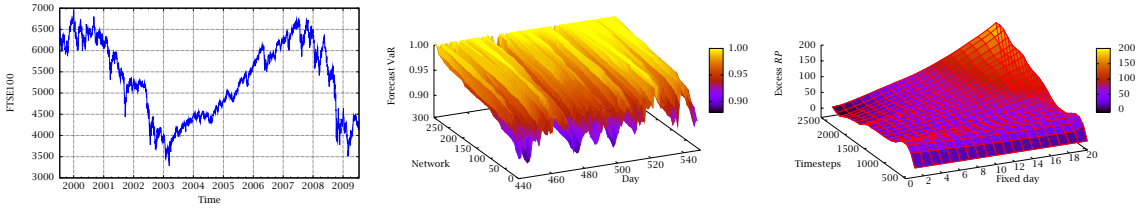
$$\mathbf{s}^{t+1} = \tanh(W\mathbf{s}^t).$$

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{t=1}^{T} l_i^t \mathbf{s}_j^{t-1}.$$

$$\mathbf{l}^t = (1 \ominus (\mathbf{s}^t)^2) \otimes (W'\mathbf{l}^{t+1} \oplus \varepsilon^t).$$



Chapter 4 Different financial applications are analyzed: market value at risk, transaction decision support, and investment.



Chapter 5 Conclusions. The shared layer perceptron topology

- is very robust. It performs regardless of asset or time span.
- adds economic value. It beats the benchmarks consistently.
- is versatile. It works well on a wide variety of financial applications.
- is easily parallelizable. It can be trained on off-the-shelf hardware.

Advanced neural networks provide sustainable and economic competitive edge in today's financial markets.

Figure 0.1: Steps towards advanced neural networks for financial applications.

- Computational requirements should be low, i.e., *economic*. Especially, computation should not require special high performance computers.
- The modeled applications should not be simple forecasts. They should offer real *competitive edge*.
- The focus is on *financial markets*.

Everything in this book is linked to the FAUN neurosimulator, FAUN = Fast Approximation with Universal Neural Networks. Since Michael H. Breitner started the FAUN project in 1996 there has been continuous development and improvement. The reader will find the following highlights:

- The FAUN neurosimulator now also uses fine-grained parallelization. This allows for easily achieved speedups on dual and quad core CPUs. End users are therefore enabled to utilize their workstations to full capacity without having to deal with the increased complexity of message passing software.
- FAUN now also features coarse-grained parallelization using an easy to install grid computing client. Via the web interface it is possible to use clusters of heterogeneous workstations. Spare computing capacity gets reused. Automatic wake up and shutdown saves power costs.
- FAUN is now well-equipped to handle time series problems. It uses a very innovative shared-layer perceptron architecture. A detailed analysis of the computational requirements for the gradient calculation is provided. The gradient calculation itself is presented extensively. Using reverse accumulation and matrix algorithms allows for very efficient computation.
- The examples are designed to provide a maximum of practicality. This includes not only the standard trading application but also market value at risk modeling and transaction decision support.
- The same dataset is used for *different* application. This offers the possibility to benchmark the performance of neural networks or more standard modeling procedures in different domains. The dataset spans 10 years. It includes bear and bull cycles and is not limited to a single up or down trend where most models perform well anyway. The models are very robust and work well without retraining over a period of 8 years.

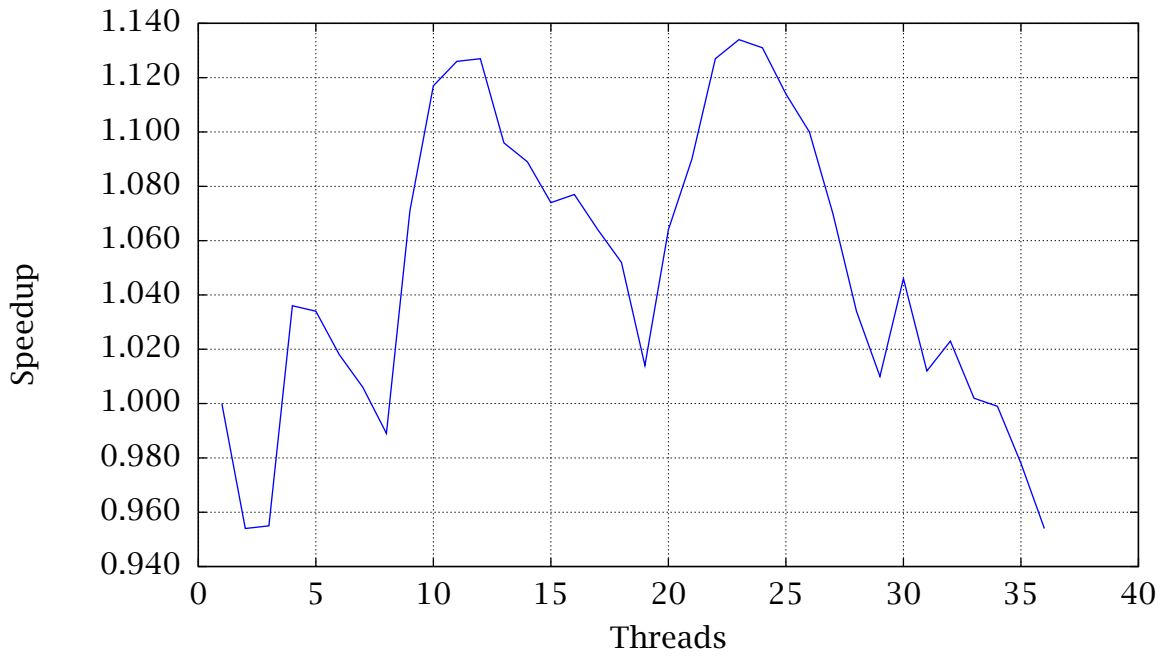


Figure 0.2: The FAUN grid computing client offers consistent speedup above 95 percent on networks of heterogeneous computers.

Grid Computing

Successfully training neural networks is also a matter of having enough computing capacity available. Neural networks are ideally suited for coarse grained parallelization. Communication requirements are low. You can distribute every single neural network to a separate thread. With the FAUN grid computing client spare computing capacity on user workstations is reused. There is no need to install specialized message passing software. The client is self contained. The achievable speedup is above 95 percent, see figure 0.2. This means that 95 percent of *theoretically* available computing power compared to a single thread is used.

It is a waste of energy to leave computers running continually. The FAUN grid computing client allows to wake up and shutdown computers remotely.

The update procedure is simple because all functionality is hosted on the server. An important feature of my client and server is that they are totally platform independent. Working combinations include the last releases of Debian and Ubuntu Linux, Windows 7, Windows Vista and Windows XP. This functionality is normally only implemented in commercial message passing software which necessitates a much more complicated setup.

The Shared Layer Perceptron Topology

The shared layer perceptron provides an elegant method to build *multi asset* and *multi step* models, see figure 0.3 on the following page. It augments the observable states $s_1, \ldots, s_N$ by hidden states $s_{N+1}, \ldots, s_D$. Hidden states allow the model to build up memory. Philosophically the shared layer perceptron acknowledges an incomplete view of the world. One does *not* assume that the «variables» are a perfect description of what happens. Rather one explicitly allows other «hidden» variables to influence the model. Training a shared layer perceptron implicitly also involves finding the right trajectory through the state space: for observable *and* hidden variables.

At each time step the state space is squeezed through the common weight matrix W and the following non linearity. This is an essential difference to standard multi layer perceptrons. Only a *single* weight matrix is used. This reduces the number of free parameters and also training times.

This topology produces at each time step *all* necessary *input* for the *next* time step. This simple mechanisms has two additional advantages. First, one *automatically* gets forecasts for *all* the observables. Second, one can *reuse* the forecast at the next time step and produce multi step forecasts.

Financial Applications

The dataset includes 25 financial time series from July 1999 to July 2009, i.e., 10 years of data. The dataset is divided into four asset classes: equity indices, interest rates, currency exchange rates and commodities. Interest rates are generally proxied by using yield curves. This dataset is challenging because it includes the boom and bust of the new economy, the bull market up to the credit crisis of 2007, the subsequent sharp bear market and even a small part of the ongoing recovery. Contrary to other studies this dataset truly represents all market cycles.

The first application models market value at risk. Interest lies in the worst expected portfolio value over the next 10 days. Figure 0.4 on page 17 shows a sample forecast for the FTSE 100 index. The goal is to model the *worst* returns as closely as possible. It turns out that the shared layer perceptron beats the benchmark historical simulation for *every* asset on a time span of 110 days. It still beats the benchmark *without* retraining on 8 years except for 5 cases. This allows institutions to reduce the margin of safety to an appropriate level.

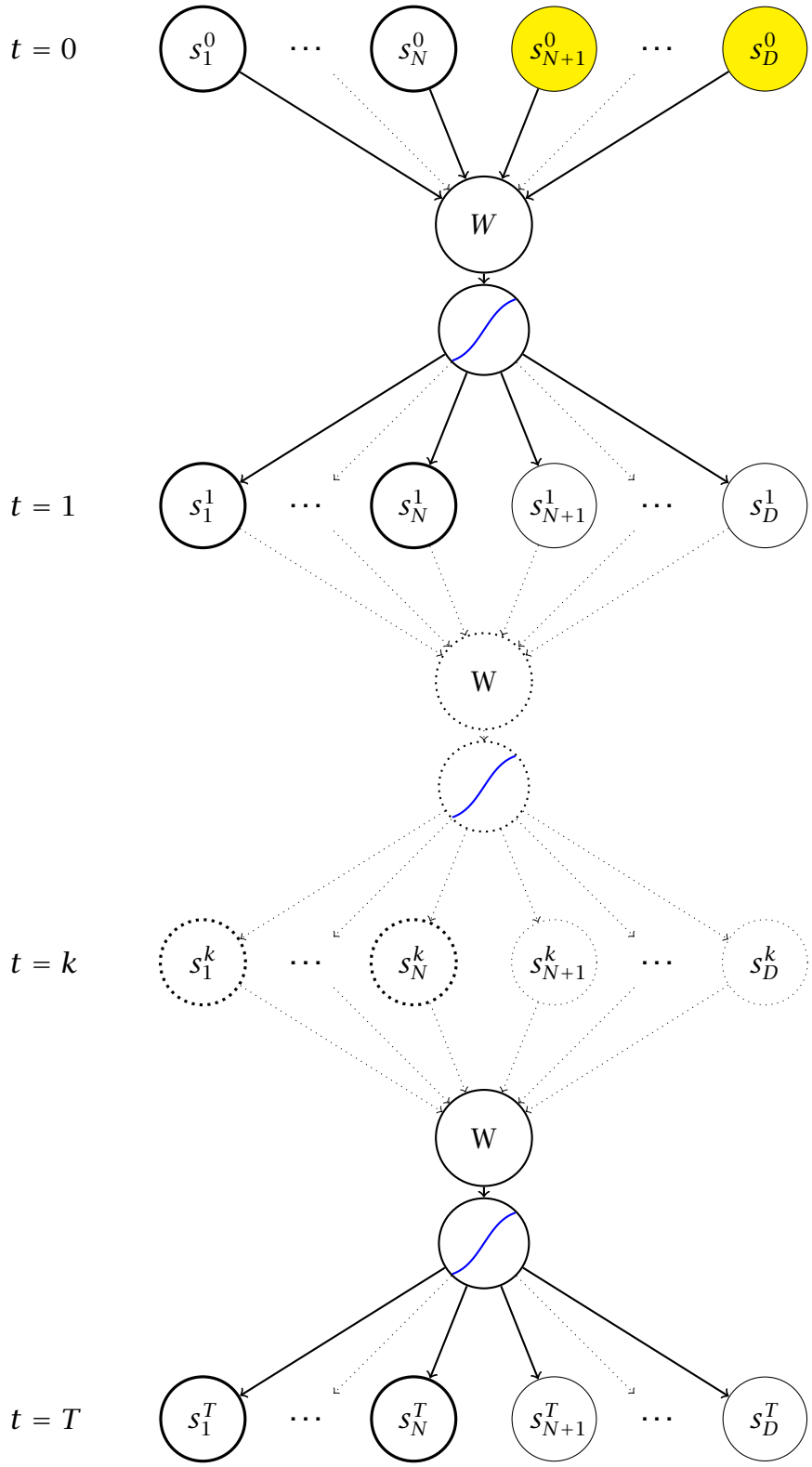


Figure 0.3: The shared layer perceptron for multi asset multi step models.

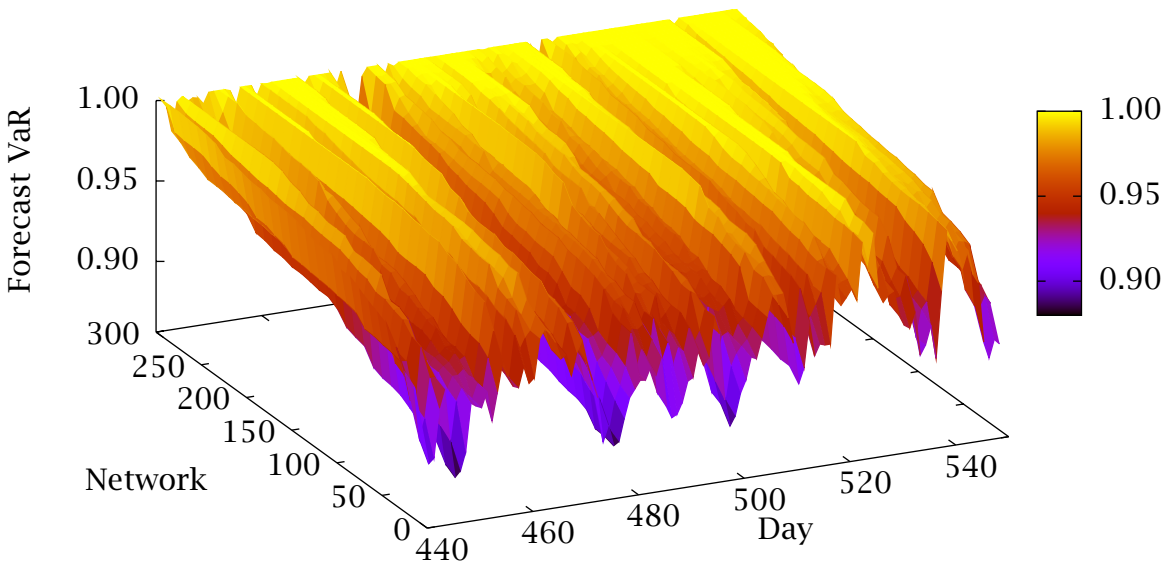


Figure 0.4: The shared layer perceptron topology models the probably worst portfolio value over the next 10 days for the FTSE 100 equity index.

Figure 0.5 on the following page shows a 20 days ahead ensemble forecast for the Baltic Exchange Dry Index. The target is to find an appropriate low entry point within the next month to secure low freight rates. One notes that the shared layer perceptron appropriately models the target: first down, then flat, then slightly up again. It does not *exactly* find the lowest price. However, the suggested lowest forecast is a sensible entry point. It is located *before* the index rises again. This models the typical challenge of a corporate treasurer: regular investments on a monthly basis. Again, the shared layer perceptron beats *every* fixed day strategy for *every* asset on 110 days. It is still very successful without retraining on 8 years.

The last application focuses on correctly forecasting the sign of next day returns. The shared layer perceptron is benchmarked against a naive strategy and a moving average strategy. The shared layer perceptron performs well or very well across a broad range of assets. Results are especially satisfactory on equities and currencies. It does not always beat the benchmark strategies. But it is at worst second best and shows very consistent returns. The benchmarks, however, show fabulous gains followed by catastrophic losses. The shared layer perceptron works robustly on the shorter and longer time span.

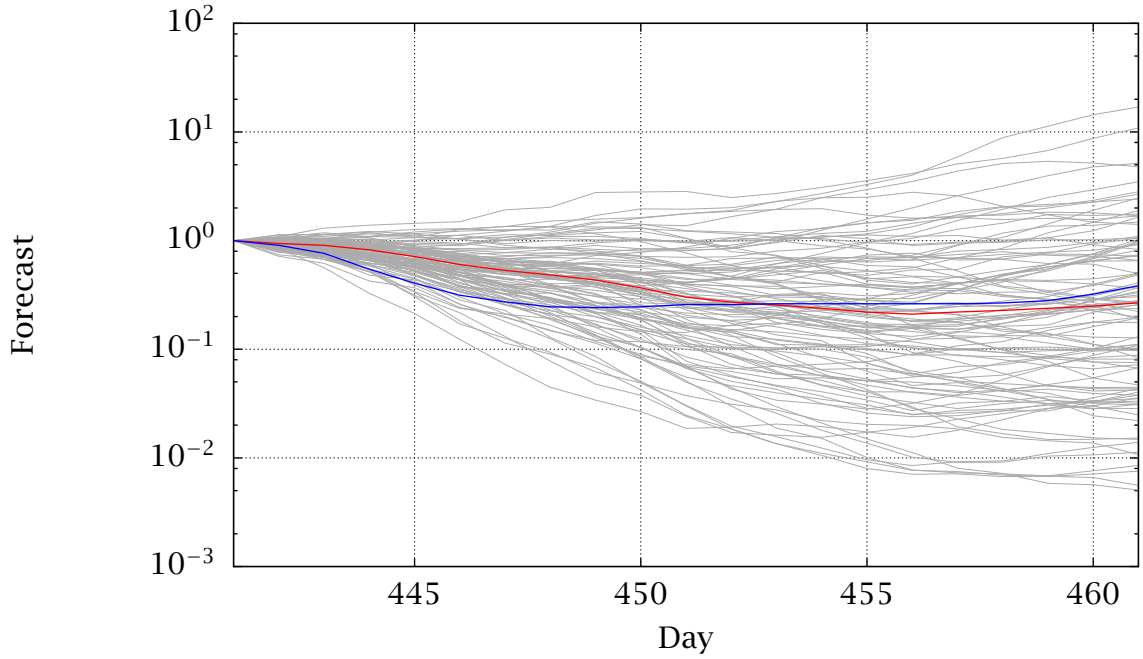


Figure 0.5: The shared layer perceptron forecasts the path of the Baltic Exchange Dry Index over the next 20 days.

Conclusions

The shared layer perceptron is a very robust model. It performs well over different asset classes. It also adapts to different market circumstances and shows consistent performance for long and short time spans without retraining. The shared layer perceptron offers a *unique* way to model a market ensemble:

- The multi step forecasts give a complete view on the portfolio value path.
- A *single* model is used. With an expert topology one gets every percentile of the underlying distribution *for free*.
- One will be more confident to use a model that works well over a broad range of assets. The shared layer perceptron works for *all* inputs *by design*.

Training the networks using coarse grained parallelization and the FAUN grid computing client provides a cost efficient and failsafe path to neural network modeling. Using the client does not require additional setup. The shared layer perceptron topology adds value to financial applications. The author recommends it as an important addition in the modeler's and forecaster's toolbox.

Nomenclature

acf	Autocorrelation function
AD	Automatic Differentiation
ADF	Augmented Dickey-Fuller test
ANN	Artificial Neural Network
BBA	British Bankers' Association
BDI	Baltic Exchange Dry Index
CAC	Cotation Assistée en Continu
CCI	Continuous Commodity Index
$cERP$	Cumulated Excess Realized Potential
CRB	Commodity Research Bureau
$\partial_{i,j}$	Partial derivative with respect to $w_{i,j}$
DAX	Deutscher Aktien Index
DRb	Distributed Ruby
ε^t	Local error at time t
ERP	Excess Realized Potential
EUR	Euro
EURIBOR	Euro Interbank Offered Rate
FAUN	Fast Approximation with Universal Neural Networks
FMADD	Floating Point Multiplication and Addition

GBP	Great British Pound
GUI	Graphical user interface
ICE	Intercontinental Exchange
JPY	Japanese Yen
LBMA	London Bullion Market Association
LIBOR	London Interbank Offered Rate
MPI	Message Passing Interface
N	State space dimension
NASDAQ	National Association of Securities Dealers Automated Quotations
NPSOL	Nonlinear Programming Solver
OAT	Obligation assimilable du Trésor
OTC	Over The Counter
pacf	Partial autocorrelation function
PVM	Parallel Virtual Machine
RP	Realized Potential
SFR	Swiss Franc
SLP	Shared Layer Perceptron
SQP	Sequential Quadratic Programming
SSD	Sum of Squared Deviations
T	Number of time steps
USD	US-Dollar, i.e., United States Dollar
VaR	Value at Risk
WOL	Wake On Lan

1 Introduction

Neural networks are an emerging modeling tool in the context of financial decision support systems. Most presented applications use standard topologies like multi layer perceptrons. Often, training occurs on personal workstations with simple algorithms. This has drawbacks:

- Multi layer perceptrons are conceptually not especially well suited for modeling and forecasting financial time series. One can do it, of course. But is does not come as especially *natural*. This results in people rejecting neural networks as unintuitive *black boxes*. This book intends to show: They are not!
- Training a neural network, especially with unsophisticated algorithms, *takes its time*. Mostly minutes but sometimes also hours or even days. This results in people rejecting neural networks as resource hogs. Again, this book intends to show: They are not!

The essence of this book consists in demonstrating the two bold statements above. It wants to show that neural networks can be *very natural* to use for forecasting financial time series. And it to presents some ways how the computational requirements of neural networks can be reduced quite effectively using advanced optimization algorithms and grid computing.

The author has been involved with neural networks for the past decade. But, when talking to most other people, the two objections cited at the beginning are often encountered. This is especially true when talking to people from the finance sector. That's why the author decided to answer the question:

«Can advanced neural networks provide sustainable and economic competitive edge in today's financial markets?»

The reader will note that the research question considers following important aspects:

- *Advanced neural networks* are investigated. This is not a standard multi layer

perceptron but a quite new topology, the shared layer perceptron, that allows easy modeling of multi dimensional financial time series.

- The models should be *sustainable*, i.e., more than just a statistical fluke, more than just a lucky hit. They should be robust over time.
- Computational requirements should be low, i.e., *economic*. Especially, computation should not require special high performance computers.
- The modeled applications should not be simple forecasts. They should offer real *competitive edge*.
- Finally, the author has to limit the scope of this book and focuses on *financial markets*.

To be specific, the book offers the following highlights:

- The FAUN neurosimulator now also uses fine-grained parallelization. This allows for easily achieved speedups on dual and quad core CPUs. While coarse-grained parallelization is still more efficient when different CPU *sockets* are involved the typical case of a user workstation profits from additional cores. End users are therefore enabled to utilize their workstation to full capacity. They do not have to deal with the increased complexity of message passing software.
- FAUN now also features coarse-grained parallelization using an easy to install grid computing client. Via the web interface it is possible to use clusters of heterogeneous workstations. Spare computing capacity gets reused. Automatic wake up and shutdown saves power costs.
- FAUN is now well-equipped to handle time series problems. It uses a very innovative shared-layer perceptron architecture. A detailed analysis of the computational requirements for the gradient calculation is provided. The gradient calculation itself is presented extensively. Using reverse accumulation and matrix algorithms allows for very efficient computation.
- The examples are designed to provide a maximum of practicality. This includes not only the standard trading application but also market value at risk modeling and transaction decision support.

- The same dataset is used for *different* application. This offers the possibility to benchmark the performance of neural networks or more standard modeling procedures in different domains. The dataset spans 10 years. It includes bear and bull cycles and is not limited to a single up or down trend where most models perform well anyway. The models are very robust and work well without retraining over a period of 8 years.

1.1 Guidelines for Readers of this Book

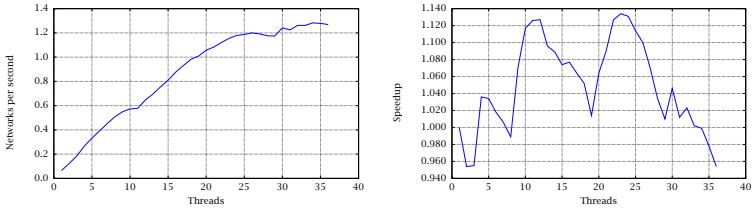
Figure 1.1 on the next page outlines the flow of this book. If the reader is in a hurry to grasp the basic concepts, the reader should just look at the figure. If there is a little more time, the summary in 150 words which is the first part of the executive summary at the beginning should be read. Then, the executive summary is the next best guess. It already provides some details.

The book is organized as follows: The remainder of this chapter is devoted to a motivation and some basic explanations of what the reader can expect from this book.

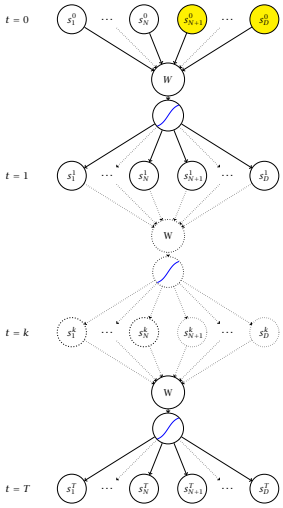
Chapter 2 introduces the FAUN neurosimulator. After a brief recapitulation of FAUN history this chapter gets quite technical. It describes in detail different kinds of parallelization for FAUN. The reader will meet fine and coarse grained parallelization — both useful techniques in their own right. Considering coarse grained parallelization the FAUN grid computing client is worth mentionning. It allows to easily distribute workload on a network of heterogeneous workstations. The reader will find a detailed analysis of speedup in different scenarios.

Chapter 3 presents the shared layer perceptron topology. As this topology is probably different from what is generally known as being a «neural network» the author motivates the shared layer perceptron philosophically, too. When using the shared layer perceptron one views the world in a more humble way. One especially acknowledges that one is *not* omniscient. Nevertheless the shared layer perceptron is first and foremost hard mathematics. That's why the remainder of the chapter deals with training it. The partial derivatives are explored and how to compute them efficiently with matrix algorithms. Teacher forcing and noise are presented as techniques which help to improve training. The next section details the sequential quadratic programming method used for optimizing. And the last section analyzes convergence depending on various meta parameters of the neural network.

Chapter 2 The FAUN grid computing client offers speedups of more than 95 %.



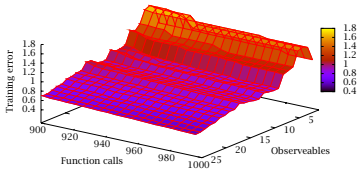
Chapter 3 The shared layer perceptron: a memory enabled neural network topology for multi asset multi time step forecasts.



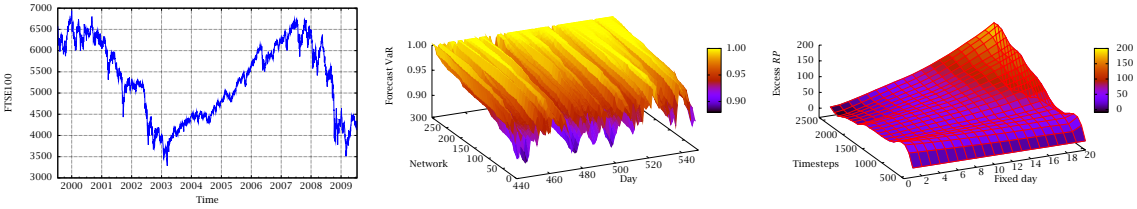
$$\mathbf{s}^{t+1} = \tanh(W\mathbf{s}^t).$$

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{t=1}^{T} l_i^t \mathbf{s}_j^{t-1}.$$

$$\mathbf{l}^t = (1 \ominus (\mathbf{s}^t)^2) \otimes (W'\mathbf{l}^{t+1} \oplus \varepsilon^t).$$



Chapter 4 Different financial applications are analyzed: market value at risk, transaction decision support, and investment.



Chapter 5 Conclusions. The shared layer perceptron topology

- is very robust. It performs regardless of asset or time span.
- adds economic value. It beats the benchmarks consistently.
- is versatile. It works well on a wide variety of financial applications.
- is easily parallelizable. It can be trained on off-the-shelf hardware.

Advanced neural networks provide sustainable and economic competitive edge in today's financial markets.

Figure 1.1: Steps towards advanced neural networks for today's financial markets.

Chapter 4 presents different applications of the shared layer perceptron. Common to these applications is that the shared layer perceptron offers a *unique* way to model them *elegantly* and *naturally*. This is not to say that these examples are not amenable to, e.g., a multi layer perceptron approach. But one would have to build 500 models instead of just one. The examples are, of course, from quantitative finance. First the dataset is introduced. It consists of 25 time series from different asset classes: equity indices, interest rates, currencies, and commodities. The salient features of this data are extracted with classical statistics. In the following three applications are presented. The first models market value at risk. The second supports transaction decisions. I.e., it helps to find the best entry point in a given time frame. The last application is the obligatory trading simulation with additional filters.

Finally, the last chapter wraps up the book. Beside a summary the reader will also find management recommendations how to put the shared layer perceptron to good use. There is also an extensive critical section. Indeed, in this book the author is only able to scratch the surface. With creativity the potential of the shared layer perceptron is vast. In this section the author details what he could achieve and what remains further research. The author also explicitly states application limits.

1.2 Motivation: Can Forecasts Work?

The main argument against successful forecasts goes along the lines of the efficient market hypothesis. Advocators of it state it in several forms: Risk free excess profits are not possible. Efficient markets cannot be beaten. However, consider the following:

- To the best of the author's knowledge nobody has asserted that it is not possible to achieve excess profits with superior models, investment in technology and manpower. Developing a model, possibly buying and maintaining the necessary hardware to run it on, staffing an investment office — all these are upfront investments, or risks, that one would not take if they were not rewarded.

- There is strong reason to suspect that some markets *cannot* even be efficient. E.g., the bond market is utilized by governments to steer monetary policy. Market operations of central banks are not necessarily directed at achieving

trading profits. Some actions, like providing liquidity to other participants by buying back bonds, may also occur at the wrong moment.

- When saying that it is not possible to beat efficient markets this often means a weak form: it is not possible to beat efficient markets *consistently*. And here, there may be a compromise. The author does not state that a single model will work well *ad infinitum*. He asserts that the presented *model class* has all it needs to be successful over time.

1.3 Literature Review

At this point only a general overview of literature which links the fields of this book is given: i.e., grid computing, neural network topologies and neural network applications in finance. As this specific combination is quite unique there is understandably only a small amount of available literature. In any case the reader *should* read [43] which links all of the above fields. As a shameless plug the author also recommends his papers [47, 327]. [63] at least analyzes distributing financial applications on heterogeneous clusters. The approach in [67] is usable for FAUN when viewed as agent.

A detailed introduction to available literature is given at the beginning of each chapter.

And now, the author sincerely hope you'll enjoy reading this book.

2 Engineering and Reengineering of FAUN Neurosimulator

2.1 Introduction

A significant part of the work in this book flows in various enhancements of the FAUN Neurosimulator, FAUN = Fast Approximation with Universal Neural Networks. In this chapter all aspects concerning software engineering are discussed. The conceptual layout of the components of FAUN is shown in figure 2.1 on page 30. The reader may want to consult this figure when following the architectural description of FAUN. This chapter however does *not* cover mathematical details of the newly implemented topology, the shared layer perceptron. See the following chapter for that. *Neither* does this chapter discuss specific applications of the shared layer perceptron. These are again saved for a later chapter.

With this said the author first looks at historical development of FAUN:

- The very first version of FAUN 0.1 is developed between December 1996 and February 1998 by Michael H. Breitner. It features three layer perceptrons which are optimized using NPSOL. Matrix algorithms are not implemented yet.
- Michael H. Breitner develops FAUN 0.2 between March 1998 and January 1999. Key enhancements are the implementation of matrix algorithms from the Basic Linear Algebra Subroutines, BLAS. This allows performance tuning by using vendor libraries. FAUN 0.2 also considerably facilitates usage by offering online and offline graphics with the free software package Gnuplot.
- FAUN 0.3 is developed by the FAUN project group since the beginning of 1999. Key developers are also Marc Ambrosius, Ulrich Kritzner, Patrick Mehmert, Lars Neujahr and Janka Zündel. Additionally a PVM version is developed to allow coarse grained parallelization.

- Development of FAUN 1.0 occurs until the end of 2004. Benefits are twofold: for local usage an enhanced graphical user interface is developed with the support of Simon König, Roland Kossow and also Frank Köller. An improved coarse grained PVM version is created by the author until August 2003.

- The FAUN 1.1 family, still under active development, has a web interface. This allows the program to be used remotely on the institute compute cluster. It relieves the user from having to perform the computation on his own PC. Compared to a simple command line interface this is more user-friendly. The web interface is mainly developed by Simon König. He also creates an MPI version. Parallelly the FAUN grid computing client is developed by me. The focus of the present chapter is among others on the grid computing client. An additional measure, the curvature tensor, is implemented by Frank Köller and Simon König in Frank Köller's dissertation.

The present work extends FAUN on different conceptual levels:

- A new neural network topology is implemented, the shared layer perceptron which is a recurrent network. The goal is to facilitate time series applications with FAUN.

- A grid computing client for FAUN is developed, which is designed to avoid the disadvantages of PVM and MPI. A requirement is, especially, that it is platform agnostic. I.e., it works on Windows, Linux and other platforms. It also offers power management functionality. Compute clients are only started when needed and powered off after the computation.

- As an additional feature the option to use fine grained parallelization is also implemented.

Integration of the shared layer perceptron is a well structured task. Since the reengineering by Simon König FAUN is divided into several well defined and documented modules. It is therefore conceptually straightforward to add another topology. However, handling of input and output data has to be slightly altered. *Implementation* of the shared layer perceptron is a well structured task, as well. But as a new algorithm has to be developed possible hurdles are to be taken into account. The development of the grid computing client is, however, badly structured: concrete requirements will have to be gathered during development itself. — The simple

requirement «It should work!» is not enough. Problems are to be expected when dealing with different platforms. Especially automatic startup mechanisms are different, e.g., between Linux and Windows. For this reason the author decided to divide the entire implementation work into several manageable modules. Completion of these modules are considered as milestones:

- Implement the shared layer topology and successfully train exemplary networks. The realization of this task is considered straightforward. However, it is not clear a priori if all mathematical and technical problems can be overcome.
- Integrate the shared layer perceptron into the FAUN command line version. The new topology is accessible via the file based interface. Due to the module structure of FAUN this task is straightforward.
- Integrate the shared layer perceptron into the web based interface. The new topology is accessible via a dedicated sub page. This task might lead to technical complications because the web interface is not a priori designed with extendability in mind.
- Implement a cross platform client-server pair which distributes control and data files and collects the results. This program is accessible via the command line. It acts as wrapper for FAUN and can be used as drop in replacement. I.e., the caller is not aware that computation occurs non locally. This task is difficult. It is not clear a priori which technical problems might occur.
- Implement wake up and shutdown functionality. Wake up is realized via a separate program using wake on lan. Shutdown is integrated into the grid client. This task is technically straightforward. However, difficulties are expected from the fact that test computers are on different routed and firewalled networks. Difficulties may also arise because the targeted student and staff cluster uses desktop computers. They may not offer the reliability and versatility of server hardware.
- Develop an installer that distributes the client and automatically schedules it to run at startup. The user of the installer only necessitates minimal information on the target platform. This task is not difficult but tedious. A significant amount of testing is expected for getting this to work on all available platforms.

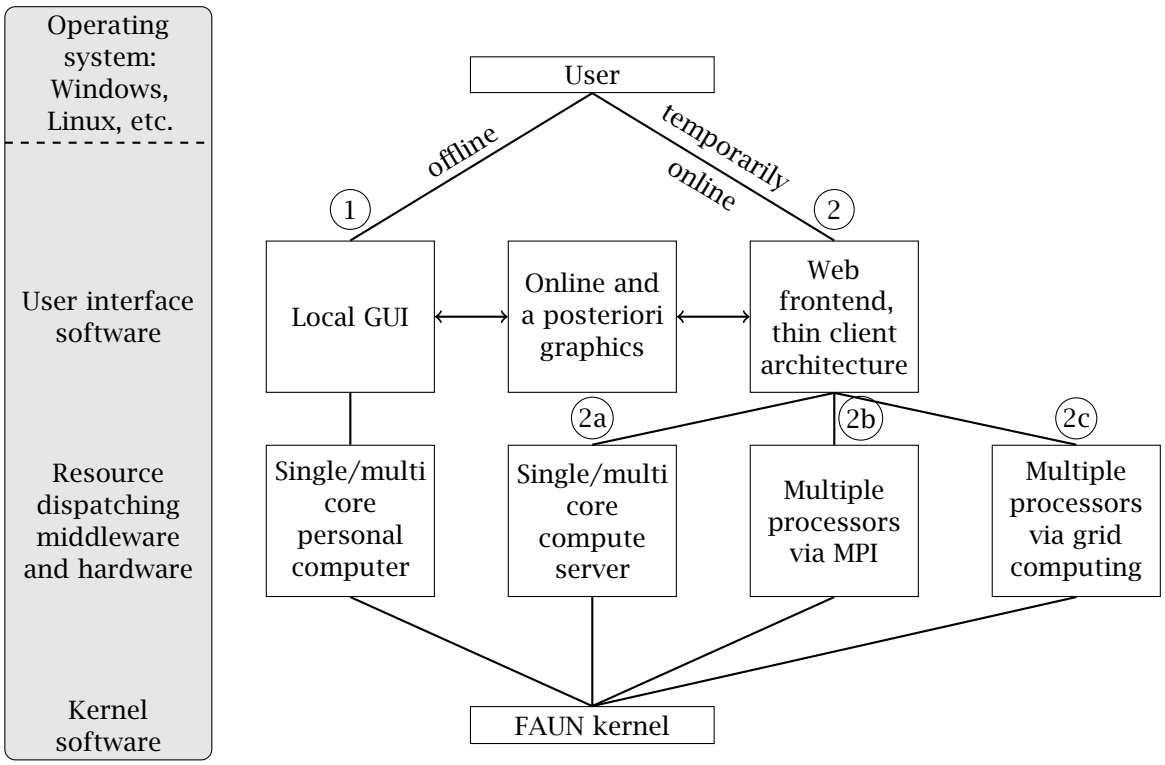


Figure 2.1: 3-layer architecture of the FAUN software suite. Users choose between local installation (1) or web frontend to access FAUN (2). The middleware distributes tasks user-definable to the FAUN compute kernel on one (2a) or many processors (2b and 2c). Applications of every layer are independently replaceable and available for Windows and Linux.

- Make the wake up and shutdown functionality available via the web interface. This task is straightforward within the above limitations.

- Implement fine grained parallelization using the OpenMP standard. This task is technically straightforward. However, performance improvements are not guaranteed a priori.

Algorithmic details of the shared layer perceptron implementation are left for the following chapter. The author discusses fine grained parallelization and the grid computing client in the following. This also includes a detailed analysis of achieved speedup. Extensive documentation of new FAUN functionality is presented. This includes user manual, administrator and developer documentation. Numerous applications of FAUN are discussed in the following section.

2.2 Literature Review

The present section focuses on references with a *technical* background, i.e., programming process and languages, and grid computing.

Programming Process

[33] deals with agile programming practice and how best to implement it. Although in the present case the author is the only member of the «development team» [74] proved useful, especially in the end phase of development. It merges agile and non agile practices. Similar approaches are discussed in [179]. [180] deals with the challenge of correctly estimating development times. The advantage of this experience report is that it is actually usable. Implementation is considered in [192,277,292,293,304]. [341] focuses on implementing agile and extreme practices in a scientific research context. [344] merges agile practices with grid computing. Arising problems are discussed in [211,335]. [228] provides guidance how the agile process should be adapted relating to complexity and uncertainty. [234] goes in a similar direction. A classical, readable and also entertaining debate of agile — and extreme — methods is found in [84]. The article is recommended.

Language

- FORTRAN: The author found [60] to deliver a very pragmatic and usable approach to the language. [248] is another very good reference which goes into more details. Not specifically FORTRAN related but used in this context is [157] which deals with compilation issues concerning OpenMP. A classic MPI reference that the author used is [165]. Also related to MPI [261] is a bit dated but conveys the basic concepts very clearly. [201] specifically focuses on using BLAS libraries with FAUN. [325] deals with the PVM implementation of FAUN.

- Ruby: The classic and very entertaining to read reference to Ruby is [305]. By reading this book one will get interesting insights into the Ruby programming language. [236] provides use cases for Ruby. These are not particularly FAUN specific but nevertheless present the language well. [163] provides several problem solving strategies the author has found useful. Testing is an integral

part of the development process. A good advanced reference — not Ruby specific — is [232].

Grid Computing

- Introduction: Everybody at all interested in grid computing should read the compact description [130] by Ian Foster, the «father of grid computing». [133], also by Foster and his colleagues, goes into more details. The basic concepts of service orientation are also well presented in [131, 132].

- Resource allocation: [48] provides a concise introduction on how to get the best out of a cluster. This paper was of considerable help when planning job utilization. [4, 221, 358] deal with the task of how best to select resources when scheduling jobs. [39] highlights the aspect of cooperation among processes. [81, 222] focus on the *on demand* aspect of grid computing which is implemented in the FAUN grid computing client *par excellence*. [135] proposes the concept of a *grid of grids*. This concept is of particular interest with respect to FAUN when different subnetworks are used. The author plans to dedicate further research to implementing just that: a whole cluster offers itself as a highly performing resource.

- Middleware and applications: [5] presents the grid application toolkit, a middleware for resource dispatching. Similarly, [134] introduces the globus toolkit. [24, 168, 265, 342] deal with specific requirements of quality of service. Both references helped in designing the FAUN grid computing client which is a middleware. [53] specifically addresses the problem of managing the workflow in a grid application. Similar aspects are dealt with in [158]. [125] proposes a method to describe grid workflows with a domain specific language. Although the author didn't use this for FAUN it helped in outlining important parts in the process of distributing and collecting data. [238] focus on grid computing in the context of computational finance. [120] introduces the publish-subscribe mechanism. This concept is implemented in simplified form by the FAUN grid computing client. Every client can also be considered as a service. This is described in [191, 264, 310, 321]. [240] is an example of the growing importance of grid computing projects. Viewing clients as independent agents is a sensible paradigm in grid computing. [67] provides a good introduction

to the topic, see also [167]. [51] carries the term grid even further and envisions a structure similar to a power grid. A similar position is found in [55]. Visionary is also [83] which proposes techniques to enhance existing grids.

2.3 Fine Grained Parallelization

Since the advent of cheap consumer grade multicore processors compiler vendors have implemented features to semi-automatically distribute tasks over different cores in a computer. Especially loops and array operations are candidates for fine grained parallelization. However, one may not hope for the same reduction in computation time as when using coarse grained parallelization. The reasons are that the distribution of tasks among threads involves significant management overhead. E.g., for a matrix multiplication the different parts of the matrix have first to be dispatched to every thread. If this involves a thread on another processor the comparatively slow inter processor connections are used instead of the processor caches. Then the actual multiplication occurs in parallel. Finally, the result data is collected and stored.

These three steps also occur *in principal* when distributing entire networks, i.e., coarse grained parallelization. The main difference is that with fine grained parallelization the typical duration of a task is in the sub millisecond or millisecond region. And the shorter the task the more important are the effects of managerial overhead. If one wants to avoid this one should parallelize the *outer* loops of a program, if possible. However, in the case of neural networks one conceptually only has one outer loop. This is the training of *several* networks. *Within* the training options are limited:

- The *forward* pass involves matrix algebra. However, with the shared layer perceptron each time step depends on the previous time step. One cannot parallelize the entire loop. But one can parallelize the matrix operations.
- Calculation of errors is parallelizable. The computational effort is however negligible: one subtracts two numbers and squares them. One does not expect a high performance gain from parallelizing these operations.
- The backward pass again has a sequential dependency. One cannot parallelize the entire loop. But the computationally intensive matrix operations can be parallelized.

- The line search of optimizer NPSOL is parallelizable. However, time spent in NPSOL is by experience generally less than five percent of total computation time. One does not expect a significant performance gain.

- Update of the weight matrix. This trivial task is fully parallelizable.

Figure 2.2 on page 37 shows the reduction in computation time when using several threads for the shared layer perceptron. With a $D \times D$ matrix a dense matrix multiplication accounts for D floating point multiplications and $D-1$ additions for every element. As there are D^2 elements in the matrix a full multiplication needs $O(D^3)$ floating point operations. Here, one does not differentiate between multiplication and addition as these operations are realized with the same computational effort. Please note that the matrix multiplication is especially suited for the FMADD operation of modern processors. This operation allows a floating point multiplication *and* addition to be carried out in a single operation. In matrix multiplication one encounters daisy chains of FMADD operations.

Figure 2.2 and table 2.1 on page 38 confirm that fine grained parallelization indeed improves computation times — as long as operations are confined to a *single* processor. The numbers are created using a dual quad core workstation. This computer has two distinct quad core processors. Connections between these two processors are comparatively slow when considering the processor caches. This is what one observes: the improvement when going from 1 to 4 threads is quite remarkable. All threads execute on the same processor. However the improvement of going from 4 to 8 threads is not so impressive. The performance penalty is incurred because additional communication is necessary between two processors.

We also note that in two cases adding another thread produces worse results. This happens when the scheduler distributes tasks in a suboptimal manner. Once a task is distributed the entire program *has* to wait for the results to arrive. The slowest task finally forms a bottleneck. Let the author illustrate this with an example. One wants to multiply two matrices with dimension 100×100. Using the above analysis this requires

$$1,000,000 \text{ multiplications and } 990,000 \text{ additions}$$

for a total of

$$1,990,000 \text{ floating point operations.}$$

Now assume that one wants to distribute this task evenly on 4 threads. Slicing the first matrix after each 25th row this is easily possible. Each thread has to compute

497500 floating point operations.

A core is clocked at 2.5 GHz or higher. This results in theoretical peak performance of $4 \times 2.5 = 10$ GFlops per second. Therefore the operation of every thread only takes 49.75μs. This is a very short time span even for modern processors and operating systems. Time slices of 10ms are common, i.e., 200 times more than the duration of a single task. One sees how time sensitive fine grained operations are.

The figure also shows that adding more threads than physical cores does not improve the results significantly. This is unsurprising as the processor does not use hyperthreading. This technique provides two or more *virtual* cores for every physical core. A small improvement is noticeable, though. It is due to the fact that even on an almost idle computer there is a competition for resources among different processes. Using additional threads induces the scheduler to allocate more resources. As these additional threads are scheduled on a core where data is already copied they do not incur that much of a performance penalty.

Figure 2.3 on page 38 shows the effect of different state space dimensions. One sees that the reduction in computation times is most noticeable for the greater dimensions. The smaller dimensions do not profit that much. This is due to time sensitivity in the sub ms domain as mentioned above.

Figure 2.4 on page 39 presents the dependency of computation time on the number of observeables in the shared layer perceptron. One sees that a performance penalty does not happen for small numbers of observeables. On the other hand one can also argue that increasing the number of observeables does not improve fine grained parallelization computation times.

Compared to coarse grained parallelization one notes that fine grained parallelization does not perform as well. While coarse grained parallelization yields speedups of more than 95 percent, fine grained parallelization only reaches 58 percent for 2 threads and 33 percent for 4 threads, see table 2.1. The reader may rightly ask *why* the author concerns himself at all with fine grained parallelization. This question is justified even more by the fact that coarse grained parallelization of entire network training seems very natural and intuitive. Fine grained parallelization is an important additional performance tool:

- The performance speedup is realized without any additional infrastructure. There is no need to install software like MPI or PVM. Special configuration on the host computer is not required. The developer can entirely *hide* the parallelism with respect to the user. The only thing the user will notice is that training is faster. This is especially interesting for people using the local version of FAUN. These versions will probably run on dual or quad core consumer processors, see below.

- The first networks arrive faster. This is especially interesting for real time applications when networks are retrained on a continuous basis. Consider, that with 8 cores training time is only 66 percent of that compared to a single core. While this is not satisfactory in terms of speedup it is still a significant reduction of computation time in its own right. Time sensitive applications can profit from this.

- Related to the previous point fine grained parallelization is especially interesting on consumer dual and quad core processors. These processors are available with clocks up to 3.4 GHz. E.g., the Intel i7 processors provide native dynamic overclocking. When, e.g., 2 cores are idle the other cores are overclocked. Fine grained parallelization can take advantage of this and provide the first results faster.

- Fine grained parallelization is obligatory when using general purpose graphical processing units. Computation on the graphics card involves several hundred threads executed with comparatively low clocks of 1 GHz or slightly more. This kind of architecture is well equipped for handling matrix algorithms which are predominant in neural network training.

Especially the last point looks very promising. Although computation on the graphics card is still in its infancy vendors begin to discover the potential for high performance computing. Nvidia offers rack mounted kits of four high memory graphic cards and proposes the CUDA architecture. ATI/AMD offers the stream technology. And both vendors work on the OpenCL standard designed to provide unified access regardless of underlying vendor chips. First inquiries have shown that for large networks the FAUN kernel is able to scale well to several hundred threads. However the results are still preliminary and are not reported here. But this is an interesting research area the author plans to devote more resources on.

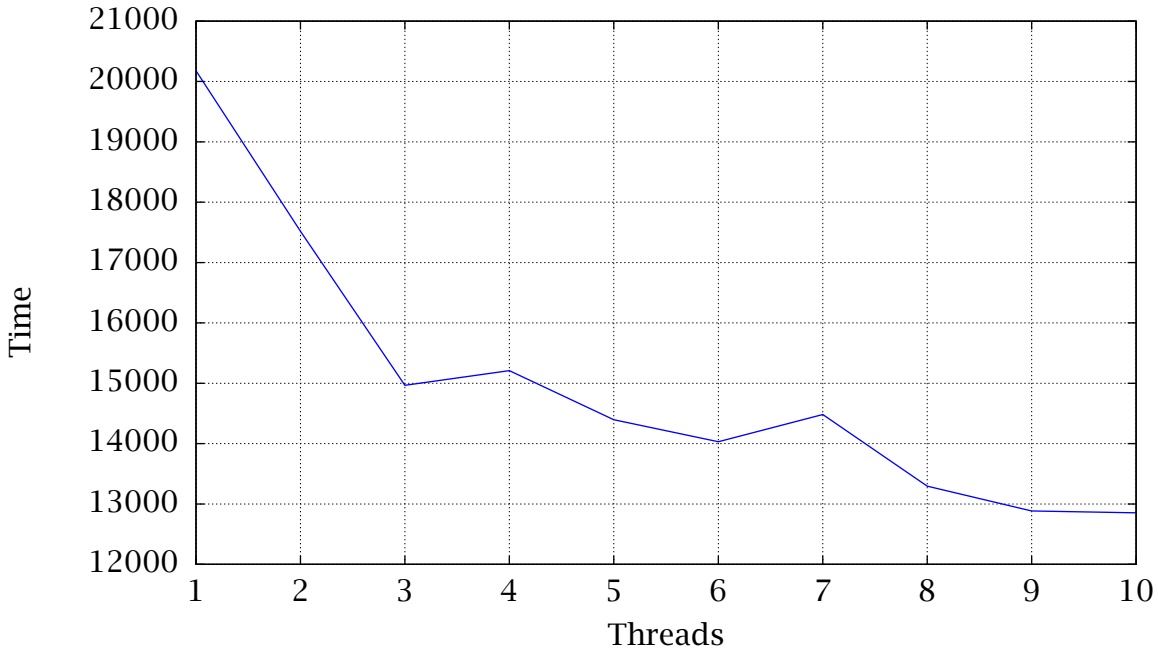


Figure 2.2: Effect of fine grained parallelization for state vector with $d = 500$, computation time in ms. The higher the dimension of the state vector the more important the speedup with fine grained parallelization, because the involved matrix and vector multiplications are easily parallelized. This example is computed on a computer with two quad-core processors. This effect is noticeable: the improvement in going from 4 to 8 threads is much less important than going from 1 to 4 threads. The resulting inter processor communication is responsible for the relatively low gain in speedup. Also, the required additional resources for managing and feeding a new thread sometimes even *increase* computation times. Finally, one notes that using more threads than physical cores unsurprisingly leads to almost no improvement. The small resulting improvement is due to a greater part of computational resources being allocated.

Threads	Time in ms	Speedup	comment
1	20178	1.00	only one processor
2	17519	0.58	
3	14966	0.45	
4	15210	0.33	
5	14395	0.28	second processor used
6	14031	0.24	
7	14482	0.20	
8	13296	0.19	
9	12885	0.17	more threads than cores
10	12854	0.16	

Table 2.1: Computation time and speedup.

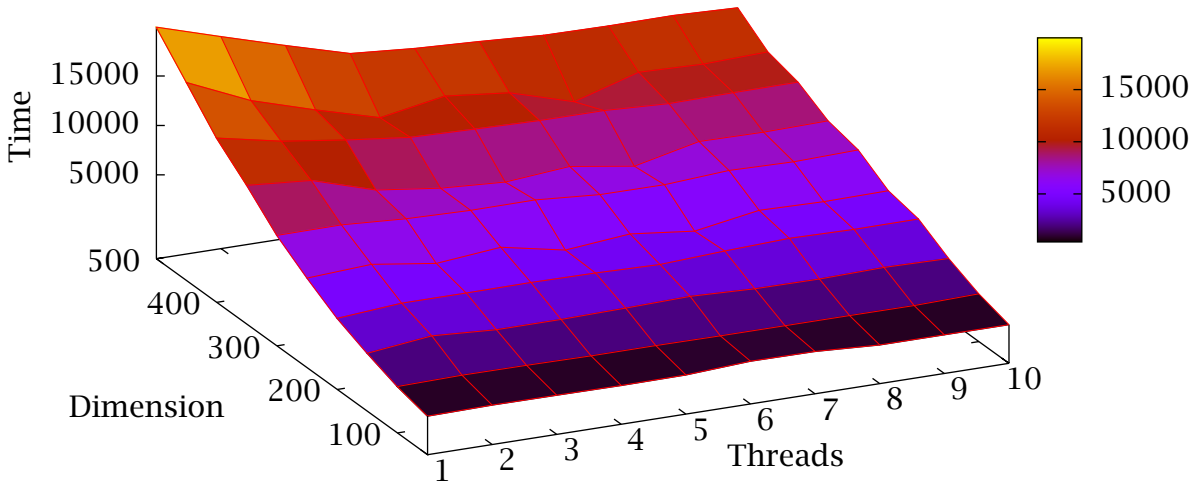


Figure 2.3: Fine grained parallelization for different number of threads and dimensions, computation time in ms. One notes that fine grained parallelization improves computation times when the dimension of the state space vector is high. This is caused by a certain overhead incurred when distributing data among cores and especially among two different *processors*. The figure suggests that one should mostly use fine grained parallelization on single processors and rather use coarse grained parallelization when inter processor communication is involved.

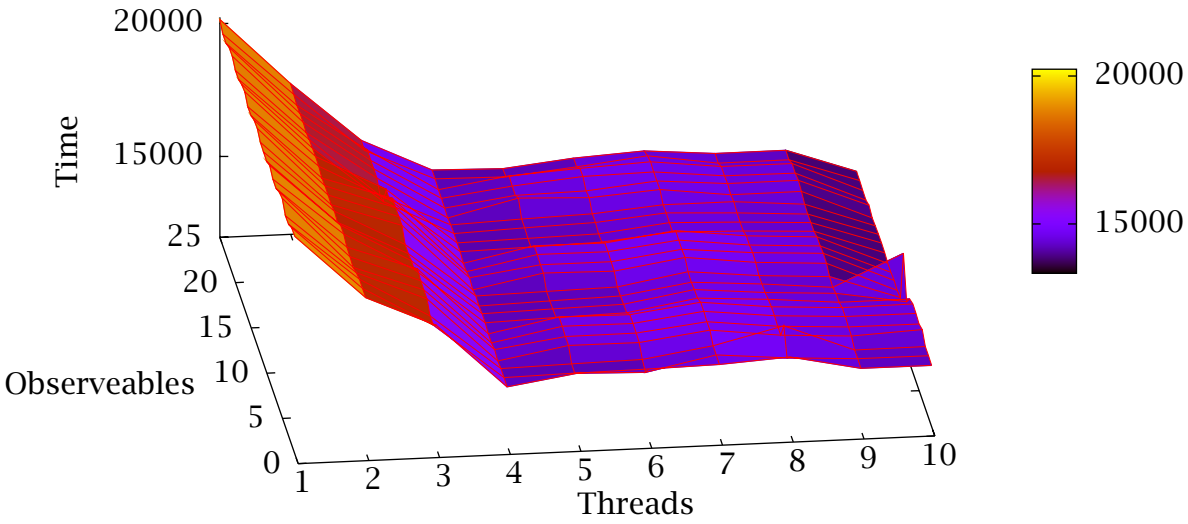


Figure 2.4: Computation time in ms for different numbers of observeables. One notes note that fine grained parallelization works equally well even for few observeables. The performance impact encountered with state space dimensions is not found here.

2.4 Coarse Grained Parallelization on Inhomogeneous Clusters: The FAUN Grid Computing Client

The present section deals with design and implementation of the FAUN grid computing client. The section does not include documentation which is saved for section 2.5.

2.4.1 Programming Language Requirements and Selection

As the grid computing client is required to run on different platforms without modifications the choice excludes languages which are only available for specific platforms. Nowadays, however, compilers and interpreters exist for a wide variety of languages and platforms and this requirement merely excludes very specialized languages, e.g., for embedded processors. It is not a restriction. The author also wants to avoid standard languages with *proprietary* extensions.

The goal is also to realize easy maintainability. On the other hand, performance is of secondary importance because the compiled FAUN kernel carries out the training. The client is intended as a wrapper and is merely responsible for task dispatching and data gathering. With this in mind the author decided to opt for an *interpreted* language in contrast to a *compiled* language. The interpreted language has the advantage that it is not necessary to produce releases for every target platform. The same program will run unmodified on all platforms. Of course, one has to take care for individual platform aspects, like, e.g., when system commands are executed. But the FAUN grid computing client is entirely platform agnostic. The requirement of an interpreted language excludes the otherwise strong candidates C/C++ and FORTRAN.

The Java language has a strong name for being platform independent. However, it is generally compiled to virtual machine code which will then be executed by a platform specific Java virtual machine. This is a dependency which the author wanted to avoid. Additionally it is also an unnecessary step and there is a priori no possibility to influence which *exact* version of Java will be installed. Download size is typically at least 10 MB.

Finally, the author decided to settle for the Ruby programming language. It offers the following strong points, especially when dealing with distributed platform independent systems:

- The single most important unique feature is that it is fully object oriented. The entire grid computing client is realized as a collection of objects which are distributed across the network.

- Ruby possesses a distributed object model which shares objects among different computers. Compared to message passing software or raw network programming results are achievable much more rapidly. The distributed object model relieves the programmer from knowing details of the underlying network protocol. It is described in detail in subsection 2.4.2.

- Ruby is a dynamic language which allows changes to methods of classes *at runtime*. This is especially important when considering that the client will have to updated — be it because of bug fixes or to enhance functionality. In any case Ruby allows to update the file and reload it at runtime.

- The Ruby interpreter is very lightweight and can be stripped down and bundled with the program to be distributed. The advantage is that not the entire

library tree gets bundled but only the necessary functionality. This allows for small interpreter files of less than 2 MB.

- The bundled interpreter does not require any installation on the host system. It runs out of the box.
- Ruby includes especially strong support for various network functionalities.
- The author has found the Ruby developer community to be responsive and cordial.
- Should the need arise Ruby *can* be compiled to Java byte code and executed with a Java virtual machine. This technology is called JRuby. Although the author explicitly does not want to use this for the present version it is a good option for potential future development.
- Last but not least, software development being a project and the author the only developer he had to keep in mind the experience of the developer team. As the author uses Ruby on a daily basis and also has realized important projects with it he deems it quite natural to choose a language which he is already experienced in. E.g., WARRANT-PRO-2 for derivative optimization and design has a wrapper graphical user interface written in Ruby.

2.4.2 The Distributed Object Model

A remarkable concept with Ruby is that every object can be *marshaled*. This means that the current instance is transformed into a byte stream and saved. It can then be loaded again for future use and start in the exact same state as it was before marshaling. Marshaling provides an easy way to distribute objects among a network. This is realized in Ruby with the distributed object model called DRb, short for distributed Ruby. This works as follows:

- The object is marshaled.
- The marshaled stream is sent over the network.
- The object is demarshaled again and available.

The advantage of this proceeding is that the calling process does not need any knowledge of the called object. Of course, it is necessary to know the appropriate

method to call. But except for these well defined interfaces the object stays opaque. Note that with the `undumped` option the calling process does not require the object source code: all calls are marshaled to the server and results are marshaled back fully transparently. This works in both directions. Every client can in principle also act as server.

In the context of the FAUN grid computing client it is important to note that one wants to keep communication requirements to a minimum. The target is to put the Windows hosts in the student computer pool to work. This ITS pool is a separate sub network which is firewalled. The server and other powerful Linux hosts are in yet another sub network. For security reasons it is not advisable to enable connections from the *outside* to the ITS pool. For this reason the clients are designed to connect to the server at startup. This is quite the opposite from the standard MPI concept. There the server uses, e.g., ssh to contact every client. Using the distributed object model the only communication requirement is that the client can reach an arbitrary port on the server.

This design choice is also intended to provide trust for additional clients. Consider the following: when one dedicates spare computing capacity to a central server one would rather like to have control over the connection than have a server connect to the computer. While such a design actually makes crashes of the client a little bit more difficult to handle it serves its purpose in the present case.

2.4.3 Wake Up and Shutdown

An important requirement in the FAUN grid computing client is to put resources to work economically and ecologically. The idea to use idle computing capacity in university clusters of workstations and individual computers is only really sensible when these computers do not have to be powered on full time. Assuming that spare capacities are not *always* needed but only on demand one requires some kind of mechanism to power the computers on and off.

Server grade hardware features an intelligent platform management interface, or IPMI, which allows to do just that. IPMI often provides much more information, like, e.g., temperature. However, IPMI is not an option in the present case. First, IPMI is not available on all computers and especially it is not available on most non business computers like those in the faculty cluster. Second, the IPMI interface itself consumes a small additional amount of energy.

The author decided that the simplest way to achieve the desired functionality is to use wake on lan, or WOL. This standard is nowadays implemented in almost every mainboard. It requires the power supply to still deliver a small amount of power to the network interface card, even when the computer is «powered off». This is also referenced as soft off state. However, the amount of power required to sustain soft off mode is small compared to leaving the computer running idle and also compared to other management solutions.

Wake on lan is realized by broadcasting a so called magic packet. This small network packet especially contains the MAC address of the network card repeated 16 times. The network card will parse every incoming packet for this sequence and wake up the computer when appropriate. Tools implementing wake on lan are readily available for every platform. They are also easy to program in almost any language able to interface with networks. Note that wake on lan is not secure. Anyone with the ability to connect to the network can send a magic packet. Although some Linux flavors will require one to be root, i.e., the administrator, to send a packet to the broadcast address this is by no means obligatory. Generally routers are designed not to forward broadcast packets across network boundaries. The potential abuse is therefore limited to the local network and people knowing the MAC address of the network card. Nevertheless let the author emphasize that the author does not recommend to enable wake on lan on computers with sensitive data on it. Although not a threat in itself the computer may yield access to badly secured data.

Wake on lan necessitates a computer in the same sub network as the hosts to wake up. In the case of the student and staff cluster this is not a problem because there is already one server running full time. If no such server is available one might consider adding a low cost low power computer as wake up server. This server then connects to the FAUN grid computing server and dispatches magic packets when needed.

Shutdown is realized by operating system commands. I.e., the client retrieves the order to shut down from the server and acts accordingly.

Note that when using a network switchable power outlet it is possible to *completely* eliminate power consumption of the computers which are otherwise in soft off mode. When several computers are switched via a single power outlet the power consumption of the outlet is negligible. While this solution might be the preferred one from an ecological point of view it is not practicable in the present case. Realiz-

Configuration	single system in W/h	entire cluster in W/h	cost per hour in Euro cent
Standby	0.4	24.0	0.51
Idle	44.7	2682.0	56.46
1 core	61.7	3702.0	77.93
2 cores	74.9	4494.0	94.60

Table 2.2: Power consumption for different scenarios in the student and staff cluster. Hourly costs are based on a rate of 21.05 Euro cent per kWh. The 1 core configuration signifies that only 1 core is loaded to full capacity, the 2 cores configuration loads both cores. This is what the FAUN grid computing client does.

ing the switching requires a significant amount of additional cabling for the student and staff cluster. Another reason is that switchable power supplies often provide non standard interfaces and not everybody uses them.

2.4.4 Cost Analysis

The student and staff cluster comprises 60 dual core E8400 systems at 3.0 GHz. Table 2.2 shows the results of the measurements. One notes that there is an important difference between the standby and idle numbers: powered on the computers consume 112 times more electricity than in standby mode.

We want to get a realistic impression of what one can save when powering off the computers. For this, one assumes that the student and staff cluster is used 40 hours per week for the *sole* purpose of computation. During the rest of the time it either runs idle or is powered off in standby mode. This is not strictly true because the cluster is also used for, e.g., teaching during the week. But it allows a conservative comparison. One also have to take into account that the cluster is air conditioned. Every Watt consumed by the cluster is finally transformed into heat and has to be cooled. A typical air conditioner produces 3 units of cooling power for 1 unit of electrical energy. The total power consumption of the cluster is therefore increased by one third. Powering off the cluster after computation leads to the following figures for a 168 hours week:

- 40 hours of full capacity on 2 cores. Costs 40h $\times$ 94.60 c/h $\times \frac{4}{3}$ = EUR 50.45

- 128 hours in standby. Costs 128h $\times$ 0.51 c/h $\times \frac{4}{3}$ = EUR 0.87

On the other hand, leaving the cluster idle but not powered off gives:

- 40 hours of full capacity on 2 cores. Costs 40h $\times$ 94.60 c/h $\times \frac{4}{3}$ = EUR 50.45
- 128 hours idle. Costs 128h $\times$ 56.46 c/h $\times \frac{4}{3}$ = EUR 96.36

Aggregating the numbers into a year with 52 weeks one gets power costs of EUR 2668.64 per year when switching off. However, not switching off leads to costs of EUR 7634.12. This leads to cost savings of EUR 4965.48 per year when using the FAUN grid computing client.

We can also view power consumption from another angle and ask: What are the added costs of using the student and staff cluster for computation? In this scenario one assumes that computation is carried out during the operating time of the cluster. I.e., the added costs are the difference between full load and idle consumption. Even when users do word processing or web research the load is very low. This yields 40h $\times$ (94.60 c/h $-$ 56.46 c/h) $\times \frac{4}{3}$ = EUR 20.34. For an entire year this sums up to EUR 1057.68. I.e., one gets the approximate performance of a dedicated cluster or high performance computer for the power costs *only*. There is no need to purchase additional hardware. Of course, some fine grained problems are simply not suited for workstation clusters. But the FAUN grid computing client speeds up almost linearly, see figure 2.9 on page 52.

2.4.5 Performance Analysis

In the following the author looks at how the grid computing client actually performs when training batches of networks. The author analyzes several scenarios:

- A purely local scenario where one only considers performance within a dual quad core computer. This scenario should give us an idea of how the grid computing client scales under *ideal* conditions.
- A one network scenario where one stays within the institute's sub network. The computers involved are connected via a common 100 MBit/s Ethernet switch. However, this switch serves the entire floor and is not exclusively dedicated to compute traffic. One expects to see some slowdown because of the comparatively narrow bandwidth provided by the switch.

- A multi network scenario for which the grid computing client is actually intended. The author will connect the user workstations from the students and staff computer cluster in the second floor to the grid server in the fourth floor and compute on all available clients. The second and fourth floor are connected using a shared 1 GBit/s fiber. I.e., the theoretical peak bandwidth to the institute network is 1 GBit/s divided by the number of computers involved. Of course, the theoretical peak bandwidth to the *server* is still lower because this computer is only connected with 100 MBit/s as mentioned above.

We can only sensibly judge the speedup in the first scenario. In the other scenarios the added computers are too different as to allow a comparison. Indeed clock frequency is only a very rough indicator of what one can theoretically expect as peak performance. For actual computations, however, other factors like processor caches and data size, in-out bandwidth, and network bandwidth to name only a few become very relevant.

The throughput in networks per second for the first scenario is shown on figure 2.5 on the next page. One sees that throughput increases almost linearly up to 8 threads. This is to be expected because the server has 8 physical cores. The scheduler will first distribute idle cores to the FAUN processes. It is very interesting to see that adding *more* threads than physical cores still increases overall throughput noticeably. As the cores are already heavily loaded the increase is not linear and one notices that going from 10 to 11 threads has only a negligible effect. This is to be contrasted to fine grained parallelization where adding more threads generally *decreases* throughput. The reason that this works well for coarse grained parallelization is that FAUN also has to deal with input and output. During this time speed is limited not by clock frequency but by memory bandwidth. There is still idle capacity which the additional threads are able to exploit. This effect is very satisfactory. It indicates that the FAUN grid computing client adds value compared to using the single threaded version. Still one caveat applies: while adding more threads than cores increases the overall throughput the *very first* networks will take slightly longer to arrive because the server is loaded to more than full capacity with 11 threads.

Figure 2.6 on page 48 shows the scaling behavior in *relative* terms, i.e., the speedup. One sees that at first the speedup decreases. Then, when the second processor is hit the speedup temporarily increase *above* 1. This surprising behavior is due to the second processor taking over administrative tasks from the first and

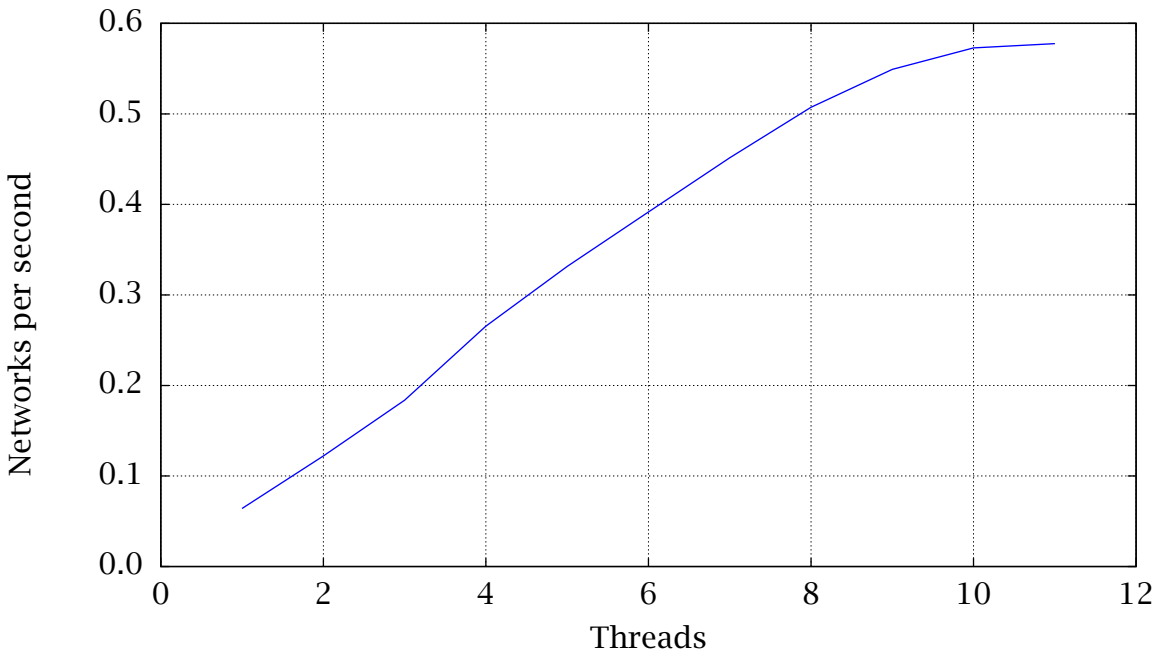


Figure 2.5: Computed neural networks per second using the FAUN grid computing client and only *one* server. One sees that using more than 8 threads still slightly improves the throughput although only 8 physical processing cores are available. This effect is due to in-out operations which do not load the core to full capacity. Therefore some capacity is still available to be exploited.

therefore relieving the overall system. While this effect is negative in fine grained parallelization because of latency effects the coarse grained version does not require sub ms latency. Then the speedup slightly reduces as expected until more threads than physical cores are added. Note that increasing the number of threads does obviously not increase the available hardware processing power. However, the threads are able to take advantage of some spare capacity. They increase the overall speedup compared to using only a single thread which is the benchmark.

Figure 2.7 on page 50 shows network throughput for the second scenario which uses diverse computers in the institute's network. Interestingly adding a second 8 core server improves results significantly, threads 12–22. Although this server is older and slower throughput almost doubles. From thread 23 onwards various dual and single core computers are added. Here, not every new thread improves the throughput. On heavily loaded computers the client is too slow to respond and

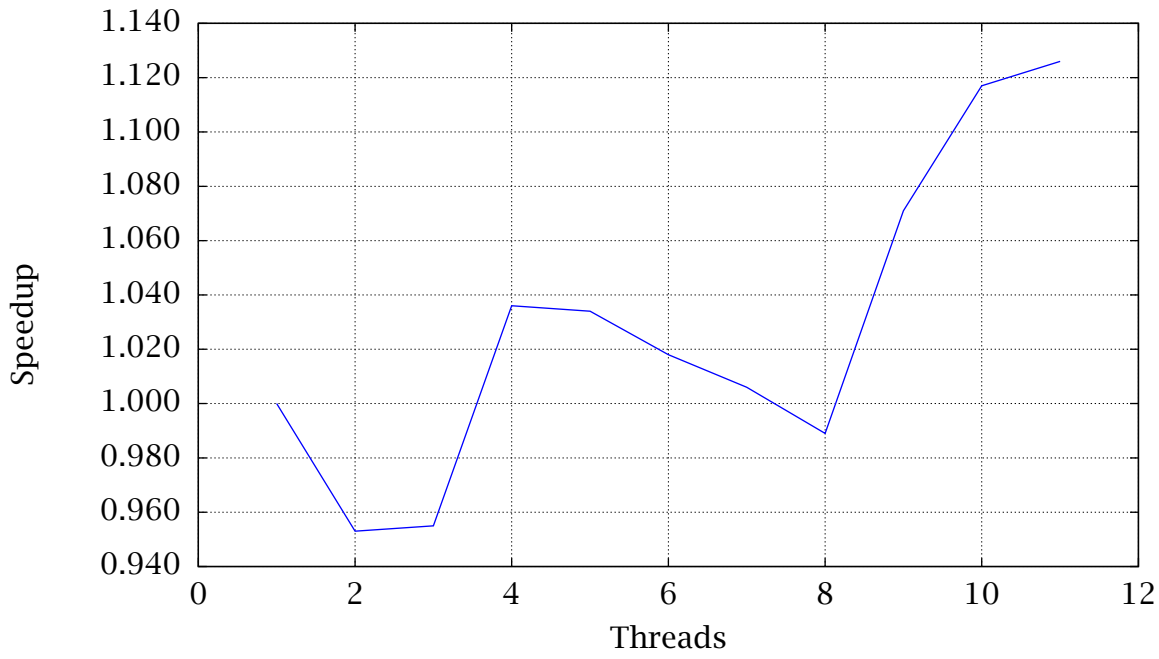


Figure 2.6: Relative speedup on a single dual quad core server. One notes that speedup is at times greater than 1. This is explained by the fact that threads are computed on two different processors. When threads start to hit the second processor the first processor is relieved from certain administrative tasks. This can be seen starting with 4 threads. The speedup declines below one until the 8th thread. Then one adds more threads then physical cores. While these threads do not have access to capacity of an entire core they still add to the bottom line of computed networks per second. Therefore they increase the speedup compared to running only a single instance of FAUN.

actually *slows down* the entire calculation. On shared computers it is necessary to set the FAUN grid computing client to low priority because otherwise services are disturbed. The threads include normal user computers, smaller and older servers hosting virtual machines and file servers on Windows and Linux. Simply put: everything one finds in an organically grown network. Not using the grid computing client with lowest priority quickly makes one very unpopular with other users. Figure 2.7 on page 50 shows a very realistic scenario where not every threads adds value. However, it is not possible to know *a priori* which computer will be loaded and which not. It is best to include all available hardware in the computation be-

cause even small improvements are improvements that one gets at no additional hardware cost. With very slow computers it can happen that the client never manages to send the results back before all others have finished. In this case a small performance penalty is incurred.

Here, the question of speedup is more difficult to answer. How can one compare a cluster of inhomogeneous computers which are as varied as can be. Operating systems Linux and Windows, professional and consumer hardware, very recent hardware from 2009 and hardware as old as 2004, different vendors, etc. the author chooses to benchmark the involved computers individually with the single threaded version of FAUN. Then, it is possible to compare how much of this possible performance is realized in the grid version. Still, the results can only be indicative because the individual load situation on every computer will vary between benchmark and production runs. With all these caveats figure 2.8 on page 51 shows the estimated speedup for the inhomogeneous cluster. The effect of a second slower 8 core server up to thread 22 is clear to see. Like in the first scenario speedup still increases when adding more threads than physical cores. In the case of this shared server the effect is due to more threads also getting a bigger share in total system resources. Even when setting the threads to low priority one should be cautious in doing this. Other applications will be less responsive. From thread 23–36 other computers are added. One sees that this — with few exceptions — decreases the speedup. The overall speedup is always above 95 percent. This number is actually very satisfactory. It indicates that only 5 percent of theoretical performance are lost in communication and management overhead.

Figure 2.9 on page 52 shows estimated speedup for the third scenario. Here, the author includes 21 dual core computers from the student and staff cluster. Because the computer room is still in the building process the author was not able to use all systems. But the figure already indicates that one can expect very good speedup. One notes that speedup is jittery. This is due to using a shared network. One can expect that, e.g., downloads of other users, IP-telephony and other network services interfere with data transmission.

2.5 Extended FAUN Documentation

[208] is the authoritative user manual for all aspects of FAUN. The present work extends FAUN functionality in several ways. The author provides compact additional

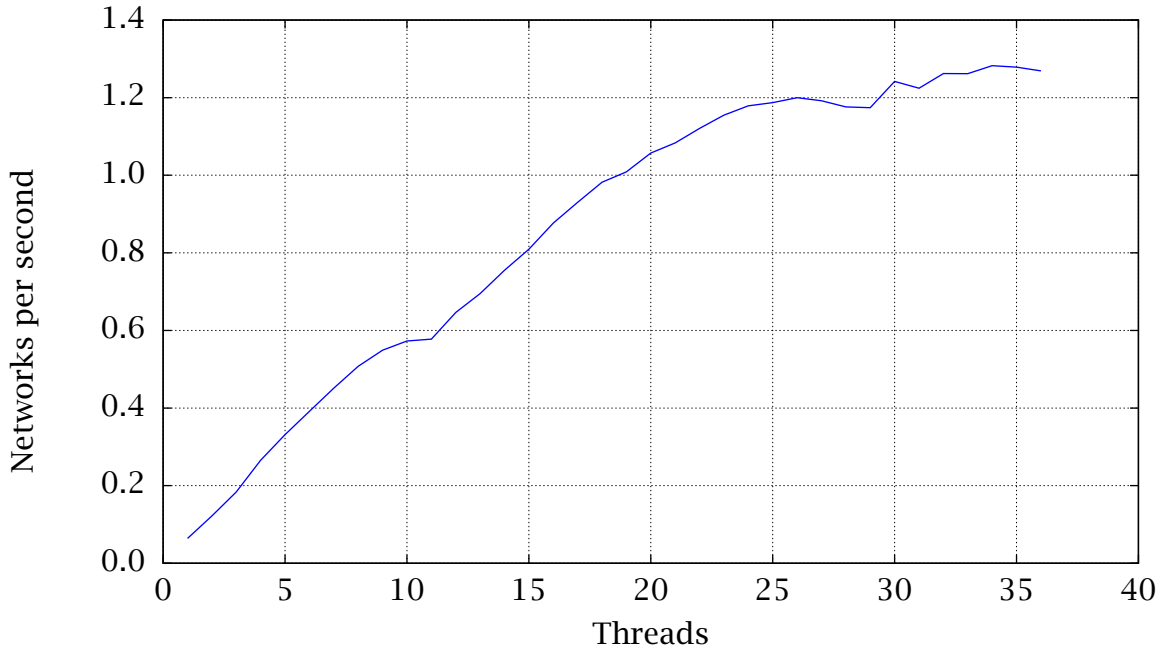


Figure 2.7: The second scenario uses different types of computers within the institute's network. The first 11 threads run on a recent 8 core server exclusively dedicated to computation. The following threads are set to low priority to avoid disturbance of user users. The threads 12–22 run on an older 8 core server which also is *professional* hardware. Threads on this server scale similarly well. The slope of the curve is reduced due to significantly slower clock speed. This server is shared with other applications. The next threads 23–36 successively use diverse single or dual core computers which can be heavily loaded otherwise: 2 of the computers run Windows 7, the others run various flavors of Linux and host virtual machines for other users. Threads 27–29 show that adding more threads does not necessarily increase throughput. This happens on systems with otherwise heavy load.

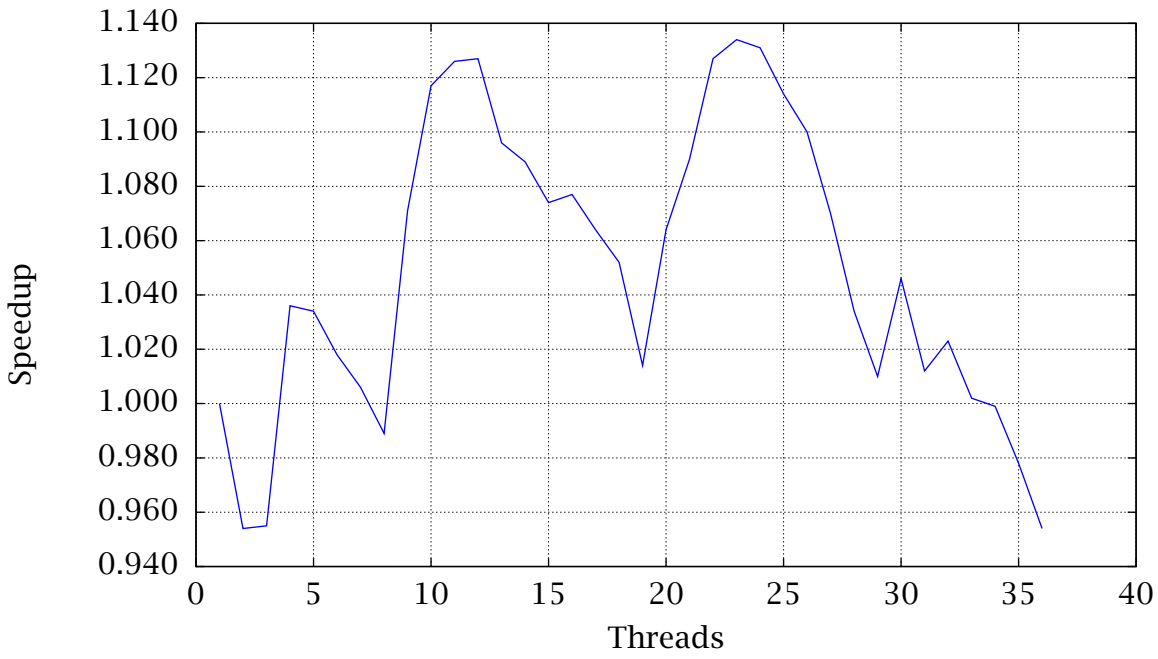


Figure 2.8: Estimated speedup for cluster of heterogeneous computers. This speedup is estimated using single threaded benchmarks on the individual computers. It is only *indicative* because benchmark and test run results depend on the actual load on the computer. One sees that speedup is stable until thread 22. In the following various loaded computers are added which — not surprisingly — worsen the speedup. It is remarkable that the speedup *always* stays above 95 percent. This is a sign of very good resource utilization. However, slow computers have a disproportional negative effect. On the one hand they only deliver few results. On the other hand they are slow in actually sending them back.

documentation for the new functionality. This includes

- How to access the shared layer perceptron topology.
- How end users can compute using additional resources from FAUNgrid.
- How the administrator can configure the grid client.
- How to extend the shared layer perceptron functionality.
- How to extend the grid client.

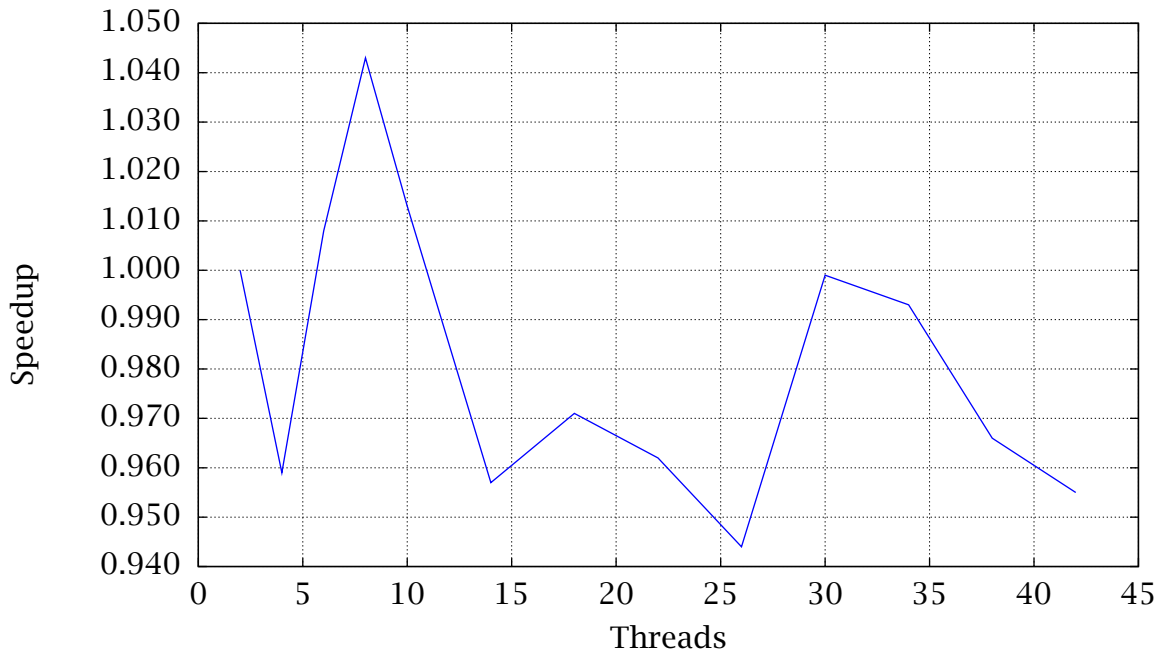


Figure 2.9: Estimated speedup including the student and staff computer cluster. Overall speedup is good but jittery. This has to be expected because the network connection is shared. E.g., downloads of other users may slow down data transmission.

2.5.1 User Manual

The most *efficient* way to access FAUN grid resources is via the command line. However, first time users will prefer the web interface. In the following the author describes both methods.

Command Line Interface

The author took special care to design the FAUN grid computing client as a drop in replacement for the FAUN kernel executable. For the end user all setup procedures stay the same:

- Provide training and validation data in the sub directory `data_files`. For unscaled data the filenames are `training_unscaled` and `validation_unscaled`. For scaled data use `training_scaled` and `validation_scaled`.
- Provide the control files in the sub directory `control_and_output_files`.

- Edit especially `control_1_1_0` suitablys. The number of networks set here is the total number of networks that are trained. The server automatically distributes smaller tasks to the clients.

- Finally start FAUN by calling the `faun` executable.

One can follow the progress of training by looking at the `output_1` file which shows every trial. `output_2` saves successfully trained network. One can parse these files with all the already available tools as the format is unchanged from previous FAUN versions. One may get a few networks more than actually required because it is not previsible at which rate clients will deliver results. If a client happens to deliver a batch of networks then, of course, superfluous networks are not discarded because they contain useful information.

Accessing the power management facility is straightforward. The `wake up` command starts all available hosts. The `shutdown` command will power off all clients which are configured to react to this command.

Web Interface

The web interface is intended for the casual user who wants fast access to FAUN functionality. [208] provides a detailed introduction to the web interface. Details for developers can be found in [207]. The following additional features are now available. Figure 2.10 on the next page shows how to use the grid computing client. By default the student and staff computer pool — ITS pool — is powered off. The institute's compute cluster is available. One should use these settings for test runs. For production runs one can wake up the ITS pool by clicking on the appropriate button. Remember to power off the cluster at the end of computation.

The *Active grid connection* box shows all clients available for computation. One will note that the same host may appear several times on different ports. This is useful for multi core systems which provide more than one client. With several clients it is possible to achieve maximum efficiency. It is even possible to involve more clients than cores. As a heuristic the author noticed that an 8 core system is loaded to full capacity with 11 clients. When one of the clients is non CPU bound, e.g., during file transfer, others take over.

Figure 2.11 on page 55 shows the *Type & topology* tab. Its structure is the same as in the previous web interface version. However, a fourth category for the shared layer perceptron is now available. As usual one can select it with the appropriate

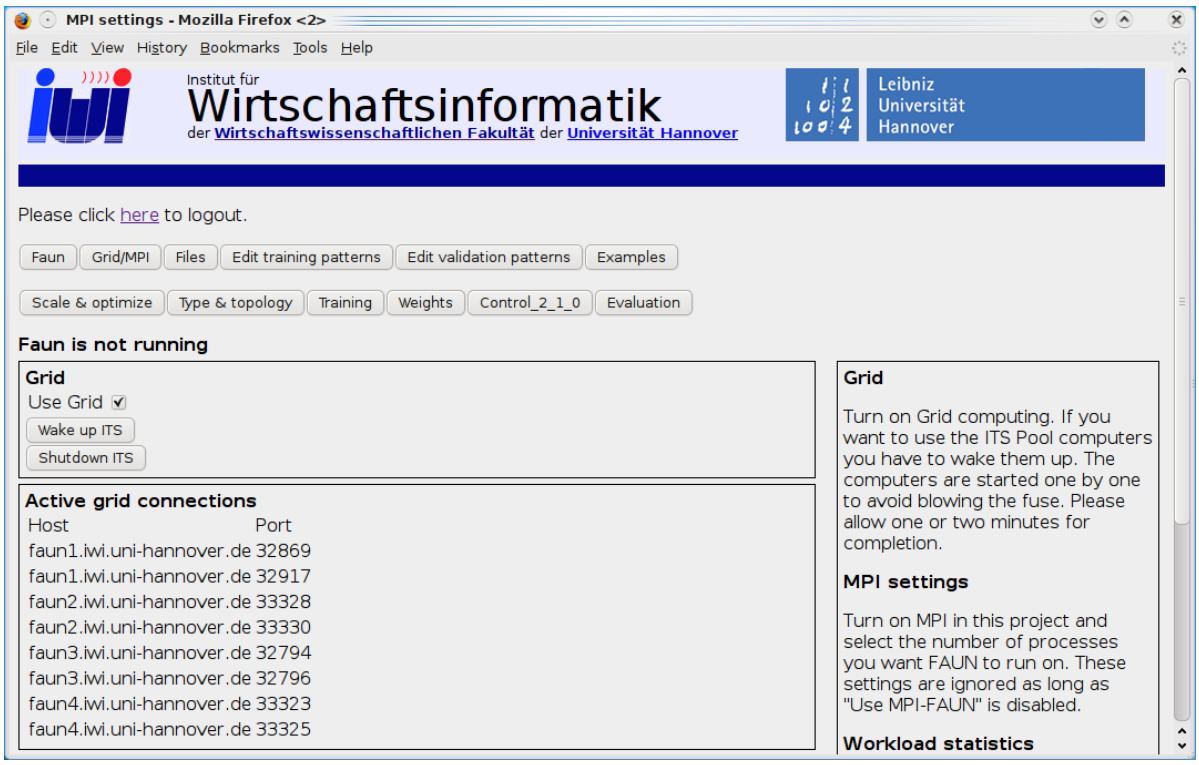


Figure 2.10: Use the grid computing client by ticking the appropriate checkbox. One can wake up and shutdown the student and staff computer cluster by clicking on the appropriate button. Note, that these are not the only clients available for computation. As a standard the institute's compute cluster is available, too.

radio button. The *Shared layer perceptron* box at the bottom of the page allows to fine tune the shared layer perceptron settings.

2.5.2 Administrator Documentation

The author considers the initial distribution of the FAUN grid computing client and the setup of the server to be an administrator and not a user task. The reason is that every institutional setup requires among others some adjustment in the connection settings. Users will not necessarily know, e.g., ports and IP addresses.

In every case one has to modify the connection settings. They are provided as module in the file `moduleConnection.rb`. This file only contains two entries. Please set them accordingly. Remember that network cards may bind to more than one IP address and choose the appropriate one. E.g., the lines

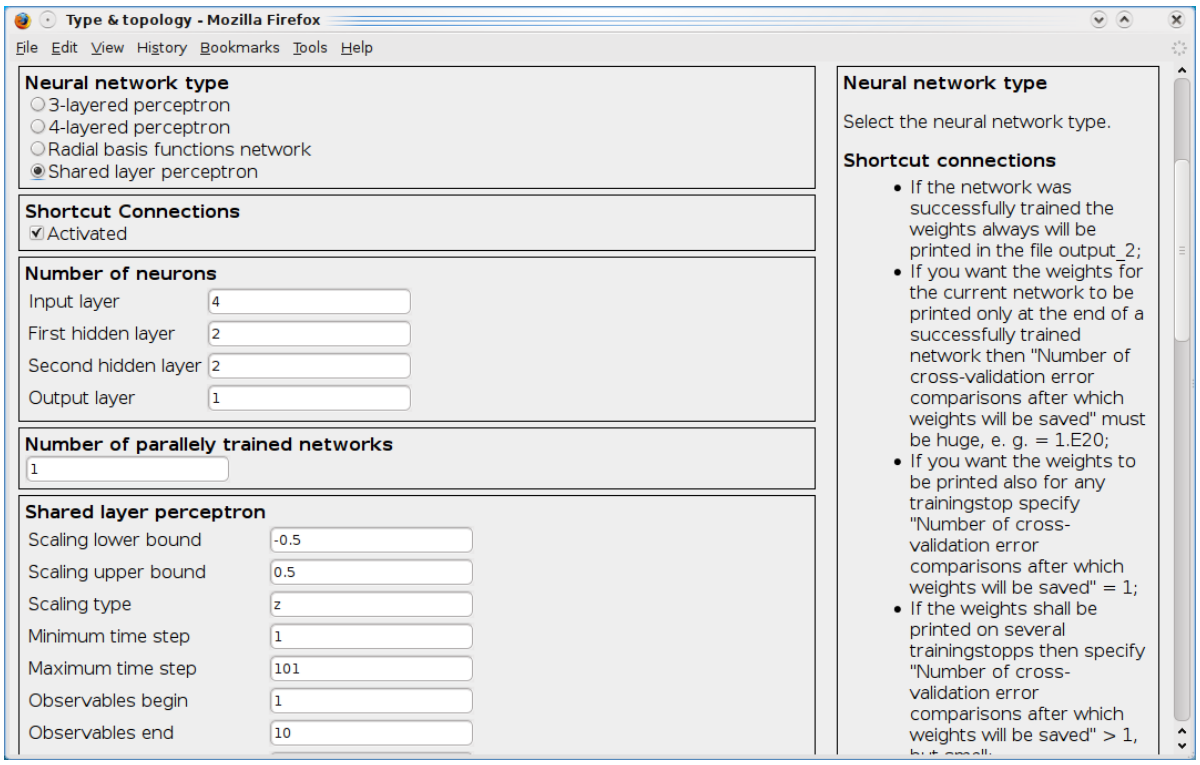


Figure 2.11: Use the shared layer perceptron by ticking the appropriate radio button. One then has the possibility to configure the topology in detail.

- `SERVER = "130.75.63.71"`
- `PORT = 1025`

will cause the server to start on the external interface with IP address 130.75.63.71 on port 1025. `moduleConnection.rb` has to be *the same* for all clients required to work in this context. The easiest way to achieve this is to modify it in the `install` directory. From this directory all necessary files are copied during setup. One can, of course, start several servers on different computers or on one computer with different ports. The server is written in pure Ruby. It is tested on different Linux and Windows flavors.

To install the client one can simply execute the `install` file. It will copy all necessary files to an appropriate destination. One can configure the destination in the `Install` class. The installer also configures the client to run at startup. Here, the mechanisms between Linux and Windows are markedly different:

- On Linux the installer creates a link in the appropriate `rc` directory. For this to work one needs to be `root`. If one modifies the link one should keep in

mind that the client needs a working network connection. It will otherwise fail writing a message to a local log file. Still, this kind of problem may be hard to debug. To avoid this the startup link should be placed at a position *after* network initialization.

- On Windows the installer creates a scheduled task. The task runs at startup and without the necessity of a user being logged in. The syntax of the task creation command can be modified in the `Install` class. A modification might be necessary on Windows XP when using localized versions of the operating system. Modifications are not needed with Windows Vista and Windows 7.

One can control the correct behavior by starting the server and wait for the client to connect. The server will log a line like

```
client '130.75.63.71 PID:10818 on i486-linux' calls get_work
```

which tells one the IP address, the process identification number and the architecture of the client platform. The process identification number is useful when several clients are running on the same host.

If this message does not appear there most probably is a problem with the firewall configuration. One should confirm this by looking at the local log file on the client. Especially when one is connecting two different subnetworks the central firewall may forbid inside-inside communication although inside-outside communication does work. This is the default case here at Leibniz Universität where two different subnetworks are not allowed to communicate except for a few standard ports. Another common problem is created by the local Windows firewall which might be overly restrictive. In this case one will have to allow the client specifically to communicate over the port one has selected. Finally, when running the server on Linux and using a privileged port, i.e., below 1024, one may need to be `root`. However, the author does not recommend any of this. One should not use a privileged port and one should not run the client as `root` for security reasons.

The installer chooses the appropriate FAUN executable to be copied to the local system. There are four files to choose from, i.e., Linux and Windows, both for 32 and 64 bit systems. These files are called `FAUNlin32`, `FAUNlin64`, `FAUNwin32` and `FAUNwin64`. If one installs on other platforms one will have to create the necessary FAUN executables yourself. This is straightforwards but necessitates a FORTRAN compiler for the target platform.

2.5.3 Developer Documentation

The code of the FAUN grid computing client and server is amply commented. It should allow one to easily alter functionality. In the following the author gives an overview of the components making up client and server.

The class `FAUNObject` is central to the functioning of client and server. It is distributed to and accessed by every client. It is managed centrally by the server. The most important switch is the `@action` class variable. It stores the state of computation. Currently there are three implemented states:

- `:doNothing` is the default state on startup. This state is also reset at the end of computation.
- `:compute` tells the client to start and continue the computation. If the previous state was `:doNothing` the client assumes that it should fetch control and data files. If however the previous state already *was* `:compute` then the client continues the computation with the already fetched data. If for any reason one wants the client to fetch new data, one has to set `:doNothing` before setting `:compute`.
- `:shutdown` causes the client to shutdown the computer. Before this happens computed results are transferred. One can therefore safely initiate a `:shutdown` without fear of loosing already available results.

In order to access state information from the server the client has to call the `get_work` method. This method will *block* until there is effectively some work to do. The advantage of using a blocking method is that the client is not obliged to poll the server regularly. Polling would generate a significant amount of network traffic. An appropriate choice of the polling interval is difficult. When returning `get_work` contains the appropriate `@action`, i.e., `:compute` or `:shutdown`.

With `get_data` the client accesses control and training data. As this will move a possibly significant amount of data across the network one should make sure to only call `get_data` when necessary. To achieve this the server assigns a unique `@id` to every new computation. The client can now simply check if the `@id` has changed and only fetch the data in this case.

The `control_1` file necessitates a special treatment. It is not enough to simply copy it, because it contains the number of networks to compute and the seed for the random generator. For every client the server generates an individual `control_1`

file with the number of networks to train and the random seed set appropriately. The client retrieves this file *always*, i.e., not only when starting the computation. This is achieved via the method `get_control_1`. When adding functionality one should keep this feature because it allows to individually set the number of networks for every client. It is, e.g., possible to first distribute several networks to train for every client. Then as results flow in the server reduces the number of networks to compute. This further reduces communication overhead.

The client uses `deliver_results` to send the results back to the server. To keep this as flexible as possible the results are simply a hash of key-value pairs. The key is the filename. The value contains the actual data. In the present case the three files `output_1`, `output_2` and `output_3` are sent back. But one can, of course, add more files as needed.

The server has to synchronize write operations of several clients. Otherwise the reporting back of the results would get mixed up. This is implemented in the `@semaphore` instance variable which locks access to output files during writes. Although this does not represent a bottleneck in the present case there might be situation where locking slows down the computation. This may happen with several hundreds of clients with very small workloads, i.e., shorter than one second. Then the writes take a significant amount of time. However, this scenario is not likely. Short workloads could easily be computed with fewer clients.

The `FAUNClient` class is responsible for all local tasks. It's principal method is `main_loop` which appropriately dispatches workloads. The `initialize` method connects to the distributed object server.

The `FAUNHelpers` class provides local auxiliary functionality for both clients and server. The important difference to `FAUNObject` is that all methods in this class are designed to work without any network connection. The distributed object is not necessary. If one wants to add functionality there are several methods of interest. `createNewFAUNDir` is responsible for setting up the FAUN directory structure from scratch. It removes old directories and sets up the new ones. If one changes how the FAUN kernel accesses control and output files one will have to make adjustments here. Most files can be passed on to the FAUN client without modification. As mentioned above `control_1` needs a special treatment and the server modifies it for every client. `parse_control_1` has the task of parsing the file. It creates a hash where every value can be modified. If one changes the structure of `control_1` one will also have to alter the parse functionality.

The module `Log` provides simple logging facilities. One should include this module in every possible extension to get unified logging. The logging facility is mostly useful for testing new functionality. Without it one will have difficulties debugging the distributed program as log messages will appear at all locations.

The module `Connection` stores connection settings for clients and server. The author already describes its application above.

When adding functionality one does not want to reinstall the client on every host. It is sufficient to simply overwrite the modified file. Changes will be incorporated at the next start. To overwrite the file it is most simple to send the file as part of a compute job. The client will notice that the sent file does not belong to FAUN control and training data and will put it in the specified location.

Targeting new platforms or operating systems creates no problems as long as a Ruby interpreter is available. One may want to check that shutdown works as intended. However, one also has to provide an appropriate executable of the FAUN kernel. The FAUN kernel does not contain any platform specific code. But, of course, one will need a FORTRAN compiler for the target platform to create it.

To summarize: extending the grid computing client is straightforward due to its modular structure. Before writing code one should check the following points:

- Do you add functionality which needs synchronization between server and client? Then put it in the class `FAUNObject` which gets distributed.
- Is the functionality needed by server *and* client, but only locally? Then the `FAUNHelpers` class is the right file to use.
- Is the functionality limited to client *or* server? Then take the local class.

2.6 FAUN Applications

Although the focus of this book is on financial forecasting the author does not want to pass over the numerous other applications which have successfully been realized with FAUN. All applications realized with FAUN are too many as that the author could describe every in detail. The selected applications should give the reader a good guidance of what is possible with FAUN. The author will expand in much more detail on forecasting applications with FAUN in chapter 4. For this reason the list given here is quite short.

Financial Forecasting

- Forecasting the German yield curve, long and short term, is one of the flagship applications of FAUN. This model, developed by Michael H. Breitner, has also been implemented in a bank. Such a system typically serves as decision support system. It helps deciding whether refinancing and/or investment best occurs at long or short term rates, see [43].
- In previous works the author analyzes short term forecasts of the EUR|USD exchange rate. This system uses high frequency data sampled at 60 seconds intervals. It forecasts the rate of return for the next 15, 30 and 45 minutes, see [324].
- Jan Bührig generates a forecast for the DAX 30 index, to appear.
- Horst-Oliver Hofmann forecasts the EEX energy future, based on electrical power prices. This leads to good short term forecasts. Here again, high frequency data is used, to appear.
- Christoph Polus analyzes the the spread of different oil brands and uncovers market inefficiencies. This study uses daily data, to appear.

Derivatives

- Michael H. Breitner first prices BASF and EUR|USD calls by using market conforming pricing models, [43].
- Patrick Bartels uses FAUN to provide real-time market valuation of options. He uses an appropriately developed web mining agent to gather the data. This leads to the WARRANT-PRO-1 software, [16–18].

Optimal Control, Simulation and Dynamic Games

- Michael H. Breitner provides tutorials on dynamic games and neural networks in [44, 46].
- Michael H. Breitner develops a model for optimal reentry guidance of a space shuttle. The simulation provides very smooth reentry paths, see, e.g., [42].
- André Meyer presents zero sum games in [249].

- Frank Köller and others realize the simulation of a callers' queue in a call center. The model allows significant savings over previous heuristics, see [19, 204, 205, 353].

- The author analyzes the enhanced cornered rat game and the homicidal chauffeur game. Both games can be seen as pursuer-evader or collision avoidance scenarios. It turns out that FAUN produces solutions which are very close to the analytical solution but much faster to compute, see [329].

Pattern Recognition

- Horst-Oliver Hofmann, Jörn-Gunnar Knie and Christoph König employ FAUN to recognize speed limit signs, seminar paper.

2.7 Summary

This chapter presents technical details of the changes the author made to the institute's neurosimulator FAUN. Part of it is creating *new* functionality, i.e., engineering. Another part consists in enhancing and ameliorating already existing functionality, hence reengineering. Mostly both parts come together and complement one another.

An important topic of this chapter is the introduction of new parallelization schemes for FAUN. With semi-automatic fine grained parallelization one can speedup the computation of *single* networks. If the task at hand requires only few networks to be trained, i.e., a few hundreds, but these have to be available fast, fine grained parallelization is a good solution. One finds that fine grained parallelization does not scale well. The results are satisfactory as long as the parallelized threads stay on a single processor. When a second processor is involved latency penalties dominate. Only a fraction of the additional processor is used. The challenge is the following: essentially, training neural networks involves linear algebra. The application of the non linear squashing function is negligible compared to computation time spent in doing matrix-matrix and matrix-vector multiplications. While these operations are theoretically easy to parallelize it turns out that copying input and writing back results takes comparatively long time. The actual computation is in the sub ms domain. This is well below the typical time slice of 10 ms of a standard operating systems.

Yet, fine grained parallelization has in the author's opinion a bright future. There are two important points:

- For the past two years increase in processor clock speed has been almost nil. On the other hand dual and quad core processors have become quite common. It is almost impossible today to buy computers with only a single processor. This development continues. Intel and AMD already sell hexa cores for standard servers and both Intel and AMD have announced octo cores to be launched in 2010. Intel also plans the release of the Larrabee processor which will share features of CPUs and GPUs. The trend clearly indicates that clock speeds stagnate or even fall and the number of cores goes up. With this development hardware and compiler vendors will *necessarily* move towards fine grained parallelization. One can expect interesting novelties within the next few years.

- Since the advent of Nvidia's CUDA in February 2007 high performance computing on the GPU has become popular. The author has no trusted source to quantify this. But he argues that seldom a month goes by without an article on GPU computing in the mainstream computer media. GPU computing generally *involves* fine grained parallelization because the number of threads executed ranges from 240–800. Today, it is still difficult to load a GPU to full capacity. But vendors, especially AMD/ATI and Nvidia, seem to be aware of the current bottlenecks. It is probable that upcoming generations of graphics hardware will facilitate effective usage of the hardware.

For both of these reasons the author plans to dedicate further research to the fine grained parallelization of FAUN. First results are already promising. The next concrete step is to create a gridded version of FAUN which makes use of additional graphics hardware when it finds it.

In the meantime coarse grained parallelization with the grid computing version of FAUN is very promising. It yields speedups of 95 percent or more and additionally offers power management. The MPI version is suitable for dedicated compute clusters. Here, upfront configuration is not a problem. One also generally *owns* — administratively or even physically — the compute cluster. However, the scenario at a typical university or company is different. One mostly has user workstations or computer pools which are idle most of the day. Administration is often not centralized. It is left to individual departments or institutes. In this case it is difficult

to justify complicated upfront configuration. The FAUN grid computing client fills the gap with the following features:

- Installation simply consists in executing the installer on Windows systems. On Linux systems one can also use the installer or install the grid computing client as a package. The administrator *once* creates a customized installer with site connection settings. Then, everybody can use this installer and automatically connects to the server.

- The FAUN grid computing version bridges different operating systems transparently. While the well known free versions of MPI do not allow hybrid setups the client and server are *operation system agnostic*. Using the platform independent Ruby programming language and distributed objects reduces the hassle of providing customized programs for every platform. Server and client have been tested with Windows XP, Windows Vista and Windows 7, and for the three latest releases of Ubuntu and the two latest releases of Debian.

- Communication is arranged such that the client initializes the connection. This represents the typical case of a firewalled network where clients are allowed arbitrary connections from the inside to the outside. But outside-inside access is generally restricted to very few standard ports. When user workstations are concerned outside-inside access is typically entirely forbidden. Contrast this to MPI where the server — or master — initiates all connections.

- The architecture is robust. If a client drops out this does not crash the computation. Conversely a client can also connect when a computation runs and will be used immediately. If the server fails the clients automatically shut down. They are not left in an undefined state. Data is not lost but available as files on the server.

- Compute clusters are *always on*. It is a waste of energy and ultimately money to let computers run idly. The FAUN grid computing client alleviates this problem by allowing the user to wake up and shutdown all involved computers remotely. In the case of the student and staff computer cluster this means savings of several thousand Euro per year. It is compared to a case where all workstations are always on. There are not only direct costs associated with having clusters always on. The student and staff computer pool is air conditioned. Each amount of energy saved in the computers is also saved at one

third in air conditioning energy costs. Additionally, consumer hardware is not designed to run permanently. Having the possibility to switch resources on and off on demand is a very important feature in the domain of high performance computing on commodity parts.

While the grid computing client is fully functionally there are further points which should be researched:

- How does the batch size affect performance? It is obvious that bigger batch sizes reduce communication overhead. On the other hand one does not want to compute *too much* networks. For the present calculations the author uses a decreasing batch size towards the end. It would be instructive to experiment with different batch size policies.

- The current architecture uses a *single* central node and only clients otherwise. In segregated networks it might be useful to introduce an additional hierarchy. All clients in a sub network then connect to a sub-server and this sub-server communicates with the main server. This reduces load on the main server and also communication on the institutional backbone.

- Clients send their data back every time they have finished a batch. This is useful for online control of the training process and allows the user to check out networks as soon as they arrive. However, the author sees the typical use case of the grid computing client as follows. *After* having determined appropriate parameters locally or on a small cluster the user runs a *batch* training job. This job runs unsupervised and online graphics are of no interest. It might be more efficient in this case to *bundle* the results and only send them back at the end of computation. There is the danger of loosing data when a client disconnects during computation.

- To further improve usability a graphical installer builder is useful. This program would advise the administrator in making the necessary connection settings and create a customized installer.

With two parallelization modes and an additional neural network topology the present book considerably enhances functionality of FAUN. As the software is written in as modular a way as possible it is the author's hope that parts of it will be reused for other projects.

3 Neural Network Topological Concepts and Enhancements

3.1 Introduction

This chapter presents a detailed analysis of a neural network topology which has been only seldom used up to today. This topology, the shared layer perceptron, is especially elegant because it treats the passage of time as perfectly symmetric. It provides parallel forecasts for several time series. Multi step forecasts are easily possible.

The chapter is structured as follows. The literature review provides the reader with all basics one might need to understand the concepts presented here. Although the shared layer perceptron topology is simple and not as convoluted as some others one may want to review some ideas, especially how neural networks are applied to dynamical systems. The reader will also see that the author uses optimization methods which are not very common in the neural network community. The reader may want to review some numerical procedures.

The following section 3.3 makes up the «meat» of the chapter. It provides motivation and mathematical formulation for the shared layer perceptron. It also shows how to compute required partial derivatives efficiently. Finally, the author analyzes computational requirements of forward and backward pass.

The next two sections present two methods of accelerating the training: teacher forcing and the addition of noise on the hidden states. The author discusses the changes needed in the algorithm.

Then, the optimization method, sequential quadratic programming, is discussed in detail. The author especially presents how the algorithm is formulated in the terminology of NPSOL, the optimizer package.

Finally the section on convergence analysis discusses the influence of the different meta parameters in the shared layer perceptron. It also provides sensible

guidance if one wants to use the shared layer perceptron and are unsure how to choose the parameters.

A summary wraps up the chapter. Here the reader will also find further ideas how to improve on the present topology.

3.2 Literature Review

Neural network introductions and basics

[176] is a classic reference and covers a broad range of topics. Coverage is — as one should expect from an introductory book — not very deep. Dynamic systems are also mentioned. Especially the section on input preprocessing on p. 144 should be read carefully. [31] is more superficial. Yet it explains basics in a more accessible manner than [176]. This also holds for [276]. [280] is an introductory overview of neural network applications. It gives a feeling of what one can actually *do* with neural networks, whereas the other introductions are more focused on theory. Starting with p. 467 [235] introduces neural networks from an information theoretical point of view. While this book has only a small part dedicated to neural networks it is still a good read — the whole book. One will get a feeling of what *models* can memorize and what not. As this book is built around the FAUN neurosimulator the author also suggests to read [43]. This provides a mathematical and thorough presentation of neural networks, model building, and applications with the added benefit that all nomenclature applies directly to FAUN and is reused in this book.

[182] proves that neural networks are in principle able to approximate every input-output mapping on an interval with required accuracy. [282] gives a similar result adapted to dynamic systems. Please note, that neither reference will tell *how* to find a network — just that it exists. This explains the plethora of literature on neural network training, see below.

Modeling of dynamic systems

[122] is *the* paper which gave birth to dynamic system modeling in the context of neural networks as one knows it today. Although it focuses on word recognition its conclusions are applicable to dynamic systems in general. [2] focuses on identifying dynamic systems.

Generally, a dynamic system model will include memory of past history. One may either model memory explicitly via inputs. Or one can provide a network internal mechanism to do this. [281] deals with long term effects. [178] offers an approach using special connections between the neurons.

Numerical methods

FAUN uses *sequential quadratic programming* for optimization. See [35] for a general introduction. [155] further focuses on merit functions. The optimizer NPSOL used by FAUN is presented in [156]. [154] documents the sub package LSSOL. NPSOL is in its essence a quasi Newton algorithm. [86] presents it. [87] analyzes global convergence of this algorithm class.

Accelerating training is the topic of various papers. [334] explains backpropagation through time for dynamic systems in detail. [338] deals with the challenges of having the network operate continuously while learning. [138] presents a boosting algorithm. [169] introduces the Marquardt algorithm in the context of neural networks. [251] proposes an accelerated conjugate gradient method. [229] presents an approach how to only consider *relevant* patterns when learning and proposes a filter. [281] proposes reinforcement learning. A totally different approach is to not use gradient methods at all but genetic algorithms, see [252]. A more general introduction is [271]. [365] presents functional reasoning, [366] uses data mining. [331] is a work in progress which reviews global optimization methods.

Training neural networks is essentially a high dimensional non linear optimization problem. [160] provides a review of possible training algorithms for these cases. [257] provides a succinct introduction. [272] reviews challenges and solutions for high dimensional optimization. [297] deals with stability issues.

One generally not only trains a single network but rather 100–10000. This allows one to only select the *best* networks for future use. Best does not necessarily mean networks with the lowest training error although it generally does. In FAUN one can also select networks based on validation error or curvature. Then, one may want to *combine* the networks to form an expert topology. [175] reviews available methods. One can also view this process as Monte Carlo simulation, see [183]. Combining neural networks gives one a *distribution* of possible outcomes. [250] exploits distributional properties.

Training requires at least first partial derivatives in most cases. [263] shows different methods to efficiently compute them dedicated to time dependent sys-

tems. [339] reviews gradient methods and the amount of computation required. For very general network topologies and first experiments it is not always possible or necessary to *explicitly* compute the derivatives by hand. This task is successfully carried out by automatic differentiation. [164] is an introduction to automatic differentiation. [29, 30] introduces the ADIFOR software package, [312] the OpenAD package. [174] introduces another good software package, Tapenade, and also introduces automatic differentiation very concisely.

Learning dynamic systems with long history may cause the error flow to become very small for certain time steps. This is known as *vanishing gradients*. The author didn't encounter the effect in this book. But the discussion in [177] is useful in its own right. [294] proposes a method to mitigate vanishing gradient effects.

When learning patterns one may argue that there is uncertainty in the observed values. To improve generalization capabilities one may add noise, see [181]. The author uses noise on the hidden states and find that it helps escape local minima at the beginning of training. [330] presents this approach, but for the outputs. Another useful modification of the learning process is teacher forcing. It replaces forecasts by desired outputs. This is discussed very briefly in [176], p. 817. [307] give mores details.

3.3 Shared Layer Perceptron

3.3.1 Mathematical Formulation

The basic idea is to model a dynamic system which forecasts every component of the system as a whole. In simple words: the system produces its own forecast. As one cannot assume that one is able to observe *everything* that is important for the system one adds hidden states which absorb the unknowns. Think about it: just because one considers something is *meaningful* it does not follow that other variables are *not meaningful*. Just look into your world: which variables do *you* see? A few thousand, perhaps a few millions? And how many variables are relevant to the world? One cannot account for all these variables in the model. And one cannot measure them with arbitrary accuracy. This is not the goal. But one can build a model that at least acknowledges hidden influences and does not ignore them. To be specific: just because one takes a fancy in forecasting exchange rates they will not suddenly become something special. They continue to move, together with eq-

uity indices, interest rates, commodities, economic indicators, market actors, other humans, molecules, atoms, and elementary particles. One has a coherent system.

We can model such a system with a $D \times D$ matrix of weights W which encodes the dynamics. At every time t one has a D-dimensional state vector $\mathbf{s}^t$. The first N elements are the observeables. These are the variables one can measure, deem *especially* important for the system and want to predict *as a whole*. The next $D - N$ elements are hidden states. For state transitions one uses the following simple neural network formulation:

$$\mathbf{s}^{t+1} = \mathbf{f}(\mathbf{s}^t). \tag{3.1}$$

In the present case one has

$$\mathbf{s}^{t+1} = \tanh(W\mathbf{s}^t). \tag{3.2}$$

Please note, that tanh is applied element-wise to the vector $W\mathbf{s}^t$. Figure 3.1 on the next page illustrates the shared layer perceptron.

Optimization algorithms like NPSOL generally need to know at least the first partial derivative of the cumulated error function E with respect to the weights of the weight-matrix W. As usually *all* partial derivatives are needed at every optimization step it is important to devise computationally efficient ways to calculate the gradient. Depending on the given algorithm and the relationship of input variables, i.e., weights, and output variables, i.e., E or the elements of the Jacobian, it is better to calculate the gradient using forward or reverse accumulation in the chain rule. Using forward accumulation is more intuitive. Because of that the author presents this method first. Reverse accumulation is known as the classical backpropagation algorithm. The author presents it second.

3.3.2 Forward Accumulation

First, the author looks qualitatively at the gradient calculation and note the following: The gradient of the error function E depends on the gradients of the local error ε^t at every time step $t = 1 \dots T$. These local errors again depend on the values of the state space $\mathbf{s}^t$. Finally, every $\mathbf{s}^t$ depends directly on the previous state space at $t - 1$ because

$$\mathbf{s}^t = \mathbf{f}(\mathbf{s}^{t-1}). \tag{3.3}$$

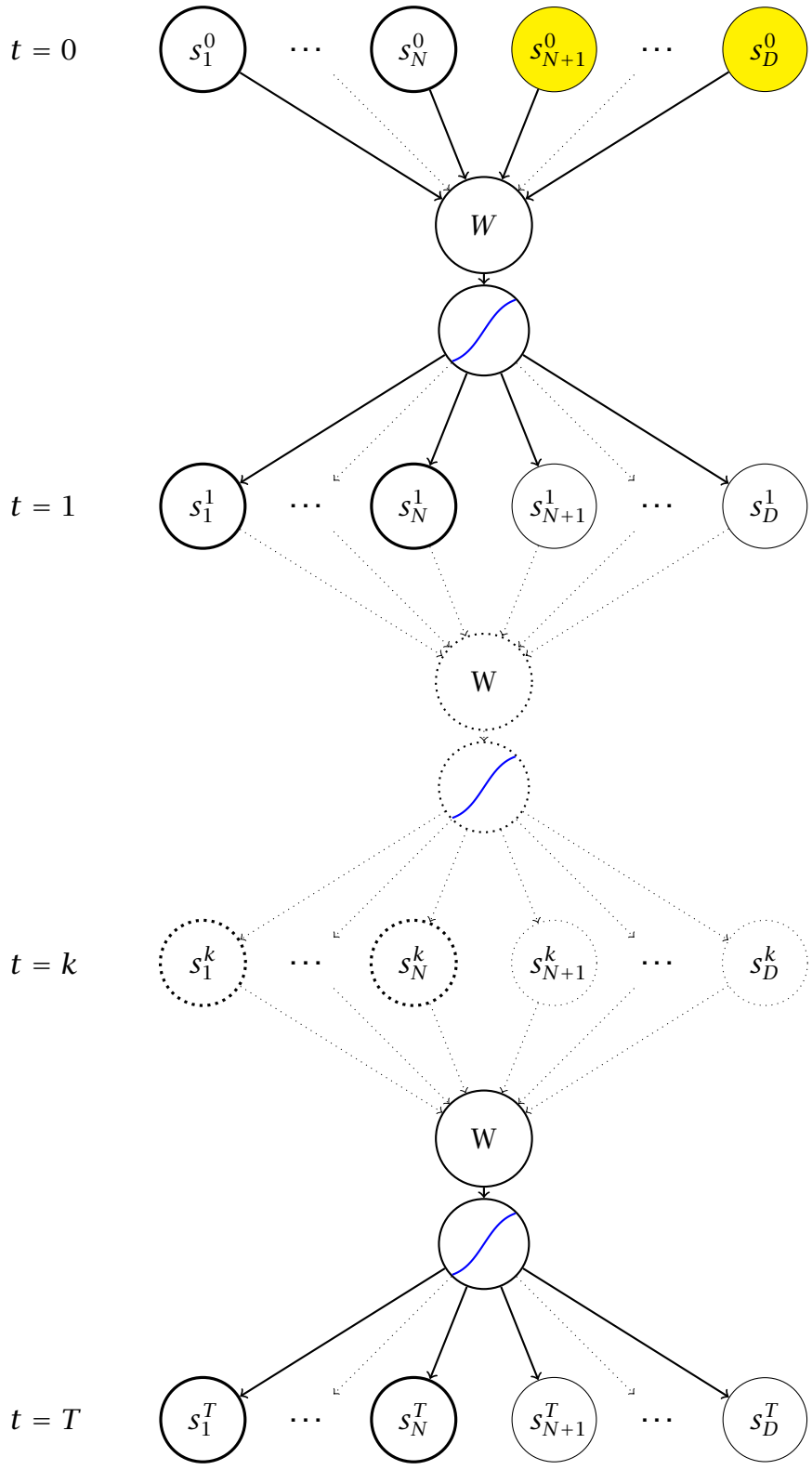


Figure 3.1: Modeling with the shared layer perceptron. The initial hidden states in yellow can also be trained.

One therefore get a recursive dependence of $\partial_{i,j}\mathbf{f}_m^t(\mathbf{s})$ on $\partial_{i,j}\mathbf{f}_k^{t-1}(\mathbf{s})$ for $k = 1 \ldots D$. The author uses the abbreviation

$$\partial_{i,j} := \frac{\partial}{\partial w_{i,j}} \tag{3.4}$$

to avoid cluttering the notation. One gets

$$\begin{aligned}
\partial_{i,j}\mathbf{f}_m^{t+1}(\mathbf{s}) &= \partial_{i,j} \tanh(W \cdot \mathbf{f}^t(\mathbf{s}))_m \\
&= \partial_{i,j} \tanh(\sum_{n=1}^{N} w_{m,n} \cdot \mathbf{f}_n^t(\mathbf{s}) \\
&= \tanh'(\sum_{N=1}^{N} w_{m,n} \cdot \mathbf{f}_n^t(\mathbf{s})) \cdot \sum_{n=1}^{N} \partial_{i,j}(w_{m,n} \cdot \mathbf{f}_n^t(\mathbf{s})) \\
&= [1 - \tanh^2(\sum_{n=1}^{N} w_{m,n} \cdot \mathbf{f}_n^t(\mathbf{s}))] \cdot \sum_{n=1}^{N} [(\partial_{i,j} w_{m,n}) \cdot \mathbf{f}_n^t(\mathbf{s}) + w_{m,n} \cdot \partial_{i,j}\mathbf{f}_n^t(\mathbf{s})]
\end{aligned}$$

One notes that

$$\partial_{i,j} w_{m,n} = \begin{cases} 1 & \text{if } i = m \text{ and } j = n \\ 0 & \text{otherwise.} \end{cases}$$

One can further simplify using

$$[1 - \tanh^2(\sum_{n=1}^{D} w_{m,n} \cdot \mathbf{f}_n^t(\mathbf{s}))] = 1 - (\mathbf{f}_n^{t+1}(\mathbf{s}))^2.$$

One can do this because all values of $\mathbf{f}_n^t$, $t = 1 \ldots T$ have already been calculated, or are at least calculated simultaneously. One needs the values anyway, e.g., for estimation of the local errors. Finally, one sees, that the rightmost summand in the right sum over n is easily expressed as matrix-vector multiplication

$$\sum_{n=1}^{D} w_{m,n} \cdot \partial_{i,j}\mathbf{f}_n^t(\mathbf{s}) = (W \cdot \partial_{i,j}\mathbf{f}^t(\mathbf{s}))_m$$

where $\partial_{i,j}\mathbf{f}^t(\mathbf{s})$ is the vector of a partial derivative with respect to every component of the state space, i.e.,

$$\partial_{i,j}\mathbf{f}^t(\mathbf{s}) = \begin{pmatrix} \partial_{i,j}\mathbf{f}_1^t(\mathbf{s}) \\ \vdots \\ \partial_{i,j}\mathbf{f}_D^t(\mathbf{s}) \end{pmatrix}.$$

This allows us to write the vector of partial derivatives very elegantly using vector and matrix operations. One gets

$$\partial_{i,j}\mathbf{f}^{t+1}(\mathbf{s}) = (\mathbf{1} \ominus (\mathbf{f}^{t+1}(\mathbf{s}))^2) \otimes (\mathbf{f}^t_{\Delta_{i,j}}(\mathbf{s}) \oplus \partial_{i,j}\mathbf{f}^t(\mathbf{s}))$$

or, even more succinctly, dropping the start vector $\mathbf{s}$

$$\partial_{i,j}\mathbf{f}^{t+1} = (\mathbf{1} \ominus (\mathbf{f}^{t+1})^2) \otimes (\mathbf{f}^t_{\Delta_{i,j}} \oplus \partial_{i,j}\mathbf{f}^t). \tag{3.5}$$

Here, $\mathbf{1}$ represents the D-dimensional vector filled with 1 and the m-th component of $\mathbf{f}^t_{\Delta_{i,j}}$ is defined as

$$(\mathbf{f}^t_{\Delta_{i,j}})_m = \begin{cases} \mathbf{f}^t_j & \text{if i = m} \\ 0 & \text{otherwise.} \end{cases}$$

The operators $\ominus, \oplus, \otimes$ indicate component-wise vector or matrix operations. To further utilize high performance matrix algorithms in forward accumulation mode one uses the following mapping. Every pair i, j of actually used weight indices in the weight matrix W is mapped onto a corresponding *single* index k, $k = 1 \ldots K$. This is realized unambiguously by starting in the first row, $i = 1$, and mapping for all indices j, $j = 1 \ldots D$. Then continue in the second row, $i = 2$, map the columns, and so on. The mapping concludes with the last matrix row, $i = D$. For a fully connected matrix W one gets the mapping

$$\begin{array}{ll} (i,j) & \rightarrow \quad k \\ (1,1) & \rightarrow \quad 1 \\ (1,2) & \rightarrow \quad 2 \\ & \vdots \\ (1,D) & \rightarrow \quad D \\ (2,1) & \rightarrow \quad D+1 \\ & \vdots \\ (D-1,D) & \rightarrow \quad (D-1)\cdot D \\ (D,1) & \rightarrow \quad (D-1)\cdot D + 1 \\ & \vdots \\ (D,D) & \rightarrow \quad D \cdot D. \end{array}$$

Keep in mind that for a sparse representation some indices are missing. It is advisable to only use the indices of the weights which are actually modified to avoid

additional computing costs. One defines the matrix

$$\partial F^{t+1} := \begin{pmatrix} \partial_{k=1}\mathbf{f}^{t+1} \\ \vdots \\ \partial_{k=K}\mathbf{f}^{t+1} \end{pmatrix}. \tag{3.6}$$

∂F^{t+1} contains as *row* vectors the corresponding partial derivatives with respect to every weight and every element of the state space. In detail, one has

$$\partial F^{t+1} = \begin{pmatrix} \partial_{k=1}\mathbf{f}_1^{t+1} & \partial_{k=1}\mathbf{f}_2^{t+1} & \dots & \partial_{k=1}\mathbf{f}_D^{t+1} \\ \partial_{k=2}\mathbf{f}_1^{t+1} & \partial_{k=2}\mathbf{f}_2^{t+1} & \dots & \partial_{k=2}\mathbf{f}_D^{t+1} \\ \dots & \dots & \dots & \dots \\ \partial_{k=K}\mathbf{f}_1^{t+1} & \partial_{k=K}\mathbf{f}_2^{t+1} & \dots & \partial_{k=K}\mathbf{f}_D^{t+1} \end{pmatrix}.$$

Analogously one defines the matrix

$$F_\Delta^t := \begin{pmatrix} \mathbf{f}_{\Delta_{k=1}}^t \\ \vdots \\ \mathbf{f}_{\Delta_{k=K}}^t \end{pmatrix}$$

as matrix of row vectors. This allows to rewrite equation 3.6 as

$$\partial F^{t+1} = \underbrace{(\mathbf{1} \ominus (\mathbf{f}^{t+1})^2)}_{\text{row vector}} \otimes \underbrace{(F_\Delta^t \oplus \partial F^t)}_{\text{matrix}}. \tag{3.7}$$

Note that $\otimes$ means that one multiplies each of the D *columns* of the matrix $(F_\Delta^t \oplus \partial F^t)$ with the corresponding value of the row vector $(\mathbf{1} \ominus (\mathbf{f}^{t+1})^2)$. This allows for fine grained parallelization using the appropriate matrix algorithms.

Finally, one needs a start point for the recursion that gives us the initial equation for $t = 1$. Note that one doesn't need partial derivatives at $t = 0$. Indeed, the initial state $\mathbf{s}$ is entirely determined by the first observation and the hidden states. These hidden states may be randomly initialized or trained themselves. But in any case they don't depend on the weight matrix W. To be specific, all partial derivatives with respect to all weights are zero at $t = 0$, i.e.

$$\partial F^0 = \mathbf{0},$$

where $\mathbf{0}$ is the zero matrix. Equation 3.7 on the previous page yields the simple

initial equation

$$\partial F^1 = \underbrace{(\mathbf{1} \ominus (\mathbf{f}^1)^2)}_{\text{row vector}} \otimes \underbrace{(F^1_\Delta)}_{\text{matrix}}.$$

These values are available because $\mathbf{f}^1$ and F^1_Δ only depend on the initial state $\mathbf{s}$.

Ultimately one is interested in the partial derivatives with respect to the *error function*. Using the notation from above one gets

$$\partial_{i,j} E = \frac{1}{2} \sum_{t=1}^{T} \partial_{i,j} \varepsilon^t$$

where ε^t is the usual local cumulated squared error at time t, i.e.

$$\varepsilon^t = \sum_{k=1}^{N} (\mathbf{x}^t_k - \mathbf{s}^t_k)^2.$$

The chain rule of calculus leads to

$$\begin{aligned} \partial_{i,j} \varepsilon^t &= \sum_{k=1}^{N} \partial_{i,j} (\mathbf{x}^t_k - \mathbf{s}^t_k)^2 \\ &= \sum_{k=1}^{N} 2(\mathbf{x}^t_k - \mathbf{s}^t_k) \cdot \partial_{i,j} (\mathbf{x}^t_k - \mathbf{s}^t_k) \\ &= -2 \sum_{k=1}^{N} (\mathbf{x}^t_k - \mathbf{s}^t_k) \cdot \partial_{i,j} \mathbf{s}^t_k \end{aligned}$$

because the observeables $\mathbf{x}^t_k$ do not depend on $w_{i,j}$. If one agrees to only take into account the first N values of $\mathbf{f}^t$ one can write the above equation somewhat sloppily using the dot product

$$\partial_{i,j} \varepsilon^t = -2(\mathbf{x}^t - \mathbf{f}^t) \cdot (\partial_{i,j} \mathbf{f}^t). \tag{3.8}$$

The above formulation leads to very efficient implementation. In terms of efficiency one can even improve on equation 3.8 using a matrix vector multiplication to obtain $\partial_{i,j} \varepsilon^t$ for all index pairs (i, j) in one step. Because of the commutativity of the dot product one can alter the order of both factors in 3.8. One defines the row vector of local errors

$$\boldsymbol{\varepsilon}^{\mathbf{t}} := (\mathbf{x}^t - \mathbf{f}^t)$$

and the vector of all partial error derivatives at time t

$$\partial \varepsilon^t := (\partial_1 \varepsilon^t, \partial_2 \varepsilon^t, \ldots, \partial_K \varepsilon^t).$$

Note that in the above equation the author again uses the mapping from an index pair to a single index $(i, j) \to k$ for ease of notation. Using these definitions one can write the vector formulation of equation 3.8 on the facing page elegantly as

$$\partial \varepsilon^t = -2F^t \cdot \varepsilon^t. \tag{3.9}$$

Finally, the total error vector of partial derivatives $\partial \mathbf{E}$ is given by

$$\partial \mathbf{E} = \sum_{t=1}^{T} \partial \varepsilon^t. \tag{3.10}$$

3.3.3 Reverse Accumulation

As stated above reverse accumulation involves traversing the chain rule from the outer to the inner part. It is the classical form of the backpropagation algorithm. *Errors* are propagated back until one sees the influence on the weights. For this one should first recall the following: at every time step t the backpropagated error consists of two part. On the one hand one has the direct error ε_j^t. This error is simply the difference of target value and output value. On the other hand one also adds the backpropagated error from previous time steps. This allows us to compute the *local gradient*

$$l_j^t = \tanh'(W \mathbf{s}_j^{t-1}) \left(\varepsilon_j^t + \sum_{k=1}^{D} w_{j,k} l_k^{t+1} \right) \tag{3.11}$$

Equation 3.11 describes how $\mathbf{s}_j^t$ changes depending on $W \mathbf{s}_j^{t-1}$. Because of

$$\tanh'(\cdot) = 1 - \tanh^2(\cdot)$$

one may simplify equation 3.11 to

$$l_j^t = (1 - (\mathbf{s}_j^t)^2) \left(\varepsilon_j^t + \sum_{k=1}^{D} w_{j,k} l_k^{t+1} \right)$$

for the shared layer perceptron. This is computationally much more efficient because one has to calculate the $\mathbf{s}^t$ anyway. Doing this iteratively one gets local gradients for all times from $t = T$ down to $t = 1$. One gets the partial derivative of total error E with respect to weight $w_{i,j}$ by weighting all local gradients appropriately:

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{t=1}^{T} l_i^t \mathbf{s}_j^{t-1}.$$

One can implement reverse accumulation very elegantly again using matrix formulation. Each sum in the above equation translates into a vector dot product. Doing this for all D dimensions leads to

$$\mathbf{l}^t = (1 \ominus (\mathbf{s}^t)^2) \otimes (W' \mathbf{l}^{t+1} \oplus \varepsilon^t).$$

The operations $\ominus, \otimes, \oplus$ refer to component-wise vector operations. W' is the transposed of W. As initial condition one sets

$$\mathbf{l}^T = \varepsilon^T.$$

Having computed all the local gradients one gets the $D \times D$ matrix ∂E with partial derivatives with respect to *all* weights:

$$\partial E = L' \cdot S.$$

Here, L is the matrix containing the local gradients,

$$L = (\mathbf{l}^1, \ldots, \mathbf{l}^T),$$

and S the matrix of the first $T - 1$ states,

$$S = (\mathbf{s}^0, \ldots, \mathbf{s}^{T-1}).$$

If one wants to also train the initial hidden states one needs the partial derivatives

$$\frac{\partial E}{\partial \mathbf{s}_{N+1}^0}, \ldots, \frac{\partial E}{\partial \mathbf{s}_D^0}.$$

One gets these, again, by using the appropriate local gradient,

$$\partial \mathbf{E}_{\mathbf{s}^0} = W' \mathbf{l}^1.$$

Obviously, one only takes the components $N+1, \ldots, D$ of $\partial \mathbf{E}_{\mathbf{s}^0}$ because only those are sensibly trained.

3.3.4 Computational Requirements

FAUN spends the dominant amount of computation time in evaluating error functions and derivatives. For the multi layer perceptron one will find the relevant dependencies in the literature. The author provides them here for the shared layer perceptron. Computation depends mainly on

- D, the number of dimension of state vector and weight matrix,
- T, the number of time steps used for training and validation.

The number of observeables $N < D$ also plays a role, but to a lesser extent. As long as one is using dense matrices the sparsity level s does not influence computation time.

Forward pass

Each forward pass consists mainly in T matrix-vector operations. Viewing the matrix as slices of vectors one notes that multiplying two D-dimensional vectors costs D multiplications and $D-1$ additions. As today's processors execute these operations in the same time one will call them floating point operations, i.e., flops. The vector-vector multiplications needs $2D-1$ flops and the matrix-vector operation $D(2D-1)$ flops. For all time steps one gets

$$T \cdot D(2D-1) = O(T \cdot D^2) \text{ flops}$$

using Landau notation. Additionally one requires $T \cdot D$ transcendental function calls for the application of tanh.

Error

Computing the error is a simple matter of calculating differences. This costs $T \cdot N$ additions and is of the order

$$O(T \cdot D)$$

considering $N < D$.

Forward accumulation

Assuming a dense weight matrix equation 3.7 first leads to the addition of 2 $D^2 \times D$ matrices, therefore $D^2 \cdot D$ additions. Building the row vector needs D multiplications and D subtractions. Multiplying the row vector with the matrix leads again to $D^2 \cdot D$ multiplications. In total one gets for T time steps

$$T(D^2 \cdot D + 2D + D^2 \cdot D) = T(2D^3 + 2D) = O(T \cdot D^3) \text{ flops}$$

just for the derivative matrices $\mathbf{F}^t$. Calculating the error derivatives adds $T \cdot 2D^2 \cdot D$ flops but does not change the order.

Reverse accumulation

The dominant computation for the T local gradients is the matrix-vector multiplication $W'\mathbf{l}^{t+1}$. This costs again $D(2D-1)$ flops. Adding the errors is another D flops. Building the «left» factor of the product costs $2D$ flops. The final multiplication adds D flops. Computing all local gradients leads to

$$T(D(2D-1) + D + 2D + D) = T(2D^2 + 3D) = O(T \cdot D^2) \text{ flops.}$$

The final matrix-matrix multiplication involves the $D \times T$ matrix L' and the $T \times D$ matrix S. This costs again

$$D^2 \cdot (T + T - 1) = D^2(2T - 1) = O(T \cdot D^2) \text{ flops.}$$

To summarize: using reverse accumulation one gets computational costs at the order $O(T \cdot D^2)$. Forward accumulation costs us one order of magnitude in D more. Where forward accumulation might be at first sight easier to compute one sees that reverse accumulation is actually much cheaper and also much more elegant to real-

ize. This is a direct consequence of the automatic differentiation rule stating that $\mathbf{R} \to \mathbf{R}^n$ functions are best treated using conventional forward accumulation. On the other hand $\mathbf{R}^n \to \mathbf{R}$ functions are more efficiently treated with reverse accumulation. Calculating the error function involves the latter case. To be fair: forward accumulation delivers a considerable amount of intermediary results. If one uses an optimization algorithm which puts these results to good use this might warrant the additional computational complexity. However, NPSOL only needs the derivatives. In the following the author uses reverse accumulation.

Until now the computational requirements analysis only considers *dense* weight matrices. Using a sparsity level of s would reduce the forward pass and the local gradient operations to this fraction s. The last matrix-matrix computation cannot be reduced because both matrices, local gradients and states, are a priori dense. Multiplying by a constant does not change the order of magnitude required:

$$O(sT \cdot D^2) + O(T \cdot D) + O(sT \cdot D^2) + O(T \cdot D^2) = O(T \cdot D^2).$$

However, taking $s = 0.10$ one reduces the theoretical amount of computation to only 10 percent of the original amount. This looks very attractive. Experiments with sparse matrix algorithms showed almost no speed gain. One explanation for this is that sparse storage only allows to access columns *or* rows efficiently; but not *both*. In the forward pass one needs the original W and in the backward pass the transposed W'. The analysis of sparse weight matrices seems worthwhile. The author plans to further work on this.

3.4 Teacher Forcing

When training error is high this can have two reasons:

- The weights are not right. One addresses this conventionally by training the network and adjusting the weights.
- One is in the wrong part of the state space. I.e., adjusting the weights is a futile effort.

The concept of teacher forcing addresses the latter issue. It gently pushes the network to operate in the appropriate region of the state space. This is realized by

replacing the observeable output with the desired outcome, see figure 3.2 on the next page for an illustration.

Note that when using teacher forcing one is actually changing the error function. One has to modify the derivative calculation at the following points. First, one overrides the backpropagated error and only inject local error. Second, one sees that teacher forcing modifies the *input* for the subsequent computation. This leads to the following change:

$$l_j^t = (1 - (\mathbf{s}_j^t)^2)\varepsilon_j^t \text{ for } j = 1, \ldots, N.$$

One also has to remember that the states used in subsequent computations are *outputs*, i.e., the result of one step of the network. The *inputs* — which are teacher forced — are only used in the forward pass.

3.5 Noise

Especially at the beginning of training one wants to avoid one of the very many bad local minima. A common technique is the addition of error dependent noise on the hidden states. Figure 3.3 on page 82 illustrates this. The noise δ is drawn from a Gaussian distribution according to

$$\delta = \min\left\{\kappa, E \cdot N(\mu = 0, \sigma^2)\right\}. \tag{3.12}$$

κ represents a small value, e.g., $\kappa = 0.1$ and represents the *maximum* amount of noise that will be added to a hidden state. Don't be confused by the *minimum* in equation 3.12. It ensures that even if the noise term is too big one won't saturate a neuron. E is the training error. As E decreases the amount of noise added also decreases. N is a normally distributed random variable with zero mean. Standard deviation σ should also be set to a small amount, e.g., $\sigma = 0.1$. This ensures that one generally does not clip the noise at κ.

There are more sophisticated possibilities of using noise. An interesting approach is to use noise dependent on the *local error* of the state space element. The author researched this. But experiments failed to improve on the much simpler version reported here. For this reason the author decided to stick with the present model. However, this is not to say that improvements are not possible.

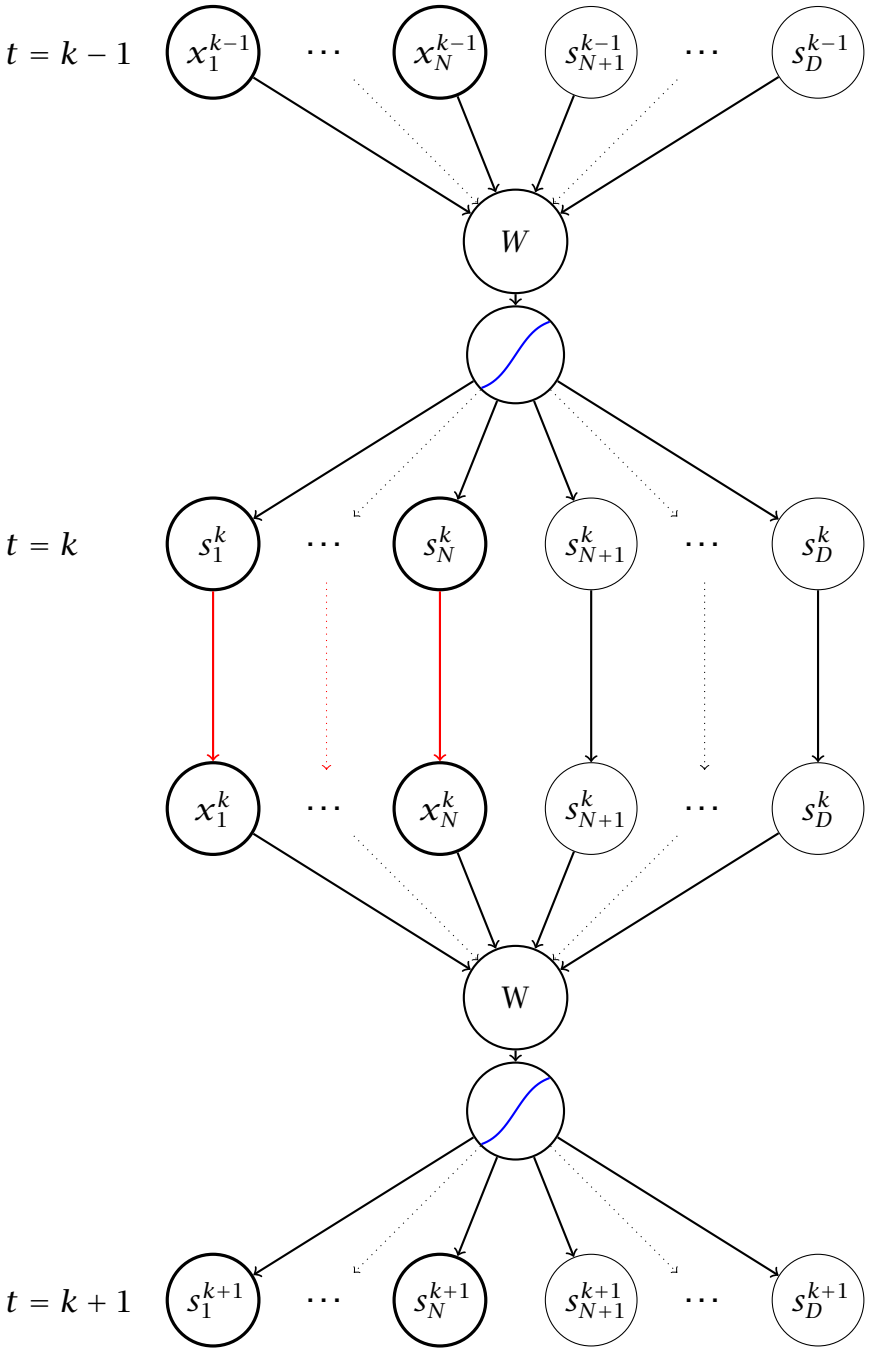


Figure 3.2: Illustration of teacher forcing. After the error has been computed the observeable states are replaced with the desired output, red arrows. Note that the error and the derivatives slightly change. However, the added computational complexity is neglegible.

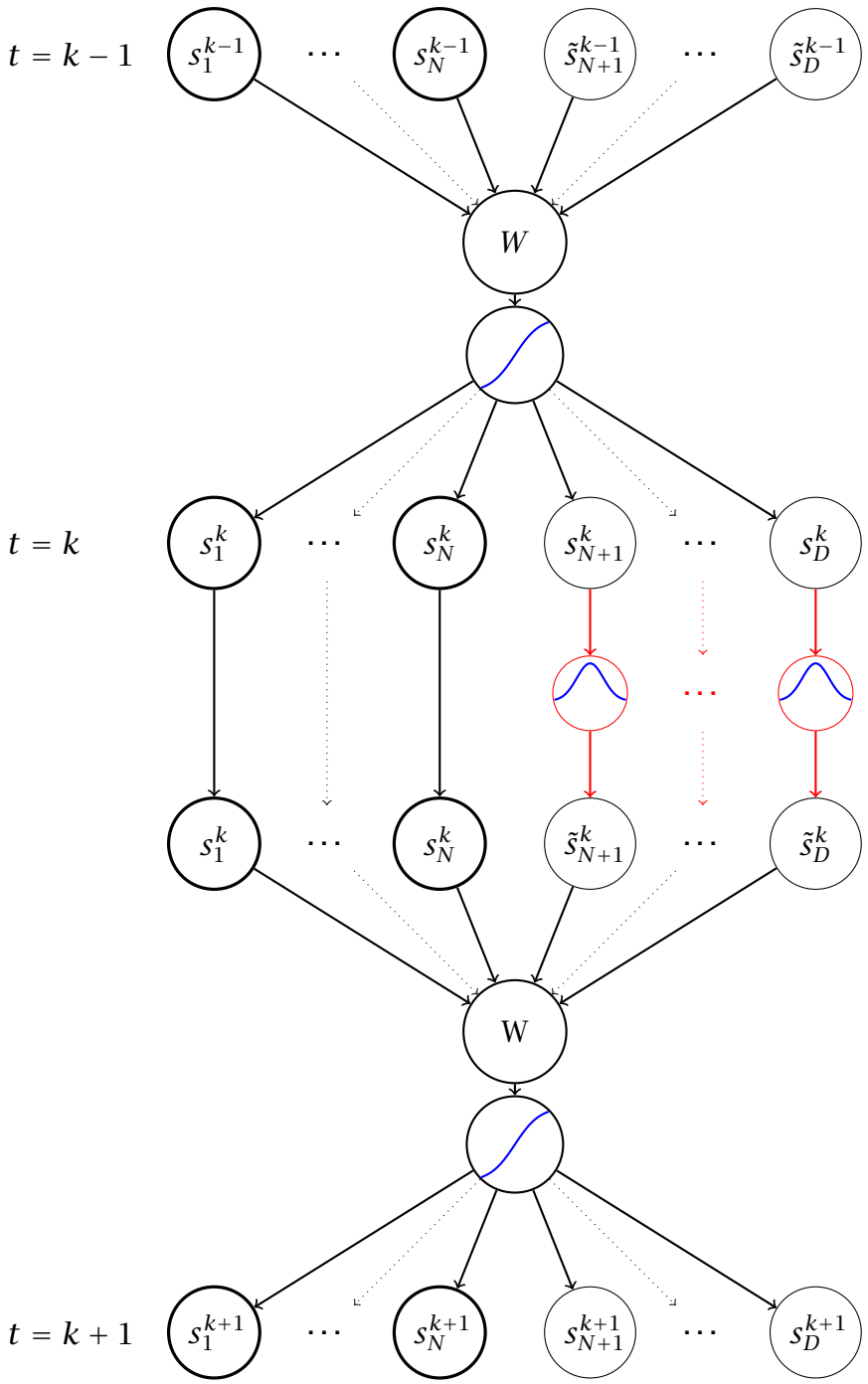


Figure 3.3: Illustration of additive noise on the hidden states. At the beginning of training Gaussian noise is added to the hidden states. This avoids initial entrapment in one of the many local minima. As training continues noise reduces according to training error and is finally switched off towards the end.

3.6 Optimization with SQP Methods

FAUN uses the nonlinear programming optimization package NPSOL, see [156]. The part of NPSOL is the following: One wants to minimize training error E on the available historical data. The error depends on weights in matrix W. It also depends on the initial hidden states, $s^0_{N+1}, \ldots, s^0_D$. These can also be trained. NPSOL evaluates the error and the derivatives. It then modifies weights — and initial hidden states — accordingly.

In the following the author discusses usage of NPSOL as it is pertinent to the problem. For a complete description of NPSOL please refer to [156]. Indeed, FAUN only uses some part of the NPSOL functionality. Especially NPSOL's ability to accommodate linearly dependent and non-linear constraints is not relevant for FAUN.

The function to minimize is E, the overall training error. This depends on the weight matrix W. Additionally one wants the weights to stay within certain bounds. This is important because otherwise a single weight could saturate the corresponding neuron. See also the discussion on convergence. For the shared layer perceptron an interval of $w \in [-2;2]$ has proven acceptable. Therefore one may incur a trade off in training error when hitting the bounds. However, in the author's experience, the bounds are rarely active. They become active mostly when the weights are initialized on too large an interval. Then, the training may fail to converge anyway.

One has a non-linear objective function and constant constraints on W. Assume one has V active weights. Then one wants to

$$\begin{gathered} \text{minimize } E(W_V),\ W_V \in \mathbb{R}^{n\times n} \\ \text{with } l \le W_V \le u. \end{gathered} \tag{3.13}$$

The subscript V indicates that one only minimizes on the V active weights of W according to the mapping specified above. Somewhat abusing notation one takes W_V as a vector of weights which are appropriately mapped back to W in the following. This means that for the jth active weight the equation $l_j \le W_{Vj} \le u_j$ holds. One has to produce values for every component of l and u. If one wants to leave the jth component free one can do so by setting a very low or very high value. In the present case one typically takes $l_j = -2$ and $u_j = 2$.

In the context of NPSOL one wants the constraints to be satisfied within a certain tolerance δ. For $W_{Vj} < l_j - \delta$ the corresponding jth constraint is violated. For $l_j - \delta \le W_{Vj} \le l_j + \delta$ the lower constraint is active. Conversely if $l_j + \delta < W_{Vj} < u_j - \delta$

the jth constraint is inactive. If $u_j - \delta \leq W_{Vj} \leq u_j + \delta$ the upper constraint is active and it is violated for $W_{Vj} > u_j + \delta$. NPSOL takes the lower and upper constraints l and u as input parameters. Additionally one also sets starting values for the active weights W_V. These values are generally initialized randomly. One also has to furnish the function $E(W)$. NPSOL also needs the partial derivatives of $E(W)$ in respect to every W_{Vj}. These partial derivatives could be calculated on the fly by NPSOL using finite-differences approximations. It has the disadvantage of reduced accuracy compared to exact derivatives. It is better to provide analytic derivatives. For the shared layer perceptrons the derivation is shown above. For more complicated topologies automatic differentiation may be used. Some further information on automatic differentiation is provided in the literature review. It is possible to implement another error function, e.g., if one wanted to optimize not for training error, but, e.g., for a financial measure. In this case one would have to keep in mind to provide new derivatives for NPSOL. For first experiments with new error functions it is still possible to use finite-difference approximations and the author recommends this. This functionality has helped the author more than once when debugging obviously erroneous analytic derivatives. One will quickly discover, e.g., sign errors or errors which differ by a multiple.

NPSOL uses a sequential quadratic programming (SQP) algorithm which the author describes in detail adapted to the relevant parts of the shared layer perceptron in FAUN. An overview of SQP methods is given in [35]. One sets $g(W_V)$ as being the vector of first partial derivatives of $E(W_V)$, i. e.

$$g_j(W_V) = \frac{\partial E(W_V)}{\partial W_{Vj}}.$$

Analogously in NPSOL nomenclature J is the Jacobian matrix of the first partial derivatives of the constraints. For FAUN this reduces to

$$J_{i,j} = \frac{\partial W_{Vi}}{\partial W_{Vj}} = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$$

The first-order conditions for optimality of (3.13) for an acceptable point c are satisfied, if one has:

- A vector of Lagrange multipliers λ exists so that the gradient of $E(W_V) -$

$\lambda^T W_V = 0$. One calls $E(W_V) - \lambda^T W_V$ the *Lagrangian*. This leads to

$$g(W_V) = J(W_V)^T \lambda$$

i. e.

$$\begin{pmatrix} \frac{\partial E(W_V)}{\partial W_{V1}} \\ \vdots \\ \frac{\partial E(W_V)}{\partial W_{Vn}} \end{pmatrix} = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$$

or

$$\lambda_j = \frac{E(W_V)}{\partial W_{Vj}}.$$

- $\lambda_j = 0$ if $l_j < W_{Vj} < u_j$, $\lambda_j \geq 0$ if $l_j = W_{Vj}$, $\lambda_j \leq 0$ if $W_{Vj} = u_j$. Any value of λ_j is acceptable for $l_j = u_j$.

As NPSOL uses an SQP method one has two types of iterations when solving (3.13): *major* and *minor* iterations. The goal of the major iterations is to finally get to a point that satisfies the first-order optimality conditions. This is represented by

$$W_V{}' = W_V + \alpha p.$$

α is the step-length, a non-negative scalar, p is the search direction. p is again the solution to a quadratic optimization problem described as

$$\begin{aligned} &\text{minimize } E(W_V) + g(W_V)^T p + \frac{1}{2} p^T H p,\ p \in \mathbb{R}^V \\ &\text{with } l \leq c + J(W_V) p \leq u \end{aligned} \tag{3.14}$$

with H being a positive-definite quasi-Newton approximation to the Hessian of the Lagrangian.

In the case of FAUN the following can be noted for the Hessian of the Lagrangian

$$L(W_V) = E(W_V) - (\lambda_1, \ldots, \lambda_V) \begin{pmatrix} W_{V1} \\ \ldots \\ W_{Vn} \end{pmatrix}.$$

One gets

$$\frac{\partial L(W_V)}{\partial W_{Vj}} = \frac{\partial E(W_V)}{\partial W_{Vj}} - \lambda_j$$

and finally

$$\frac{\partial^2 L(W_V)}{\partial W_{Vj}\partial W_{Vk}} = \frac{\partial^2 E(W_V)}{\partial W_{Vj}\partial W_{Vk}}.$$

This means that the Hessian of the Lagrangian is actually the Hessian of $E(W_V)$. One could also determine the Hessian exactly instead of using an approximation. Automatic differentiation would be a viable alternative. If the computation of the Hessian is efficiently possible interior points methods are advantageous compared to SQP. When implementing other simpler training error functions it could be of interest to also try out this way although it would mean changing the current optimization algorithm NPSOL.

When solving (3.14) NPSOL uses the integrated package LSSOL, see [154] for more information on LSSOL. Major and minor iterations work like the following. First, the minor iterations provide a solution for p. Then, in the major iteration, one has to determine an appropriate step-length α. This means that the new iterate W_V' should decrease sufficiently when using an augmented Lagrangian merit function. Without nonlinear constraints the Lagrangian merit function is simply the objective function, see also [155]. In the last step one updates the approximate Hessian H to reflect the change from W_V to W_V'.

It remains the question how to find an initial acceptable point that satisfies the first-order optimality conditions. This is not difficult. As one has seen the vector of Lagrange multipliers is simply given by the gradient of $E(W_V)$. And as the only constraints one has are constant one is also able to find a point which satisfies the constraints.

The quasi-Newton updates during each major iteration are remarkably simple given the accommodating nature of the constraints. NPSOL requires a positive-definite approximation to the Hessian. Several alternatives can be used, see, e.g., [87]. FAUN uses BFGS quasi-Newton updates of the form

$$H' = H - \frac{1}{s^T H s} H s s^T H + \frac{1}{y^T s} y y^T,$$

setting $s = W_V' - W_V$ the amount of which W_V changes from iteration to iteration. If one starts with a positive definite H the following holds, see also [86]:

$$H' \text{ is positive definite } \Leftrightarrow y^T s > 0.$$

Without nonlinear constraints one has

$$\mathcal{Y} = \mathcal{G}(W_V') - \mathcal{G}(W_V),$$

i. e., the change in gradient between two iterations. If this doesn't lead to the required result one takes an appropriately scaled $\mathcal{Y}$.

3.7 Convergence Analysis

[182] proves that feed forward neural networks are universal approximators. More recently [282] also proves that recurrent neural networks can approximate — in principle — every dynamical system with arbitrary accuracy. This is done by viewing a recurrent network as feed forward neural network. Both proofs, however, do not tell how *exactly* one is supposed to find the weights that approximate the system of choice. Training is and always was heuristic.

For this reason the present section looks at empirical convergence behavior of the shared layer perceptron. One only analyzes training error, i.e., one ignores validation error momentarily. Then one trains for several iterations and observe convergence. The following meta parameters are of interest:

- number of observeables
- number of time steps
- state space dimension
- sparsity
- initialization domain
- random distribution of weights
- training of initial states
- part of the data selected.

The crucial point is whether training is always well-behaved in the sense that the error shrinks steadily. Or are there setups where convergence does not seem to occur?

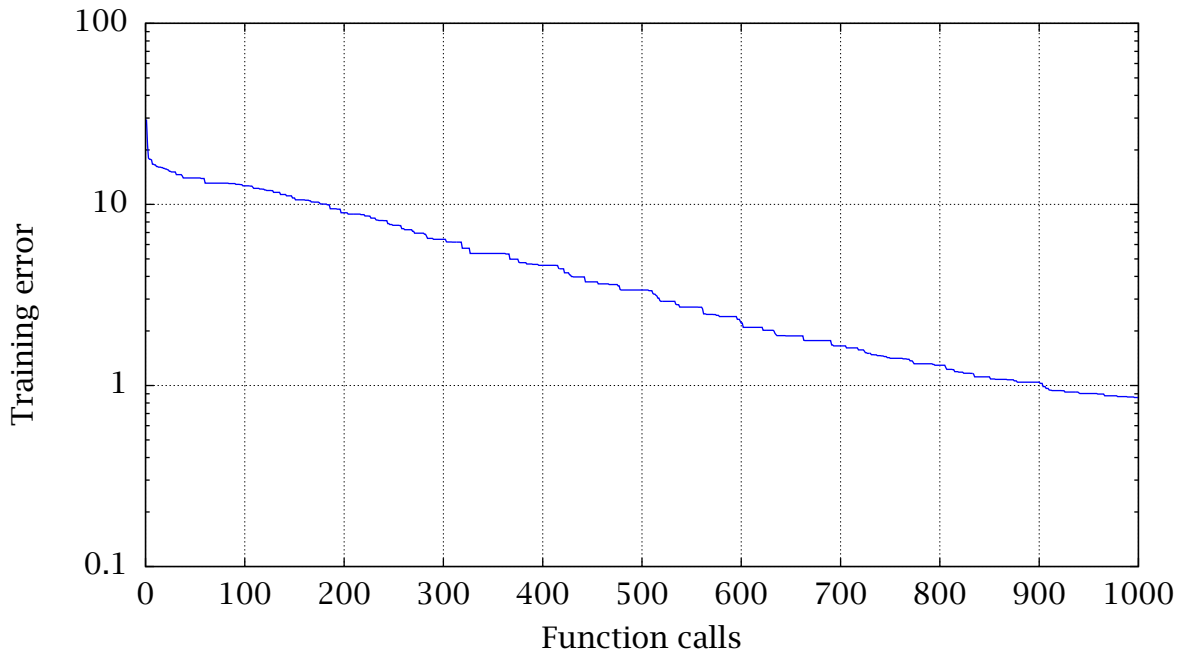


Figure 3.4: Typical convergence behavior: the initial error is mostly very high because one is on a random point in weight space. This is quickly corrected. Then the error decreases steadily. Each function call involves evaluating the network error and computing the corresponding partial derivatives. Function calls are not to be confused with optimization iterations. Each iteration may involve several function calls and not every function call necessarily yields an improvement. More than 95 percent of computation time is spent in function calls. Function calls are a good comparative measure.

Figure 3.4 shows typical convergence behavior. One sees that initial error is high. This is easily explainable by the fact that one generally starts on a random point in weight space. This point does not have to be and generally is not particularly good. The first few function calls quickly reduce the error. In the following the error decreases steadily. Every function call includes evaluating the error and computing the appropriate partial derivatives. Function calls are not equal to optimization iterations because the optimizer may need *several* function calls before finding an improved point. Function calls are the best measure in terms of computation time to compare different training processes. More than 95 percent of computation is spent in function calls and only a comparatively small amount in the optimizer.

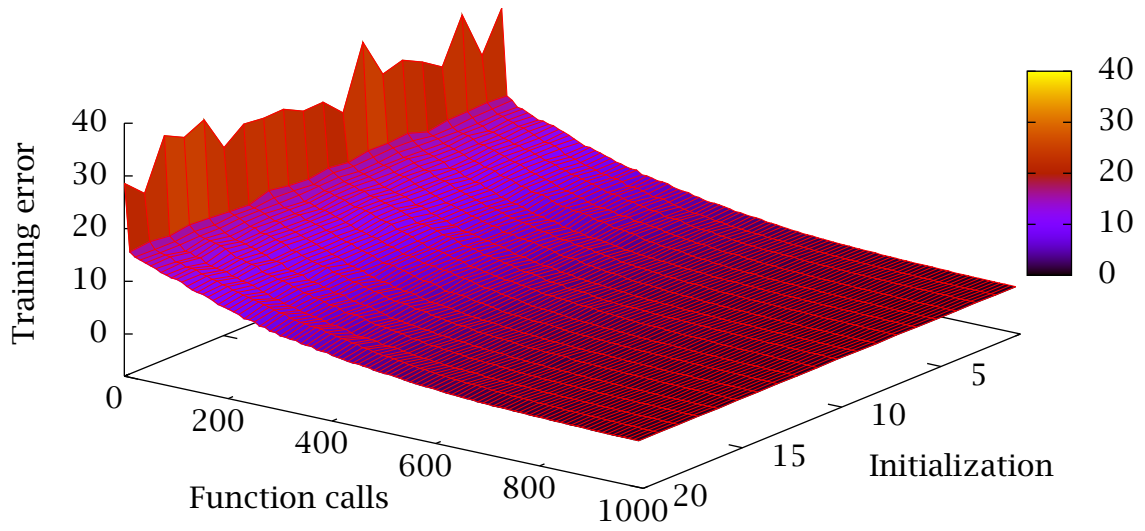


Figure 3.5: Convergence behavior for different random initializations. Varying initial errors are corrected quickly. This empirical analysis shows that differently initialized weight matrices do not influence convergence significantly.

Figure 3.5 shows the effect of different matrix initializations on convergence. All meta parameters stay the same. For every of the 20 runs different positions in the weight matrix are initialized with different values. One sees that initial errors vary. But after less than 50 function calls errors are comparable regardless of the individual initialization. This behavior is important. It makes us confident that convergence is not due to luck of finding a particularly good starting point in weight space. One sees here the same good natured convergence behavior as FAUN already shows for multi layer perceptrons.

Figure 3.6 on the next page presents convergence behavior depending on the number of observeables included in the optimization. Naturally, the initial error is greater when more observeables are considered. There is a mean error that results from taking a random point in weight space. These errors accumulate for more observeables. However, a detailed look at convergence behavior on figure 3.7 on page 91 proves very interesting. The figure shows only the last 100 function calls of 1000 function calls in total. One sees that the achievable training error actu-

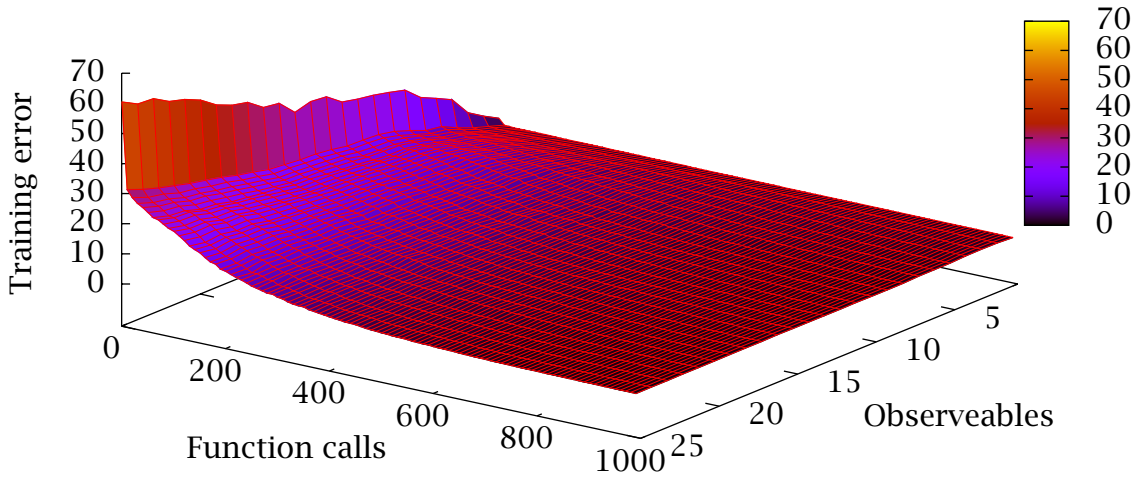


Figure 3.6: Convergence behavior for different numbers of observeables.

ally *decreases* when the number of observeables *increases*. This is very satisfactory because it gives us trust in the underlying modeling procedure. Adding more observeables which together form «the market» allows the optimizer to explain the behavior of some observeables by others. Looking at figure 3.7 in the context of chapter 4 shows that adding different asset types to a general market model does help to explain the *overall* development better. Especially note that the *cumulated* error is shown. One would already be satisfied if the error did not *increase* when adding more observeables. However, the best achievable error after 1000 function calls with 25 observeables is more than 50 percent smaller compared to the error achievable with few observeables.

Figure 3.8 on the next page shows the dependence of convergence on state space dimension. One sees that increasing the dimension of the state space also improves convergence. The reason for this is twofold. On the one hand increasing the state space dimension also increases the number of weights. However, as one sees on figures 3.10 and 3.11 this effect saturates. On the other hand one also increase the memory capability of the system. This also only leads to a noticeable improvement in training error up to a dimension of 300.

Next one addresses the question what happens when one increases the length

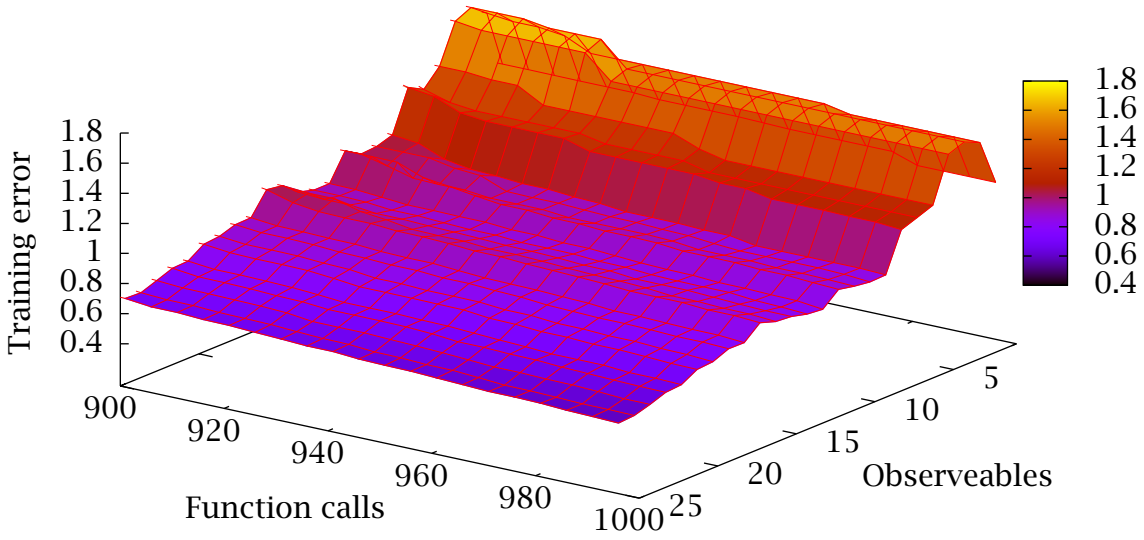


Figure 3.7: Convergence behavior for different numbers of observeables, detailed analysis.

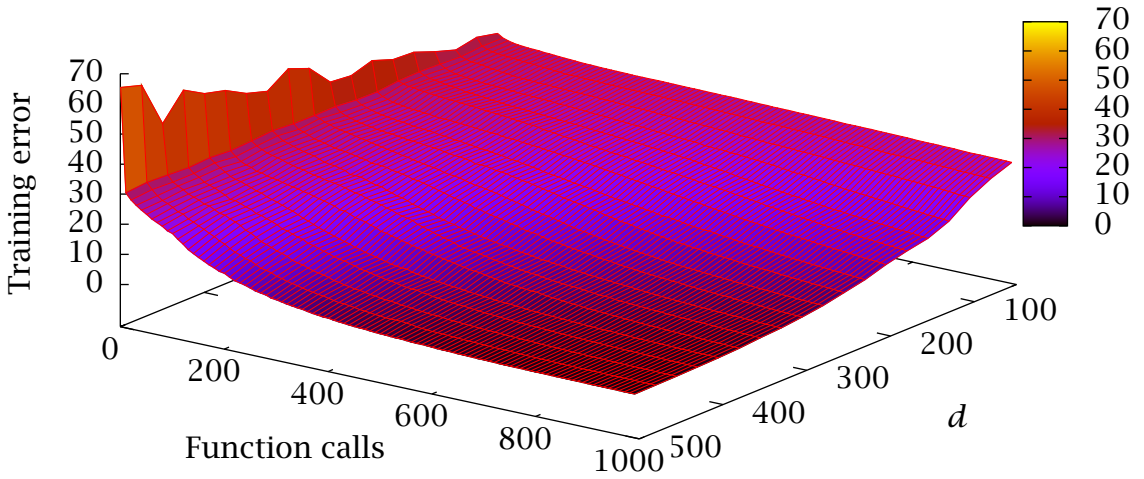


Figure 3.8: Convergence depending on state space dimension.

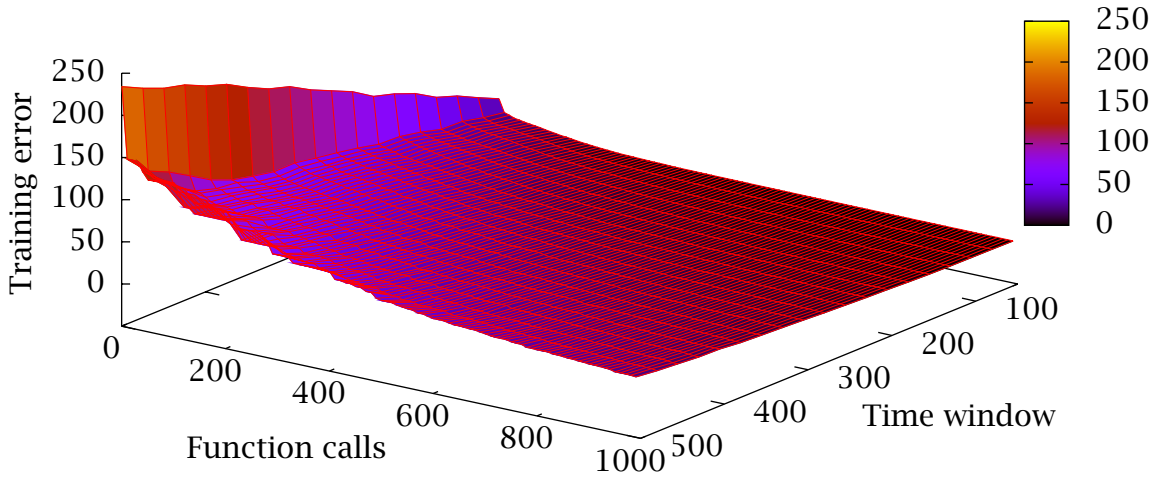


Figure 3.9: Convergence depending on the length of the time window included in the optimization.

of the optimization time window. I.e., one increases past history to learn. One sees on figure 3.9 that convergence is again well behaved. Evidently the initial error increases. However this error is reduced quickly and convergence is always smooth from that point on. As should be expected a *longer* history is more difficult to learn. The longer past history the less likely it is that actual dynamics as encoded in the weight matrix stay the same. Still the increase in training error is no obstacle for good forecasting results. Chapter 4 shows that the model is indeed robust even to very long time spans.

Figure 3.10 on the next page depicts convergence depending on sparsity. When one makes the matrix more dense one adds more weights. More weights will theoretically improve the best achievable training error. However, the same as with multi layer perceptrons, one sees that this only holds up to a certain point. Figure 3.11 on the facing page shows this in detail for the last 100 function calls. One sees that up to a sparsity level of 0.10 the achieved training error indeed decreases significantly. There is not much improvement when adding more weights.

One may also ask if one is perhaps especially lucky to get a time period which is easy to learn. For this reason figure 3.12 on page 94 shows a sweep with different

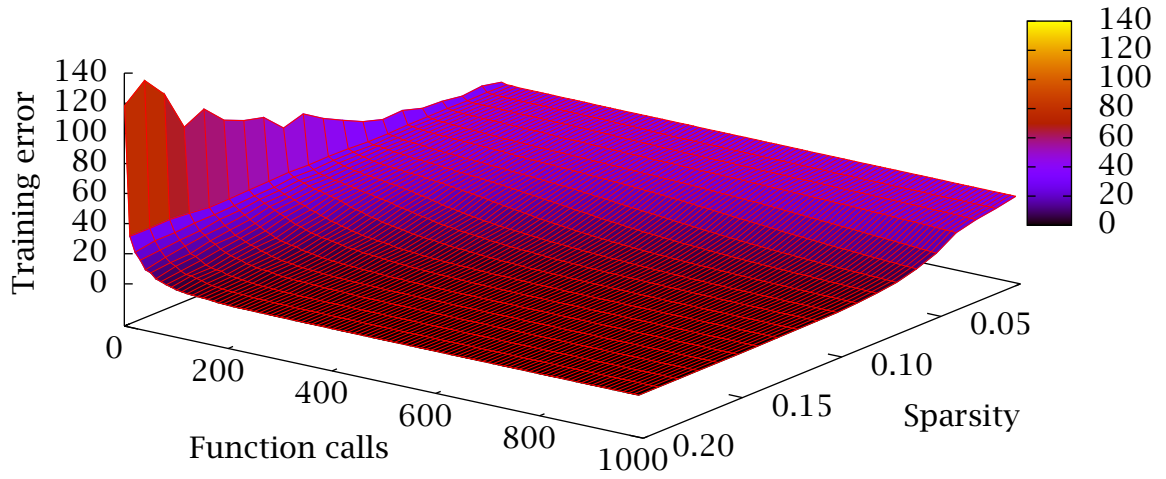


Figure 3.10: Convergence for different sparsity levels. Adding more weights by making the weight matrix more dense evidently helps but only up to a sparsity level of 0.10, see also figure 3.11.

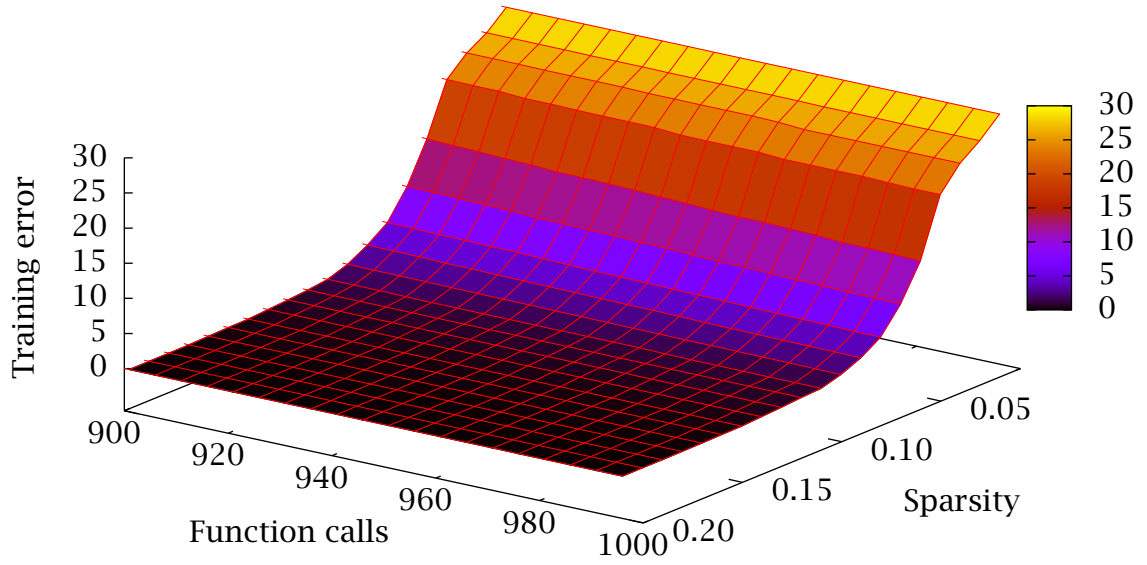


Figure 3.11: Detailed convergence for different sparsity levels and the last 100 function calls. One sees that saturation takes place.

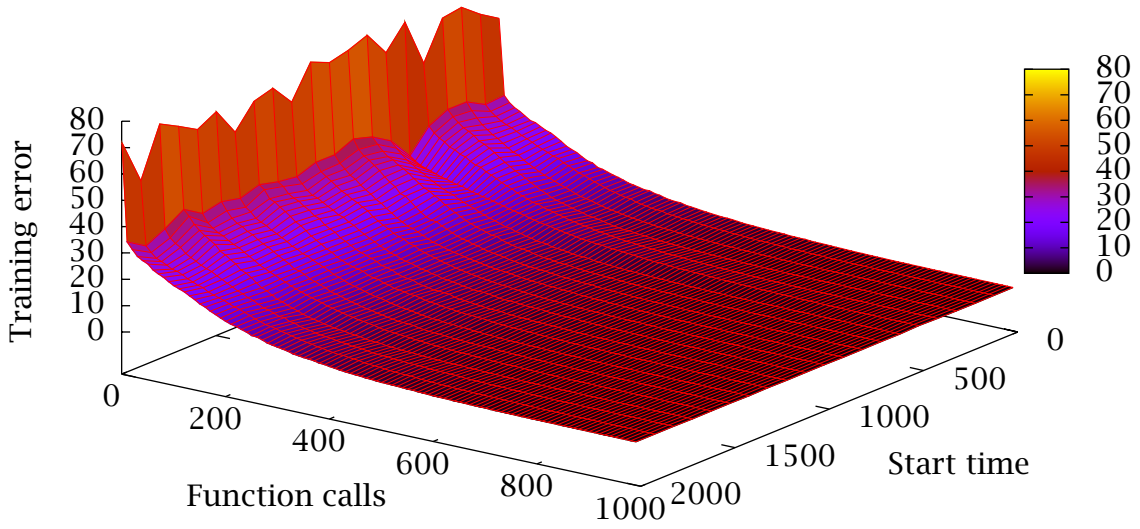


Figure 3.12: Convergence depending on the start time in history. One sees that during the first 300 function calls learning may progress at different speeds. However, this evens up and after 1000 function calls there is no noticeable difference in training error.

start times. There are indeed starting periods where initial convergence is faster than in other periods. This evens up after 300 function calls. The differences in training error after 1000 function calls are negligible. This finding is very satisfactory because it shows that the *same* weight matrix is able to encode different histories equally well.

At the beginning of every training the weights have to be initialized with a random start value. This is done by using a symmetric initialization interval around 0. Figure 3.13 on page 96 shows training errors when using different initial intervals. E.g. a value of 0.1 means that the weights are chosen from the interval $[-0.1; 0.1]$. During experiments the author faced a dilemma which the reader can also see in the figure. If one chooses the initialization interval too small all weights are very close to zero. One starts the search essentially from the zero matrix. While the training converges it is *very* slow. If on the other hand the initialization interval is too large the optimizer may get stuck at points in weight space where improvement simply is not possible. Fortunately this is detected and training stops. The reason

for this behavior is that when initializing on a large interval it may happen that one or a few weights dominate all the others. The optimizer then focuses on these weights and tries to change them. But this may happen in the *wrong* direction for the overall problem.

One notes that convergence occurs until an initial interval of $[-0.15; 0.15]$. Then there is — purely by chance — a last successful training for $[-0.17; 0.17]$. But other intervals above this do not lead to convergence. FAUN detects this automatically and stops the training. FAUN also offers the possibility to choose initial and runtime boxes for the weights. In the present case the author suggests to explore the limits and then choose a bound *well within* the secure area. As one sees in the figure convergence improves marginally for initial bounds between 0.1 and 0.15. But the author does not recommend these settings because one may inadvertently enter the domain of non convergence. As a general rule one should use *smaller* intervals if one has *more* weights. The rationale for this is that each individual output, i.e., each neuron, is the sum of its inputs at the previous time step.

A squashing through the nonlinear tanh occurs but nevertheless summing up is the essential feature. One wants to avoid driving the neuron into saturation. If a neuron is saturated it is useless for all practical purposes. If one changes one weight by a small amount which happens during optimization the neuron's output stays almost unchanged. To cause a change all weights would have to move in the *same* direction. These weights then lose their usefulness for the other outputs to be learned. A drastic method to investigate in further research would be to simply stop optimizing on these weights. This requires scaling down the number of active variables at runtime.

One sees that the shared layer perceptron converges well in all cases. When varying all available meta parameters over a sensible range convergence still occurs. The robustness of the model especially shows in the fact that adding more related observeables *decreases* overall training error. However, the choice of the initial bounds requires some care.

3.8 Summary

Key model features of the shared layer perceptron are:

- It forecasts *several* time series at once.

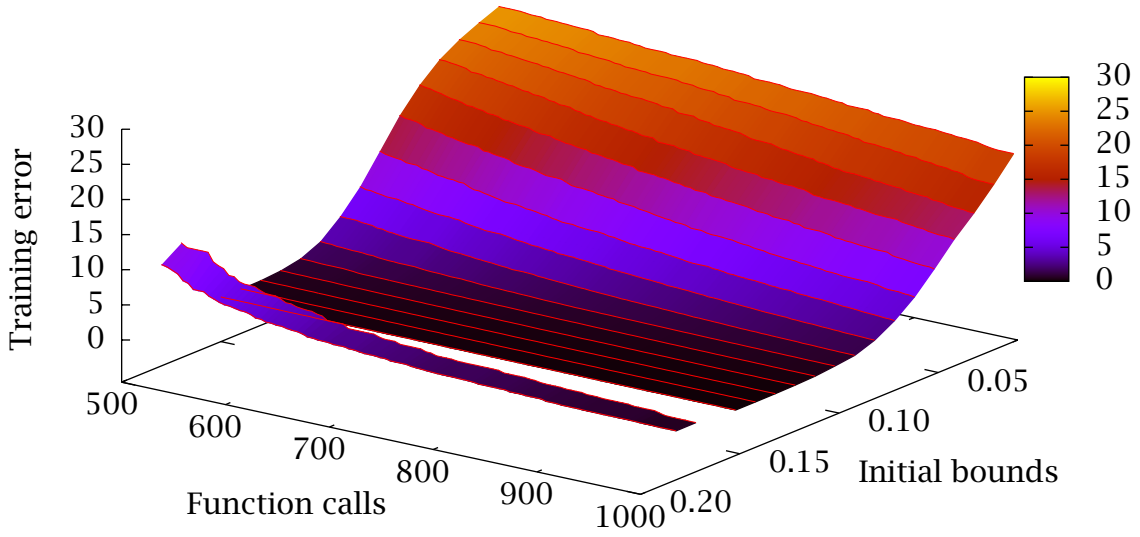


Figure 3.13: Convergence depending on the initial bounds of the weights. Up to 0.15 the training converges. Above 0.15 it is more often the rule than the exception that the optimizer is not able to improve the error and gives up after a few hundred function calls or even earlier. Here only the *last* 500 function calls are shown to preserve details.

- This allows to easily generate multi step forecasts.
- Time is treated symmetrically, i.e., one gets from past to future — and future to past — by simple matrix operations.
- Memory modeled by hidden states can be varied accordingly to the model builder's view: a shared layer perceptron with zero hidden states corresponds to a memoryless system while adding more hidden states allows for a practically infinite memory.
- A *single* matrix is used at every time step. This shared matrix approach reduces the number of weights.
- Sparsity level of the weight matrix allows to model loosely or densely coupled systems.

One sees that the shared layer perceptron is a very versatile architecture. It is designed to model dynamic systems, possibly with memory. By calculating partial derivatives with reverse accumulation one gets to overall computational requirements of $O(T \cdot D^2)$. I.e., computation time increases linearly with the number of time steps T involved in training and validation data and quadratically with the dimension of the state space. Using sparse matrices one may further save computation time.

Training error converges very robustly. The detailed convergence analysis shows that varying meta parameters does not lead to jumps in the achievable training error. Rather, transitions occur smoothly. The only parameter that necessitates some care is the initialization of the weights. One should not choose the initialization interval too large.

The shared layer perceptron topology may be further explored in the following directions:

- It is easy to deal with missing data by not accounting for the corresponding error during training. An interesting question is, if mixing data with different time frames improves forecasts. E.g., one could mix daily data with weekly data and simply ignore the error on weekly data for not available days.

- Now, the weight matrix is initialized randomly, weight value *and* position. This needs not be the case. One could imagine setting the *structure* of the matrix. This represents the model builder's view of causalities. Would this improve forecasts?

- By using all available data the shared layer perceptron is predestined for applications where only few data is available. How *few* is enough? In this book the author analyzes only daily data which is amply available. However, yearly panel data might only include 10–20 observations. Can the shared layer perceptron produce sensible forecasts even then?

- How much memory is adequate? Or: which is the optimal dimension of the state vector? It is of high interest to analyze different problems and produce a heuristic how to best set the state space dimension.

- Could dummy variables, like day of the week, improve forecasts? Of course, the model does not have to *forecast* them. They are simply included in the multi step forecast.

- Is it sensible to include a variable and some transformation of it in the same forecast? E.g., does adding a moving average improve the forecast?
- Can one forecast three interdependent variables and judge the quality of the forecast by recalculating the dependence? I.e., can one build an effective transitive filter?
- What does the distribution of *different* shared layer perceptrons tell us? Can one find a heuristic that allows us to judge the probability of the forecast?
- Can one interpret *single* weights? I.e., when looking at the weight values, is the result compatible with the view of the world?

All of these aspects should be analyzed in future work. The author thinks that every single question is very interesting. Answers will not only lead to a better understanding of the shared layer perceptron topology. They will also provide insights into the entire class of dynamic systems as well. However, every book must have a limit *somewhere* of what it can cover.

4 Examples

4.1 Introduction

In this chapter the shared layer perceptron is finally put to work. The focus shifts from *technical* and *mathematical* to practical applications on financial time series. To this end the author first reviews available resources on neural networks in finance, section 4.2. Then section 4.3 concentrates on the selection of appropriate data. From 31 financial time series one constructs 25 modified time series. This section also delivers some motivation for the inclusion of specific assets. Please keep in mind, though, that a definitive theory on input selection for neural networks has yet to be found. Therefore, the arguments given for selecting specific inputs are necessarily based on either general economic theory or heuristics. The portfolio comprises 2609 daily observations in a time span of 10 years from July 1999 to July 2009. This analysis, covering more than a complete business cycle, makes the present work stand out from other studies. Another feature is important: the first 330 days are used for training, the following 110 days for validation.

This merits a short discussion: one generally selects validation data to cover as much as possible *different* situations in the time series under consideration, see especially [43], p. 58 and p. 62. As one is dealing with financial time series it is, e.g., advisable to select a bull and a bear period and a sideways period. This is not necessarily fulfilled by choosing the *last* observations in the series. It is sensible to split validation data into several periods. However, as the present model applies to 25 different time series it would be necessary to select different appropriate validation data for *every* asset. This is doable. But it transforms the modeling process from a straightforward one into an arbitrary procedure. Think about it: everybody would probably select slightly different validation data. By selecting the most recent observations as validation data one gains two advantages:

- The model is validated on recent data and not on data in the past. Older data might not reflect current market dynamics.

- Validation data is the same for every asset. So every asset has the same odds of performing or not.

This procedure also opens room for further improvements. If one wanted to focus on one asset specifically one would check, if different validation data produces better performance.

Following applications then include an out of sample time span of 110 days and 2169 days, i.e., more than 8 years. Again, the analysis of the robustness of a model for this long time span is unique. Finally, let the author emphasize, that the entire portfolio of 25 assets is treated with the *same* model. While an individual optimization could lead to marginal increases in performance the attractiveness of the shared layer perceptron results from having one robust model which works well for all assets.

Section 4.5 presents basic descriptive statistics and necessary data transformations. One tests the data for normality and stationarity. Because of non stationarity of the raw level series one transforms the data into *rates of return*. This is often necessary when preprocessing data for neural network use, see [98]. The transformed series are stationary. They are suitable for neural network training.

The following three sections are entirely devoted to three unique examples. These examples, especially the first two, exploit the added possibilities by using shared layer perceptrons. The rationale for choosing these examples is based on practical applicability. If one is only interested in the potential of shared layer perceptrons one should go straight to the examples sections.

The first example deals with modeling value at risk. This well-known risk measure is designed to be indicative of the worst losses that may affect a portfolio within the next few days at a given confidence level. Standard estimation procedures, like historical simulation, often provide very conservative estimates. While this is good from the regulator's point of view it unnecessarily forces a financial institution to provide reserve capital. This results in lost opportunities when the capital could better have been invested elsewhere. The purpose of this example is *not* to replace standard value at risk procedures. They exist in their own right. Instead the idea is to model the worst expected return over a 10 day period as accurately as possible. The results are benchmarked against historical simulation.

The second example deals with a typical purchasing decision problem: on an ongoing basis some kind of asset has to be purchased. This might be a real commodity, like gold or oil, for industrial production. It can be a currency, that one

needs for regular foreign exchange transactions with business partners. Or, for a financial institution, it might be equity, or an equity index. Saving plans that regularly invest into assets to average in come into mind. In every case there is a need to buy within the next, say, 20 business days. But the exact day is not constrained. The challenge then is to buy at the *lowest* price within the time span under consideration. To achieve this the author uses multi step forecasts. The realized potential is benchmarked against a fixed day strategy where investments occur on a specific date within a four weeks time frame.

The third example is a standard problem for neural networks: accurately forecasting the sign for next day returns. In this domain numerous papers exist, see the next section. The unique feature of the present example is that it forecasts all 25 assets *at once*. There is no need to build models for every asset. This leads to a well performing, robust solution. Transaction costs are also taken into account. One enhances the neural network model by two filtered strategy. The first uses a threshold filter to avoid trades with low profit probability. The second also takes into account the forecast for the day after tomorrow and only trades on the trend. The neural network model is benchmarked against two technical trading strategies: the naive strategy and a moving average strategy.

The last section summarizes the results. It also offers an outlook on potential further research in the application of shared layer perceptrons to financial time series.

In the following, the author will among others look at currency exchange rates. As Tim Weithers from UBS so aptly puts it in [333]: «Foreign Exchange: It's not difficult; It's just confusing.» One of the many possible sources of confusion results from quotation conventions. The author use Weithers' suggestion and quote currencies as, e.g., GBP|USD 1.6240. The interpretation is quite easy with the simple rule: replace the sign «|» by «1» and add a «=» which gives GBP 1 = USD 1.6240. This means that for 1 Great British Pound one will get 1.6240 US Dollars. Historically quoting conventions haven been chosen so as to avoid having quotes starting with a zero. E.g., the Yen is quoted against the US Dollar as USD|JPY 92.83 to avoid an ugly JPY|USD = 0.01077 quotation. Another source of confusion results from unreflected designations like *home* and *foreign* currency. Depending on who and where you are, your concepts of home and foreign differ considerably. The author avoids these terms altogether and name the currencies involved. It is usually convenient to use three letter ISO codes to abbreviate currencies. These are unambiguous.

4.2 Literature Review

This section only deals with resources concerning the application of neural networks to financial time series. Technological aspects of parallelization and training and topology of neural networks in general have already been presented. Exploiting the potential power for forecasting financial time series is not a new idea. Since the early 1980s, where computational resources began to penetrate Wall Street, there have always been papers studying the forecasting accuracy of neural networks on financial time series. That is also the most common application. Others include volatility forecasts and — related to that — option pricing. To the best of the author's knowledge present literature does not take into account multi asset forecasts. Application of multi step forecasts is also very limited. Therefore the presentation of literature is quite straightforward by application class. Additionally the author also gives hints to non neural network references, where appropriate. This includes seminal articles on advances in financial time series analysis. References to aspects of quantitative investment and trading are also included.

Forecasting returns

Equity A classic application is presented in [58, 128, 213, 227, 258, 260, 279, 290, 303, 315, 316, 318, 319, 348]. [194] provides a long-term analysis of stock markets. A similar but shorter term analysis is shown in [196]. [8] provides a novel contribution by augmenting the neural network decision by fuzzy logic rules. These especially allow to avoid trades with low success probability. [22] applies recurrent neural networks based on volatility to stock indices. It is therefore a mixed approach. [49, 50] give evidence for inefficiencies in FTSE futures. [61] uses self correcting networks. In [62] the same authors exploit inefficiencies in the Taiwanese stock market. [73] augments neural network forecasts by a regime switching component and applies it to the Cyprus stock market. [76] analyzes technical trading rules for a 12 year period on the Hang Seng Index. [94] adds a GARCH model, [109] an ARMA model. In [162] one will find a somewhat dated but still pertinent compilation of the challenges in forecasting equity. [317] develops a methodology for neural network models in stock markets.

Interest rates [43, 246] provide studies concerning yield curves, here the German yield curve. Neural networks are trained with FAUN. [115] models the UK 10

year bond. [190] predicts inflation with interest rate spreads. [364] forecasts yield curves with error correction networks.

Exchange Rates [91,99,110,112,118,136,137,184–187,203,206,210,223,225,288, 309,336,345,346,349–352,355,363] present classic studies. [324,326,327] use FAUN. [66] presents a recent study focused on high-frequency exchange rates. [28,143] analyzes the potential of carry trades. [6] uses genetic algorithms to improve training times and generalization capability. A similar evolutionary approach is presented in [40]. [68, 354] use a wavelet approach. A general comparison is found in [85,245]. [275] investigate intra day arbitrage. In [322] a model is developed to trade option on the Dow Jones.

Commodities [104, 105, 108] trade commodity spreads. [106, 107, 343] additionally apply filters. [116, 226, 242] focus on precious metals. [25] presents an early application of a hybrid approach combining neural networks and expert systems. [360] applies historically correct neural networks.

A general direction of change analysis is carried out in [23]. Pure neural networks are compared to neural networks with a fuzzy component. [26] presents neural network models for different assets, unfortunately without enough details to reproduce the results. [253] chooses a slightly different approach with reinforcement learning.

Most studies show, that neural networks are able to reduce forecast errors compared to standard models. However, only few studies provide a thorough evaluation of returns when transaction costs are applied. Often, standard risk measures are not provided. If financial measures are given, they are often limited to annualized return *before* transaction costs. Indeed, from an investment point of view the *error* of fit is almost irrelevant. It says nothing about the performance of a strategy in a real environment. Even returns without risk measures are almost worthless: what good is a paper profit of 50 percent when the model incurs a drawdown of 90 percent? In this case return might be due to learning a single *unpredictable* event, see [200] for a warning on this topic. It may be that nothing of this kind happens but without additional information *one simply cannot tell!* The reason the author emphasizes this problem is that there are examples for very thorough studies. All necessary trading and risk measures are given and results are of practical use for quantitative investment. Not the only but the most consistent author of this kind is Christian Dunis from Liverpool John Moores University. Following analysis will be based on his methodology.

Forecasting volatility and risk

Equity [129] analyzes volatility in S&P500 options. [20] use a volatility mixture to measure risk in the DAX. [113] actually *trades* volatility. [247] present a hybrid model with GARCH.

Interest rates [127] prices options on yield curves.

Exchange rates [195] forecasts volatility. [100, 217] focus on risk management while [101] emphasizes trading aspects. [146, 313] use support vector machines. The study [302] stands out by analyzing tick data.

Commodities [111] models VaR.

[59] provides a VaR analysis. [198] more generally introduces volatility options. [273] deals with model risk. [289, 306] deal with aspects of trading volatility.

Option pricing

[16–18, 285] use FAUN within the WARRANT-PRO-1 software to price options with empirically mined web data. See also [45, 323, 328] on the WARRANT-PRO family. [41] uses specialized neural network hardware to efficiently compute neural networks for option pricing.

A standard application is presented in [269, 284]. [7] prices options using a combination of neural network *and* parametric models. Similar in spirit [214, 268] provides another hybrid approach. [123] focuses on market making. [144, 150] focus on *hedging* options. Model risk is considered in [149, 215].

As general introductions to option pricing the author recommends [32, 188, 340]. [145] deals specifically with foreign exchange options. See also [52] for the underlying concept of implied volatility. Market efficiency is considered in [65].

Modeling aspects

An introduction to forecasting applications is given in [77]. [244] specializes on finance applications. [78, 79, 102, 170, 172, 220, 256, 262, 347, 356, 357, 361, 362] give more details. A collection of modeling papers with a strong focus on neural networks is [98]. An introduction to historically correct neural networks is given in [359]. In [166] a wide variety of neural networks is analyzed. [219, 337] focus on decision support at financial services providers. [189] adds stochastic differential equations to the model.

[1] discusses the aspect how to calibrate a neural network model to financial data without overfitting. The 2009 paper [14] provides several experiments on how to best *train* neural networks for nonlinear time series. [15] goes further and proposes an automatic modeling procedure, but only for univariate time series. [97] focuses especially on genetic algorithms. [47] illustrates diverse aspects of using neural networks, and especially FAUN, in the context of decision support at financial services providers. The problem of missing data is addressed in [103]. A simpler yet effective class of neural networks, higher-order neural networks, is presented in [141]. Another class of networks, Gaussian mixture models focusing on distributions, is discussed in [224].

As a general comparison, [197] provides a good overview between linear and nonlinear models. A similar analysis is carried out in [57, 82, 90, 117, 239, 241]. [64] focuses especially on neural networks. [34] specifically analyzes regime-switching models. This is also the topic of [36] which focuses on foreign exchange rates. [11–13] analyze statistical aspects of foreign exchange markets in the years 1989–1992. They focus on daily and intra-day returns and represent early work on market microstructure. The question of how forecasts are best combined is treated in [69, 92, 93, 95, 308]. Filters are introduced in [114]. A good introduction to modeling is also [142]. The information theoretic point of view is discussed in [235]. As a toolbox for nonlinear modelers [286, 287, 291, 298, 314, 365, 366] are helpful. [27] highlights valuation factors.

Everyone dealing with financial time series should read the original paper on GARCH [37]. Another historic reference is the original paper on «the behavior of stock-market prices» [126]. [301] provides a general introduction to modeling. [38, 233, 237] make for an entertaining reading on the dangers of *too much* quantitative modeling. Neural networks are not explicitly mentioned. On the other hand, [56] encourages one to implement a quantitative point of view and also mentions neural networks in a practical context. A similar goal is reached by [75, 96, 243, 296] with a focus on long *and* short strategies. The special aspects of high-frequency strategies are discussed in [80, 151, 216, 218]. Technical strategies are introduced in [119, 199, 200, 254, 255, 270, 332]. Their potential is further analyzed in [10, 147, 148, 159, 202, 300, 320]. Finally, don't miss out the classic [161] for an *opposite* and yet similar point of view.

4.3 Data

4.3.1 Data Selection

As a definitive theory explaining the interconnection between currency exchange rates, stock markets, interest rates, and commodities is still missing, it is plausible to include in forecasting and risk management applications, data that is *a priori* likely to influence the target of interest. A univariate analysis is generally said to be inferior to a multivariate analysis. Therefore the author selects data in accordance to what economic theory would suggest. It is beyond the scope of this book to provide a unified theory of currency, fixed income and equity markets. The following arguments should be seen as *motivation* why specific datasets are included in further analysis. The author refers the reader to the relevant literature for detailed theoretical arguments. The author's analysis is purely experimental in the sense that the author assumes very little a priori interdependencies and do not assume a specific model.

It is, e.g., common understanding that there is a linkage between currency exchange rates and interest rates. The direction is generally seen as interest rate influencing the exchange rate. I.e., higher interest rates make a currency more attractive and therefore more investors will tend to buy this currency. The currency will *appreciate*. However, this relation needs not be so clear. Above average interest rates may also signal inflation. In this case the corresponding currency appears less attractive. For the five major currencies in stable economies, generally, the attractive argument holds.

Can exchange rates also influence interest rates? Certainly, but the argumentative chain is a bit convoluted. If an exchange rate is not appropriate in the eyes of the respective central bank there might be an intervention to alter the exchange rate. One instrument of a central bank is to set the reserve interest rate. This will typically also influence interbank interest rates. Because of the importance of interest rates and exchange rates not only for financial markets but especially for the *real* economy the author decides to include a wide range of exchange rates and benchmark interest rates in the dataset.

The linkage between equity markets and interest rates is popular. The general assumption is that when interest rates go down, equities rise and vice versa. This is motivated by the fact that low interest rates generally provide cheaper access to capital for companies. This stimulates new investments and innovation. Cheap

funds also stimulate by inciting consumers to take on, e.g., mortgages.

On the other hand, high interest rates are a hindrance for most companies. However, the author suggests that such over-simplifying assumptions should be seen with circumspection. There are situations where the room for maneuver for central banks is quite limited. As the time of writing the Federal Reserve has a target rate between 0.0 and 0.25 percent. The effect on the economy is, at best, mixed and some economists argue that the rate should actually be negative. This is not possible, of course. Nevertheless, it seems reasonable to include the most important equity markets in subsequent analysis.

Finally, most research agrees on a link between currency exchange rates and equities. The direction of the link is however not always the same. Generally, an export oriented country will profit when its own currency *depreciates*. This makes the exported products relatively cheaper for others to buy. Shares of export oriented companies will therefore tend to increase in price. Equity is often seen as hedge against exchange rate fluctuations: The share gives right to a *part* of a company. If the assets themselves do not change value for some fundamental reason the share price should increase when the home currency depreciates. Yet, the effect on the economy can be mixed. Imported products without local substitute become more expensive. This can increase inflation which is perceived as bad. Again, the author decides to include the most important equity indices in the dataset.

Often, a fourth asset class is considered when describing economic links: commodities. This is, however, a very generic term. It encompasses at least

- precious metals: silver, gold, platinum and palladium
- other industrially used metals: e.g., copper or aluminum
- energy products: the most notable being oil, especially Brent and WTI crude
- agricultural products: e.g., corn, wheat, soybeans and sugar and
- livestock: e.g., cattle.

From these commodities two generally receive much attention in research and popular media: gold and oil. Gold is, even today, accepted as reserve currency and hedge against inflation. Oil, on the other hand, is a very important consumable in developed countries and also has political importance. The author deems it inadequate, however, to include every other commodity in subsequent analyzes because this would unnecessarily inflate the dataset.

Because of this, the author selects a commodity index for inclusion, see below. Charter rates are also important. They reflect the actual costs of transportation for certain commodities but are also considered a leading indicator, see below. Again, the link of commodity prices to other parts of the economy can work in both directions. Oil and gold, quoted in USD, will exhibit an inverse relationship. The connection to other currencies is still unclear. Rising oil and gold prices will, on the other hand, tend to benefit oil and gold producers. Especially the oil companies are often among the largest companies by capitalization of a country. They will tend to bias the relevant equity indices upwards. Other companies, which employ oil or any other commodity as input factor will face a lower share price. To summarize: the link between equities and commodities is one of the areas where further research is especially needed.

4.3.2 Data Description

The author selects the following data pool for subsequent analysis. The author bases his selection on methodology from [118]. The data is also summarized in table 4.1 on the next page. An overview of the exchanges is presented in table 4.2 on page 118. Additionally the author gives a reason for choosing this specific time series. The data in the following description is ordered by *world regions* whereas the data in the tables is ordered by *instrument type*. This is due to the fact that general explanations on selecting specific datasets are more relevant to *regions*. Furthermore, this order also corresponds to a typical trading day which begins in Asia Pacific moves to Europe and closes in the United States. But subsequent analysis will show that time series offer greater correlation, and often predictive power, when one considers related *instruments*. The attribution of a specific time series to a world region is in some cases somewhat ambiguous: an exchange rate, e.g., Yen to US Dollar, belongs in fact to two different regions. But as this reference rate is mostly stated to compare different economies vis-à-vis the US Dollar the author decides to put it in the Asia Pacific part. The same applies respectively for the other exchange rates. Some time series are not attributable to *specific* regions. They are referenced at the end in the *World* category. The author visualizes the data on a world map on figure 4.1 on page 110. This figure illustrates clearly the timezone considerations when using time series from different world markets.

Name	Instrument	Region	Datastream
FTSE 100 Index	Equities	United Kingdom	FTSE100
DAX 30 Index	Equities	Germany	DAXINDX
CAC 40 Index	Equities	France	FRCAC40
FTSE MIB	Equities	Italy	FTSEMIB
Dow Jones Euro Stoxx 50	Equities	Europe	DJES50I
S&P 500 Index	Equities	United States	S&PCOMP
NASDAQ 100 Index	Equities	United States	NASA100
Nikkei 225 Index	Equities	Japan	JAPDOWA
Kospi Index	Equities	South Korea	KORCOMP
3 months LIBOR	Interest rate	United Kingdom	ECUK£3M
12 months LIBOR	Interest rate	United Kingdom	BBGBP12
Germany 3 months	Interest rate	Germany	ECWGM3M
France 3 months	Interest rate	France	ECFFR3M
Italy 3 months	Interest rate	Italy	ECITL3M
EURIBOR 3 months	Interest rate	Euro area	ECEUR3M
Eurodollars 3 months	Interest rate	United States (Europe)	ECUS$3M
Benchmark Bond 3 months	Interest rate	Japan	ECJAP3M
Benchmark Bond 10 years	Interest rate	United Kingdom	UKMBRYD
Bund Future 10 years	Interest rate	Germany	BDBRYLD
Benchmark Bond 10 years	Interest rate	France	FRBRYLD
Benchmark Bond 10 years	Interest rate	Italy	IBRYLD
US Treasuries 10 years	Interest rate	United States	USBD10Y
Benchmark Bond 10 years	Interest rate	Japan	JPBRYLD
US Dollar to Great British Pound	Exchange rate	United Kingdom	USDOLLR
US Dollar to Swiss Franc	Exchange rate	Switzerland	SWISFUS
US Dollar to Euro	Exchange rate	Euro area	USEURSP
Yen to US Dollar	Exchange rate	Japan	JAPAYE$
Gold Bullion	Commodity	United Kingdom (world)	GOLDBLN
Brent Crude Oil	Commodity	Europe (world)	OILBREN
CRB Index	Commodities	United States (world)	NYFECRB
Baltic Exchange Dry Index	Commodities	world	BALTICF

Table 4.1: The time series used in the following examples, ordered by instrument type. The given region indicates not necessarily where the instrument is quoted but rather that it's important and widely followed in the given area. Refer to the text for further explanation and especially time zone considerations.

Figure 4.1: Schematic representation of data used for subsequent analysis, ordered roughly geographically by *source*. Note that one has to be careful when considering which lags to use: the trading day begins in Asia Pacific in the East, goes to Europe and ends in North America. At the end of trading in North America Asia Pacific again takes over with a *new* trading day. Consequently, when using time series for forecasting only data which comes from a market place *East* of the series to forecast can be taken into account for the same day. As a rule of thumb Asia Pacific data is available at 11.00 GMT, European data at 17.00 GMT and North America data at 22.00 GMT. Thus, from a European perspective North America data arrives *too late* to be traded upon. See the text for exact quotation times. Source: overlay own creation, background world map from Wikipedia released into the public domain.

Asia Pacific

Nikkei 225 Scope: Japan, Quotation: Tokyo Stock Exchange, 8.45 GMT. This index comprises the 225 largest companies in Japan. See [70], p. 1 for a very brief introduction. It is calculated by the Nihon Keizai Shimbun newspaper and published in real time.

Data from Asia Pacific is generally already available in the European morning session because of the considerable time zone lead. It is important to include especially Japanese data as Japan is the second biggest economy of the

world and generally representative for the Asia Pacific region. (Datastream JAPDOWA)

Kospi Index Scope: South Korea, Quotation: Korea Exchange, 10.45 GMT. The Korea composite stock price index comprises all companies listed on the Korea Exchange, see [209], p. 8. It is published in real time. Closing data is already available in the morning session. The South Korean economy is also considered as a leading indicator for the world economy, see [152], p. 328 and [171] for a more general introduction on leading indicators.

This is due to the fact that South Korea is an important manufacturer and exporter of electronic devices and the largest shipbuilder in the world. It's capital, Seoul, ranks among the ten most important financial capitals. The rationale for the acceptance as leading indicator is, that, in the long term, a weakening world economy tends to announce itself by reducing demand of goods and therefore a reducing demand of *shipping* goods. This effect is felt first by ship manufacturers. See also the explanation for the Baltic Exchange Dry Index below. (Datastream KORCOMP)

Benchmark Bond 3 months Scope: Japan, Quotation: Collected by the European Central Bank and provided by Reuters, 21.50 GMT. This bond reflects the short term interest situation in Japan. On the one hand it is indicative, with an additive premium, for the rate at which Japanese companies may borrow funds. On the other hand it also indicates how attractive short term yen investments are. Until the period 2007–08 Japanese carry trades were very attractive. Carry traders also influenced the exchange rate, see [143]. (Datastream ECJAP3M)

Benchmark Bond 10 years Scope: Japan, Quotation: Collected by the European Central Bank and provided by Reuters, 21.50 GMT. As for the 3 months benchmark bond the 10 year bond allows an outlook on the Japanese interest rates, but for the *long-term*. Again, one assumes that the inclusion of Japanese interest rate data adds valuable forecasting power as Japanese time series move more independently as the following analysis shows. (Datastream JPBRYLD)

Yen to US Dollar Scope: Japan, Quotation: Reuters, 21.50 GMT. The exchange rate between Japan and the United States is interesting for several reasons: The United States and Japan are the two biggest *national* economies of the world and therefore the interaction of their exchange rate bears significant economic impact. Additionally, the Japanese economy has had a tendency to develop

relatively independently from other economies. Finally, due to very low interest rates in Japan carry trades have flourished. And when carry trades are reversed this influences the exchange rate significantly, see also the comments above. (Datastream JAPAYE$)

Europe

FTSE 100 Index Scope: United Kingdom, Quotation: London Stock Exchange, 16.35 GMT. This index represents the 100 largest companies in the United Kingdom, see [173]. It is published in real-time by the FTSE Group, originally a joint venture of the Financial Times and the London Stock Exchange. The author uses the closing price as realized in the closing auction between 16.30 and 16.35 GMT. (Datastream FTSE100)

DAX 30 Index Scope: Germany, Quotation: Deutsche Börse, Xetra, 16.35 GMT. This index consists of the 30 largest companies in Germany, see [88]. It is published in real time by Deutsche Börse until close of trading in Frankfurt. An index update is available as X-DAX until 21.00 GMT but less liquid. The published value is the result of the closing auction between 16.30 and 16.35 GMT, see [89]. (Datastream DAXINDX)

CAC 40 Index Scope: France, Quotation: Euronext Paris, 18.30 GMT. The index consists of the 40 largest companies in France, either traded on the Bourse de Paris *or* with significant influence on the French market, see [259]. It is published in realtime by the Euronext division of the New York Stock Exchange. (Datastream FRCAC40)

FTSE MIB Scope: Italy, Quotation: Borsa Italiana, 19.30 GMT. This index consists of the 40 largest companies in Italy, see [140]. It is published in realtime by the FTSE group. (Datastrem FTSEMIB)

Dow Jones Euro Stoxx Scope: Euro area, Quotation: derived from Dow Jones Stoxx 600 Index, 8.00 GMT. This index consists of the 50 largest companies in the Euro area. It is updated in real-time depending on local trading hours, see [299]. The time series available in Datastream reflects *opening* or *pre-opening* prices in the relevant local markets. (Datastream DJES50I)

3 months LIBOR Scope: United Kingdom, Quotation: British Bankers Association, 11.00 GMT. The LIBOR interest rate series is published daily by the British Bankers' Association by taking the quotes of participating banks, see [21]. To

avoid a distortion or influence by interested parties the top and bottom quartiles of quotes are discarded before calculating the average. LIBOR reflects the price at which banks with good credit *could* borrow GBP from each other. It is a good proxy for the *risk free rate*. During the credit crisis of 2007–2009, however, LIBOR surged. But even at the higher rates banks were not willing to lend. Nevertheless, LIBOR is generally widely followed and amply commented on in the financial press. (Datastream ECUK£3M)

12 months LIBOR Scope: United Kingdom, Quotation: British Bankers Association, 11.00 GMT. The same as for 3 months LIBOR, see above, applies. The 12 months rate is equally widely followed. (Datastream BBGBP12)

3 months Benchmark Bond Scope: Germany, Quotation: collected by the European Central Bank and distributed by Reuters, 21.50 GMT. The data series is a synthetic benchmark for a 3 months debt from the German government. The rationale for including data from different European countries is that although the rating of European governments is generally the best, AAA, the *perceived* credit worthiness varies. One can assume that to include yields from different European countries offers predictive power compared to a *single* synthetic rate. (Datastream ECWGM3M)

3 months Benchmark Bond Scope: France, Quotation: collected by the European Central Bank and distributed by Reuters, 21.50 GMT. See explanation for the German benchmark bond above. Again, the French economy is slightly different but generally similar compared to the German. A model should be able to use this information. (Datastream ECFFR3M)

3 months Benchmark Bond Scope: Italy, Quotation: collected by the European Central and distributed by Reuters, 21.50 GMT. Compared to German bonds Italian bonds often trade at a discount, i.e., with higher implied yields. This can especially be seen when looking at the time series of the years 2007–09. Economic data and especially the Italian budget deficit seem to enhance the probability of default in the eyes of potential investors. Whether this perception is accurate or not shall not be discussed here. The point is that the difference between German and, especially, Italian government bond yields could be exploited by a model. (Datastream ECITL3M)

3 months EURIBOR Scope: Euro area, Quotation: British Bankers Association, 11.00 GMT. The European Interbank Offered Rate is published daily by the British Bankers' Association. Like LIBOR it is based on the quotes of participating

banks. The reason of including EURIBOR in addition to the 3 months European government yields is that floating interest rate products are often based on 3 months EURIBOR. It is, e.g., common for corporate bonds to offer a floating rate of 3 months EURIBOR plus a premium. This rate is equally widely followed and reported in the mainstream financial media.

A second reason for including this time series is that the spread between an interbank offered rate and the corresponding same maturity government rate can be quite informative on the actual state of the economy. Generally, the spread is small because the creditworthiness of important banks is perceived almost as good as the creditworthiness of a European government. However, in crisis times, like, e.g., the years 2007–09, the spread widens considerably. This is due to a lack of trust among banks. At the height of the credit crisis in 2008 the banks regularly feared that their counterparty could unexpectedly go bankrupt and that the posted collateral would also be worth nothing.

The yield spread could therefore work as leading indicator. As an aside: It is a common misconception that LIBOR and EURIBOR are fundamentally different. Actually, EURIBOR is a kind of LIBOR because the *L* in LIBOR stands for *London* and both rates are indeed quoted in London. By choosing the names LIBOR and EURIBOR the British Bankers Association actually introduced euphonious marketing names — but also a potential source of confusion. (Datastream ECEUR3M)

UK gilts 10 years Scope: United Kingdom, Quotation: reference quotation by the Bank of England and the Government's Debt Management Office, 16.00 GMT. The gilts, i.e., the United Kingdoms government bonds, are issued by the Government's Debt Management Office and can also be bought there directly, see [311]. They are also traded on the stock exchange. These benchmark bonds are widely followed and trade liquidly. (Datastream UKMBRYD)

Bund Future 10 years Scope: Germany, Quotation: Eurex, 21.00 GMT. The Bund Future is a derivative based on the delivery of an imaginary German government debt obligation, Bundesanleihe, with maturity between 8.5 and 10.5 years, see [124]. Bund futures are highly liquid instruments and a benchmark for expected long term Euro interest rate. (Datastream BDBRYLD)

10 years Obligation assimilable du Trésor Scope: France, Quotation: benchmark quotation by Agence France Trésor, 16.00 GMT. The *obligations* are issued by the French government via Agence France Trésor a department of the French

finance ministry, see [3]. They can be traded in the secondary market at Euronext Paris and are among the most liquidly traded bonds in France. (Datastream FRBRYLD)

10 years Buoni del Tesoro Polianuali Scope: Italy, Quotation: benchmark quotation by the Dipartimento del Tesoro, 16.00 GMT. The *buoni* are the long term government bonds issued by Italy. (Datastream IBRYLD)

US Dollar to Great British Pound Scope: United Kingdom, Quotation: Reuters, 16.00 GMT. While the GBP|USD exchange rate is traded around the clock in most financial centers the London fixing is a widely followed and published price. GBP|USD is among the five most actively traded currency pairs and is therefore included for subsequent analysis. (Datastream USDOLLR)

US Dollar to Swiss Franc Scope: Switzerland, Quotation: Reuters, 16.00 GMT. The same technical details as above apply. The rationale for including the Swiss Franc is that the market often considers it as a *safe heaven* because of Switzerland's *neutrality*. Although trading volume is much smaller compared to EUR|USD it is nevertheless interesting to see if changes in SFR|USD have predictive power for other time series. (Datastream SWISFUS)

US Dollar to Euro This is the most actively traded exchange rate of the world. Numerous studies have investigated the behavior of this exchange rate and it is generally accepted that the exchange rate market is very efficient. Because of the significant economic and technical impact of EUR|USD it is included. (Datastream USEURSP)

North America

S&P 500 Scope: United States, Quotation: Standard & Poors, 0.30 GMT. This index comprises the 500 largest companies in the United states, see [295]. The companies are either traded on New York Stock Exchange or NASDAQ OMX. The index is published in real-time. The price available in Datastream reflects the closing price of after-hours trading. Keep in mind that from a European perspective the value is only available *after* the European market close. To avoid a look-ahead bias it is necessary to take the price from the *previous* day. (Datastream S&PCOMP)

NASDAQ 100 Scope: United States, Quotation: NASDAQ OMX, 1.00 GMT. This index comprises the largest 100 companies on the National Association of Se-

curities Dealers Automated Quotations system, see [9], p. 10. The companies are US but also foreign companies and, importantly, *non-financial*. The widely followed NASDAQ 100 index is mostly known as technology index. There is an intersection of S&P 500 and NASDAQ 100 companies. Nevertheless, it is interesting to include the NASDAQ 100 in its own right and analyze, if the predictive power of subsequent models is enhanced. The index is published in realtime. The same time zone consideration as for the S&P 500 applies when dealing with mostly European data. (Datastream NASA100)

3 months Eurodollars Scope: United States, Quotation: Reuters, 21.50 GMT. This somewhat misleading term refers to US Dollars deposited in bank accounts *outside* the United States, see [333]. These US Dollars are therefore not subject to different restrictions imposed by the Federal Reserve and the 3 months interest rate payed on the deposits is assumed to better represent current market conditions.

Nevertheless, Eurodollars will also react to the Federal Reserve policy because the target rate determines, among others, the attractiveness of the currency. Although most Eurodollars are actually deposited in *European* banks there is no connection at all to the Euro or even to Europe. US Dollars held, e.g., in Tokyo would technically be called Eurodollars, too. Eurodollars can be traded at the Chicago Mercantile Exchange using futures, see [71]. As an aside see [139] for a historical and thought-provoking introduction to Eurodollars. (Datastream ECUS$3M)

10 years Treasuries Scope: United States, Quotation: Reuters, 21.50 GMT. The 10 years treasury notes is the benchmark bond for long term interest rate perception in the United States, see [212]. They are auctioned by the United States Treasury and afterwards traded in a very liquid secondary market. (Datastream USBD10Y)

World

Gold Bullion Scope: World (United Kingdom), Quotation: London Bullion Market Association, 15.00 GMT. The benchmark for gold prices is the priced published in the fixing by the London Bullion Market Association twice daily, see [231]. Interestingly even in London the price is quoted as *US Dollar* per troy ounce. Gold has a special meaning as *replacement* currency, especially

related to the US Dollar. Furthermore, Gold is important in the electronics and jewelry industry. The economic importance of Gold seems stable, see [333]. (Datastream GOLDBLN).

Brent Crude Oil Scope: World (Europe), Quotation: Intercontinental Exchange, 6.30 GMT. Together with West Texas Intermediate and Dubai Crude the Brent Crude Oil coming from the North Sea belongs to the benchmark oil notations. It's price is widely followed and can be traded via futures on the Intercontinental Exchange and with lower liquidity on the New York Mercantile Exchange. (Datastream OILBREN)

CRB Index Scope: World (United States), Quotation: Commodity Research Bureau, 3.00 GMT. The continuous commodity index is determined daily by the Commodity Research Bureau based on the price in US Dollars of 22 different commodities, see [274]. It consists of two broad categories: industrial materials and food. The index is focused on the United States price of commodities. But due to the fact that many goods are traded in US Dollars for similar prices worldwide it also reflects the worldwide commodity price level. As the commodities listed in the CCI serve as input materials for other products the CCI is regarded as leading indicator: a general price rise in commodities will cause a price rise, or *inflation*, later in end consumer products. (Datastream NYFECRB)

Baltic Exchange Dry Index Scope: World, Quotation: Baltic Exchange, 17.00 GMT. This index is an aggregate of shipping rates on the 20 most important dry bulk shipping routes, see [266]. It reflects the price for the transport of merchandise. This index is also considered as leading indicator using the following rationale: a slowing or shrinking economy will manifest itself, among others, in reduced demand. This causes demand for transportation capacity to shrink. Because shippers can not simply mothball their ships an overcapacity occurs and the prices for bulk shipping drop. (Datastream BALTICF)

For the following analysis one applies a small preliminary transformation to the interest rates. The actual *level* of interest rates is not as important as the *difference* between long and short term interest rates. Fixed income professionals will typically talk of this *yield curve* saying, e.g., «I have a flattish view.» when they expect the difference between long and short term rates to decrease, see also [212]. Typically, long term interest rates are expected to be higher than short term interest rates. There is ample economic theory to explain this, see again [212] and also [333]

Name	Trading Hours	Website
Tokyo Stock Exchange	0.00–2.00 and 3.30–6.00	www.tse.or.jp
Korea Exchange	0.00–6.00	www.krx.co.kr
London Stock Exchange	8.00–16.30	www.londonstockexchange.com
Xetra (Deutsche Börse)	8.00–16.30	deutsche-boerse.com
Euronext Paris	8.00–16.30	www.bourse-de-paris.com
Borsa Italiana	8.00–16.30	www.borsaitaliana.it
Eurex	8.00–21.00	www.eurexchange.com
New York Stock Exchange	15.30–22.00	www.nyse.com
NASDAQ OMX	15.30–22.00	www.nasdaq.com
Chicago Mercantile Exchange	11.00–10.00 (electronic)	www.cmegroup.com
Intercontinental Exchange	12.30–11.30 (electronic)	www.theice.com

Table 4.2: Relevant exchanges for time series. Trading hours are uniformly in GMT. Keep in mind that every time series analysis must account for timezone differences. Using European data the *same-day* closing data from the United States is not available to us, one has to use data from the *previous* day.

for an introduction. The quintessence is that investors want to be compensated for taking a longer liquidity penalty.

However, at times, the yield curve is termed *inverted* when short term interest rates are higher than long term interest rates. Inverted yield curves are actually quite common as one can see, e.g., on figure 4.3 on page 136. Inverted yield curves may serve as leading indicator for a recession: investors then prefer the additional security of a long term government investments. I.e., demand for long-term government bonds increases and therefore the yield goes down. For all these reasons the author uses the following albeit crude definition of a yield curve:

$$\text{yield curve} := 10 \text{ years interest rates} - 3 \text{ months interest rates}.$$

The author is aware of the fact that the yield curve generally consists of much more points starting with overnight yields up to 50 years yields. 3 months and 10 years rates should provide a sufficiently accurate view of the market.

4.4 Side Note: Data Acquisition Caveats

The author noticed during preparation of the dataset that data acquisition is not always straightforward. In the few following paragraphs the author describes the relevant details of what one should look out for when assembling market ensemble datasets.

First, it is important to note that the data has to come from a reliable source or cleaned by hand. As the model building and subsequent training procedure depends entirely on the data outliers or missing data have to be identified carefully. Adequate measures for dealing with this kind of problems have to be taken. This is detailed in section 4.5. It helps considerably when data comes from a unified source. In the present case data is taken from Thomson Reuters Datastream. Most of the historical data is also available freely, often at the exchange or source of origin. But the work one has to put into unifying the data is considerable and warrants the use of a database provider.

Second, when analyzing markets from different regions around the world, time-zone considerations are very important. If not catered for look-ahead bias is easily incurred as detailed above. But how can one determine the time of publication of a financial value? The following rules are helpful:

- When dealing with data from a *single* specific exchange check the exchange website for trading hours. Closing prices are generally determined in a closing auction which lasts five minutes and published immediately afterwards. Do not take into account *after hour* or similarly called trading. After hour trading generally occurs with reduced liquidity and is not widely followed in the media. The above applies, e.g., for the non composite equity indices.

- Closing prices of composite indices are published just after the close of all involved exchanges.

- Benchmark interest rates are determined once a day by asking important market participants and averaging. The rate is often published late in the morning trading session, local time. The exact publication time is noted on the relevant website.

- Exchange rates are quoted around the clock. Nowadays exchange rates are even quoted on weekends. Different institutions and exchanges publish *reference* rates. These are often published middays or at the close of trading at

the given locality. The author prefers to choose an important financial center which is close to the region of his principal interest. In the analysis the author takes London closing prices. Another possibility for Europe is to use ECB reference rates. Other financial centers with certainly tradeable rates are New York and Tokyo.

- It may not be clear where your data series come from. For Datastream the author suggests to use the GEOGN datatype which describes the regional market of a series. Datatype EXNAME returns the name of the exchange, if any, from which the data series comes. Finally, if nothing else helps, the author highly recommend to ask the technical support. With some perseverance the author was able to get an amazing amount of undocumented information.

Third, one might need some creativity to build meaningful datasets. Your data provider might not be able to provide *exactly* the data series one wants but perhaps only something slightly similar. E.g., the author uses the evolution of the yield curve as input. This is, however, not available as time series on its own. The reason is that it is a priori not clear which and how many points of the yield curve the user would want. Yet, it is easy to build the yield curve by calculating the differences of relevant benchmark interest rates.

4.5 Data Preprocessing and Analysis

4.5.1 Correlation

In the following the author describes the salient features of the dataset. Although subsequent focus lies on analysis of the data with neural networks it is nevertheless useful to start with basic descriptive statistics. For multivariate time series the cross-correlation analysis provides important information, see tables 4.3–4.6. For the analysis the author uses Pearson's correlation coefficient

$$R(X,Y) := \frac{1}{T-1} \sum_{t=1}^{T} \left(\frac{x_t - \bar{x}}{s_x} \right) \left(\frac{y_t - \bar{y}}{s_y} \right)$$

for two time series $X = (x_1, x_2, \ldots, x_T)$, $Y = (y_1, y_2, \ldots, y_T)$, the means $\bar{x}$, $\bar{y}$ and standard deviations s_x, s_y. One has to keep in mind that the *same time* correlation matrix has no *predictive power*. I.e., if the data is correlated today there is *no*

necessity for it to be related today *and* tomorrow. However the correlation analysis is helpful when estimating which time series might be linearly related and which might have a non-linear relationship. In a *same time* cross-correlation analysis one can obtain two qualitatively different results:

- The absolute value of R is high, e.g., $|R| \geq 0.7$. This indicates a strong positive or negative linear relationship. One anticipates that linear models would work well on describing this particular relationship and that a neural network would use small weights.

- The absolute value of R is small, e.g., $|R| \leq 0.3$. This indicates linearly uncorrelated time series. In the context of market modeling this might mean that there is indeed no mutual influence of the markets. However, it might well be the case that a non-linear relationship exists that can be favorably exploited by neural networks.

A value of R in between signalizes a mixed mutual influence and one can not draw any a priori conclusions. In any case one shouldn't put too much weight on an interpretation of the correlation coefficient, because it is really only designed to discover *linear* relationships, see, e.g., [72]. Remember, that many obvious or interesting relationships have a correlation coefficient of zero. Think, e.g., of x-y pairs of points distributed along a circle. So, why look at all at correlation coefficients in a book which focuses on neural networks, a notably *non-linear* technique? One reason is, that the concept of linear correlation is easy and intuitive to grasp: the higher the absolute value of R the «more linear» the relationship is. The second reason is that a correlation analysis can prove valuable when selecting which inputs to include in a neural network topology.

The author starts the analysis with an overview of the *internal* cross correlation of the four markets: equities, interest rates, exchange rates and commodities. The first block consists of the cross-correlations between different equity indices. These are found on tables 4.3 on page 125 and 4.4. One expects the indices to exhibit a significant positive correlation because of world markets moving together. Indeed, all coefficients are positive. However, one notices that the South Korean Kospi index shows a notably lower correlation to all other indices.

At this point it is not possible to say without further analysis if the Kospi indeed might *lead* the others. As it is relatively independent it might offer valuable additional information as stated above. One also note that especially the technology

loaden NASDAQ and the more traditionally oriented Kospi are almost uncorrelated with $R = 0.009$. Not so marked but still noticeable is that the NASDAQ is generally less correlated to all other indices.

For the interest rate analysis it is useful to first concentrate on the yield curves. One notes that *Western* yield curves are quite correlated: interest rates in the United Kingdom, Germany, France, Italy and the United States move almost in lockstep with the US being noticeably more independent. Especially the three countries from continental Europe, i.e., Germany, France, Italy, show a very high degree of correlation, $R \geq 0.955$.

This could indicate that one of the yield curves could serve as proxy for the other two. Presumably one would take the German yield curve as this is the most important economy in continental Europe. One must be careful not to draw premature conclusions.

The Japanese yield curve features a significantly less important correlation to the other nations. Interestingly the correlation between the US and Japan is very low, $R = 0.058$. This reflects the traditionally independent interest rate politics of Japan. One can hope that the Japanese yield curves gives us valuable additional information as Japan is the second biggest economy by GDP after the US. Or, including the combined output of the European Union, Japan is on rank 3. Finally one sees that 12 months LIBOR and 3 months EURIBOR are, not unsurprisingly, positively related but not strongly so. This reflects the relative independence of the UK vis-à-vis continental Europe.

Looking at the currency cross-rates on tables 4.5 on page 127 and 4.6 on page 128 one first notes a very high correlation between the Swiss Franc to US Dollar and Euro to US Dollar exchange rates, $R = 0.981$. This is due to the fact that Switzerland is mostly seen to be economically *similar* to the Euro area. It is therefore not unanticipated that the respective exchange rates behave correlatedly. Further one notes a less strong correlation of the Great British Pound exchange rate to SFR|USD and EUR|USD.

This moderate correlation backs the thesis that European exchange rates behave similarly but that the United Kingdom is more independent. Finally, one notes a very low correlation to all exchange rates for USD|JPY. Especially the correlation between GBP|USD and USD|JPY is surprisingly low, $R = -0.153$, although Japan and the United Kingdom are important trade partners. In fact, Japan is the third largest export market of the United Kingdom as computed by the UK trade invest service.

The generally low correlations of the Japanese exchange rate to the other exchange rates can be attributed to relative independence of the Japanese economy. Note, that the coefficients are negative because the exchange rate is quoted in *Yen per US-Dollar* and not vice-versa.

A look at the correlation of the commodity linked time series proves very interesting. First, one notes a very high correlation between the CRB Index and the commodities gold and oil, $R = 0.927$ and $R = 0.964$. This is partially explainable by the fact that gold and oil are indeed constituents of the CRB Index. But it is also a tribute to the dominant effect that gold and especially oil prices have on most other commodities.

A detailed analysis of the inter-commodity relationships might prove insightful. It is, however, beyond the scope of this book. Additionally one notes that the Baltic Exchange Dry Index has the highest correlation with the oil price, $R = 0.797$. This relationship is explainable by the significant influence that carburant prices have on shipping costs. The similarly high correlation to the CRB Index, $R = 0.787$, is also due to many commodities having to be shipped.

Let us now leave the analysis of same instrument category and analyze inter-market relationships. The referred numbers are again found in tables 4.3–4.6. In the following the author highlights remarkable correlation and give a rationale for the finding. Keep in mind, that the data spans ten years time, which is much longer than in most comparable analyses where only a few years or even a few months are analyzed.

The possibility that the observed correlation is simply a statistical fluke is therefore remote. In the author's opinion the most revealing finding is the at first glance surprisingly high inter-market correlation of the South Korean Kospi to other assets, see especially table 4.4 on page 126. One notes a strong correlation of $R = 0.903$ between the Kospi and the CRB index. Both, Kospi and CRB, are considered leading indicators. It is consistent that they both exhibit a correlation in the same direction.

When the economy is preparing to boom, commodity prices should rise and the South Koreans should get more demand on ships and electronics products. Additionally, the South Korean economy is not so dependent on the commodities listed in the CRB and therefore a price rise in those commodities doesn't impair their economy.

A similar argument applies especially for the strong correlation between the Kospi and the oil price with $R = 0.867$. High oil prices do not harm the Koreans but signify that demand for and consumption of motor powered vehicle is high, ships included. The high $R = 0.839$ of Kospi and Gold is perhaps less clear but can be supported by the argument that Gold demand could be self-induced by the Korean economy. Indeed, Gold is in small but nevertheless substantial quantities indispensable for the manufacturing of electronics products. A flourishing Korean economy could cause the demand for Gold to rise, causing the prices to appreciate. This argument is consistent with an $R = 0.819$ between the Kospi and the EUR|USD exchange rate. As the exported products are generally priced in US Dollars the Koreans profit from a *depreciating* Dollar. As the Dollar looses value their exported products are more attractively priced, e.g., in the Euro area.

Next, one notes a high correlation between the CRB index and the EUR|USD exchange rate, $R = 0.902$. This comes as no surprise: a depreciating Dollar will *almost automatically* cause the price of imported goods that are priced in Dollars to rise.

This is immediately noticeable, too, when one looks at the correlation between gold and EUR|USD, $R = 0.870$, and oil and EUR|USD, $R = 0.868$, see table 4.6 on page 128. Similar arguments also apply for the SFR|USD exchange rate which exhibits the same behavior, albeit with slightly lower correlation coefficients.

To conclude the same-day correlation analysis of the dataset one may remember the following points:

- Intra-market relationships are generally strong. I.e., one finds strong relationships among the world's equity indices, interest rates or yield curves, exchange rates and commodities.
- Intra-region relationships can be very strong. This is true especially for the European market.
- Japanese instruments are generally more independent compared to European and North American instruments.
- The Kospi index fulfills a special role in that it is highly correlated to time series outside the Asia Pacific area.
- A same-day correlation analysis does not tell us if one time series «predicts» the other. It also does not involve any causality. But correlated assets are a good starting point for further analysis.

Mnemonic	FTSE100	DAXINDX	FRCAC40	FTSEMIB	DJES50I	SPCOMP
FTSE100	1.000	0.923	0.957	0.919	0.935	0.951
DAXINDX	0.923	1.000	0.915	0.816	0.888	0.881
FRCAC40	0.957	0.915	1.000	0.970	0.987	0.920
FTSEMIB	0.919	0.816	0.970	1.000	0.961	0.907
DJES50I	0.935	0.888	0.987	0.961	1.000	0.884
SPCOMP	0.951	0.881	0.920	0.907	0.884	1.000
NASA100	0.731	0.700	0.786	0.758	0.831	0.713
JAPDOWA	0.920	0.819	0.873	0.866	0.846	0.907
KORCOMP	0.409	0.555	0.266	0.150	0.162	0.449
BBGBP12	0.783	0.717	0.786	0.789	0.788	0.839
ECEUR3M	0.631	0.721	0.675	0.588	0.691	0.561
UKyc	-0.696	-0.689	-0.685	-0.673	-0.652	-0.756
GERyc	-0.453	-0.609	-0.449	-0.342	-0.395	-0.418
FRyc	-0.477	-0.606	-0.471	-0.385	-0.413	-0.461
ITyc	-0.528	-0.617	-0.527	-0.472	-0.471	-0.539
USyc	-0.784	-0.767	-0.756	-0.714	-0.699	-0.730
JAPyc	0.234	-0.015	0.264	0.381	0.279	0.272
USDOLLR	0.248	0.275	0.133	0.145	0.022	0.441
SWISFUS	-0.171	-0.007	-0.304	-0.376	-0.410	-0.051
USEURSP	-0.013	0.141	-0.156	-0.230	-0.265	0.115
JAPAYEUSD	0.106	-0.018	0.157	0.244	0.173	0.071
GOLDBLN	0.035	0.277	-0.079	-0.234	-0.175	0.043
OILBREN	0.279	0.437	0.164	0.057	0.061	0.344
NYFECRB	0.217	0.406	0.101	-0.016	-0.009	0.278
BALTICF	0.251	0.382	0.160	0.107	0.079	0.419

Table 4.3: Correlation matrix of the analyzed data. Note that the respective interest series have already been transformed into yield curves. A same-day correlation matrix offers no predictive power but is nevertheless useful for determining which data might exhibit linear or non-linear relationship. This and the following table clearly show us that most world equity indices are highly correlated. We can especially notice a strong correlation among European equity indices. On the other hand, the Korea composite or Kopsi index is more independent from other *equity* indices. See table 4.1 on page 109 for an explanation of the mnemonic codes. Continued on tables 4.4–4.6

Mnemonic	NASA100	JAPDOWA	KORCOMP	BBGBP12	ECEUR3M	UKyc
FTSE100	0.731	0.920	0.409	0.783	0.631	-0.696
DAXINDX	0.700	0.819	0.555	0.717	0.721	-0.689
FRCAC40	0.786	0.873	0.266	0.786	0.675	-0.685
FTSEMIB	0.758	0.866	0.150	0.789	0.588	-0.673
DJES50I	0.831	0.846	0.162	0.788	0.691	-0.652
SPCOMP	0.713	0.907	0.449	0.839	0.561	-0.756
NASA100	1.000	0.742	0.009	0.694	0.463	-0.517
JAPDOWA	0.742	1.000	0.401	0.768	0.438	-0.622
KORCOMP	0.009	0.401	1.000	0.198	0.201	-0.383
BBGBP12	0.694	0.768	0.198	1.000	0.697	-0.828
ECEUR3M	0.463	0.438	0.201	0.697	1.000	-0.705
UKyc	-0.517	-0.622	-0.383	-0.828	-0.705	1.000
GERyc	-0.133	-0.251	-0.512	-0.416	-0.798	0.728
FRyc	-0.121	-0.282	-0.513	-0.459	-0.795	0.765
ITyc	-0.157	-0.346	-0.483	-0.546	-0.807	0.819
USyc	-0.560	-0.745	-0.463	-0.596	-0.557	0.776
JAPyc	0.314	0.396	-0.335	0.232	-0.304	0.114
USDOLLR	-0.093	0.328	0.750	0.305	-0.007	-0.450
SWISFUS	-0.380	-0.139	0.741	-0.196	-0.207	-0.092
USEURSP	-0.262	0.021	0.819	-0.049	-0.109	-0.214
JAPAYEUSD	-0.206	0.059	-0.211	0.079	0.180	0.058
GOLDBLN	-0.215	-0.005	0.839	-0.153	0.073	-0.130
OILBREN	-0.063	0.220	0.867	0.205	0.281	-0.456
NYFECRB	-0.126	0.173	0.903	0.094	0.214	-0.372
BALTICF	-0.029	0.211	0.754	0.294	0.220	-0.439

Table 4.4: Correlation matrix. Continued from table 4.3. This table offers interesting insights in the Kospi index as leading indicator. We note especially the high correlation of the Kospi to the EUR|USD exchange rate, to the Gold and oil price and to the commodity index CRB. All this can be explained by the fact, that the South Korean economy will generally be among the first to profit from a booming world economy as they are among the world largest shipbuilders. We also see that yield curves, indicated by the suffic »yc« are still dependent on one another but more independent than the equity indices in the previous table.

Menemonic	GERyc	FRyc	ITyc	USyc	JAPyc	USDOLLR	SWISFUS
FTSE100	-0.453	-0.477	-0.528	-0.784	0.234	0.248	-0.171
DAXINDX	-0.609	-0.606	-0.617	-0.767	-0.015	0.275	-0.007
FRCAC40	-0.449	-0.471	-0.527	-0.756	0.264	0.133	-0.304
FTSEMIB	-0.342	-0.385	-0.472	-0.714	0.381	0.145	-0.376
DJES50I	-0.395	-0.413	-0.471	-0.699	0.279	0.022	-0.410
SPCOMP	-0.418	-0.461	-0.539	-0.730	0.272	0.441	-0.051
NASA100	-0.133	-0.121	-0.157	-0.560	0.314	-0.093	-0.380
JAPDOWA	-0.251	-0.282	-0.346	-0.745	0.396	0.328	-0.139
KORCOMP	-0.512	-0.513	-0.483	-0.463	-0.335	0.750	0.741
BBGBP12	-0.416	-0.459	-0.546	-0.596	0.232	0.305	-0.196
ECEUR3M	-0.798	-0.795	-0.807	-0.557	-0.304	-0.007	-0.207
UKyc	0.728	0.765	0.819	0.776	0.114	-0.450	-0.092
GERyc	1.000	0.992	0.955	0.651	0.602	-0.288	-0.279
FRyc	0.992	1.000	0.982	0.654	0.551	-0.348	-0.273
ITyc	0.955	0.982	1.000	0.652	0.456	-0.399	-0.213
USyc	0.651	0.654	0.652	1.000	0.058	-0.299	-0.057
JAPyc	0.602	0.551	0.456	0.058	1.000	-0.080	-0.463
USDOLLR	-0.288	-0.348	-0.399	-0.299	-0.080	1.000	0.752
SWISFUS	-0.279	-0.273	-0.213	-0.057	-0.463	0.752	1.000
USEURSP	-0.342	-0.342	-0.297	-0.173	-0.427	0.826	0.981
JAPAYEUSD	0.038	-0.027	-0.126	0.138	0.235	-0.153	-0.499
GOLDBLN	-0.500	-0.457	-0.353	-0.251	-0.596	0.546	0.868
OILBREN	-0.625	-0.630	-0.600	-0.385	-0.407	0.732	0.792
NYFECRB	-0.613	-0.606	-0.554	-0.373	-0.492	0.736	0.862
BALTICF	-0.439	-0.471	-0.498	-0.209	-0.299	0.839	0.717

Table 4.5: Correlation matrix. Continued from table 4.4. This and the following table show a high correlation between the world's leading exchange rates. We especially notice one of the highest correlation of the entire analysis between the SFR|USD and EUR|USD exchange rate. This shows that on the monetary market Switzerland and the Euro area are perceived to be very similar.

Mnemonic	USEURSP	JAPAYEUSD	GOLDBLN	OILBREN	NYFECRB	BALTICF
FTSE100	-0.013	0.106	0.035	0.279	0.217	0.251
DAXINDX	0.141	-0.018	0.277	0.437	0.406	0.382
FRCAC40	-0.156	0.157	-0.079	0.164	0.101	0.160
FTSEMIB	-0.230	0.244	-0.234	0.057	-0.016	0.107
DJES50I	-0.265	0.173	-0.175	0.061	-0.009	0.079
SPCOMP	0.115	0.071	0.043	0.344	0.278	0.419
NASA100	-0.262	-0.206	-0.215	-0.063	-0.126	-0.029
JAPDOWA	0.021	0.059	-0.005	0.220	0.173	0.211
KORCOMP	0.819	-0.211	0.839	0.867	0.903	0.754
BBGBP12	-0.049	0.079	-0.153	0.205	0.094	0.294
ECEUR3M	-0.109	0.180	0.073	0.281	0.214	0.220
UKyc	-0.214	0.058	-0.130	-0.456	-0.372	-0.439
GERyc	-0.342	0.038	-0.500	-0.625	-0.613	-0.439
FRyc	-0.342	-0.027	-0.457	-0.630	-0.606	-0.471
ITyc	-0.297	-0.126	-0.353	-0.600	-0.554	-0.498
USyc	-0.173	0.138	-0.251	-0.385	-0.373	-0.209
JAPyc	-0.427	0.235	-0.596	-0.407	-0.492	-0.299
USDOLLR	0.826	-0.153	0.546	0.732	0.736	0.839
SWISFUS	0.981	-0.499	0.868	0.792	0.862	0.717
USEURSP	1.000	-0.473	0.870	0.841	0.902	0.796
JAPAYEUSD	-0.473	1.000	-0.462	-0.265	-0.327	-0.195
GOLDBLN	0.870	-0.462	1.000	0.836	0.927	0.626
OILBREN	0.841	-0.265	0.836	1.000	0.964	0.797
NYFECRB	0.902	-0.327	0.927	0.964	1.000	0.787
BALTICF	0.796	-0.195	0.626	0.797	0.787	1.000

Table 4.6: Correlation matrix. Continued from table 4.5. In this table we can note the high correlation among commodities. We also note that the Baltic Exchange Dry Index behaves more independently. However, the BDI exhibits a high correlation to the GBP|USD exchange rate with almost as high a correlation to the EUR|USD exchange rate. We can interpret this as a general dependence of the BDI on the US Dollar with the BDI appreciating when the US Dollar depreciates.

4.5.2 Descriptive Statistics for Level Series

In the following the author has an exemplary look at the problems which arise when dealing with financial time series. The author uses the Jarque-Bera test to verify normality, see [193]. The Jarque-Bera statistic JB measures how much a given distribution with unknown kurtosis and skewness deviates from a normal distribution. It is given by

$$JB = \frac{T}{6}\left(M_{\text{skewness}}^2 + \frac{(M_{\text{kurtosis}} - 3)^2}{4}\right)$$

where T indicates the sample length and $M_{\dots}$ the appropriate moment of the distribution.

JB measures *excess* kurtosis as the normal distribution has a kurtosis of 3. Therefore any additional skewness and kurtosis increases the Jarque-Bera statistic. The corresponding p-value on table 4.7 on page 133 describes how likely it is to obtain the given JB *if* the underlying distribution is indeed normal, see [283]. As this probability is below 1 percent for every series in the dataset one can conclude at the 99 percent confidence level that *all* level series are non normal. Starting with a normal distribution one can interpret the skewness and kurtosis values in table 4.7 on page 133 as follows:

- The skewness indicates an asymmetry in the tails, where the normal distribution is perfectly symmetric with $M_{\text{skewness}} = 0$. A negative skewness means that the left tail is more developed. As an example take the FTSE100 distribution on figure 4.2 on page 135, however this is perhaps not so obvious from the graph.

  A positive skewness indicates a longer right tail. A very clear example of this is shown in the distribution of the Gold Bullion prices on figure 4.5 on page 138. See below for more explanation of this phenomenon. The skewness is the third moment of a distribution. It is calculated as

$$M_{\text{skewness}} = \frac{\frac{1}{T}\sum_{t=1}^{T}(x_t - \overline{x})^3}{\left(\frac{1}{T}\sum_{t=1}^{T}(x_t - \overline{x})^2\right)^{3/2}}.$$

- The kurtosis shows how «peaked» the distribution is. As explained above a kurtosis below 3 indicates a platykurtic distribution. This can be interpreted

as fewer values being near the mean and a *lower* probability of extreme values. The multimodal distribution of the GBP|USD exchange rate level series, see figure 4.4 on page 137, is a good example for low kurtosis. On the other hand a kurtosis above 3 indicates a leptokurtic distribution with more values proximate to the mean and a *higher* probability of extreme values. These are also called *fat tails*.

One meets fat tails again when looking at *return* series, see, e.g., the highly peaked distribution of the UK yield curve *differences* on figure 4.8 on page 146. The skewness is the fourth moment of a distribution and is calculated from the sample by

$$M_{\text{kurtosis}} = \frac{\frac{1}{T}\sum_{t=1}^{T}(x_t - \overline{x})^4}{\left(\frac{1}{T}\sum_{t=1}^{T}(x_t - \overline{x})^2\right)^2}.$$

The author now looks at exemplary statistics from each of the four asset classes, i.e., the FTSE 100 equity index, the UK yield curve, the GBP|USD exchange rate and the Gold Bullion. Each of these four examples shows individual characteristics of the given asset class and common characteristics of financial time series *in levels*. This is intended to give us a taste of which kind of transformation, if any, may be necessary before trying to fit a model. All time series are plotted and the corresponding detailed statistics are found in table 4.7 on page 133.

FTSE 100 Index The FTSE 100 level series and corresponding distribution is plotted on figure 4.2 on page 135. The sample period is representative of different market phases: it includes the bear market from 2000–2003 following the collapse of the «new economy». Then, one sees the impressive bull market of 2003 to mid 2007. And then again, one observes the precipitated collapse until mid 2009 which is known as «credit crisis».

The sample thus includes more than one full business cycle which is said to last approximately seven years. Indeed, this business cycle can be observed from 2000–2007, peak to peak, but the author won't delve further into the analysis, whether and why this market lore might be justified. Looking at the distribution of levels one notes first a slight negative skewness, which is also noticeable in the graph.

However, the distribution is bi-modal with peaks around 4250 and 6000. A third smaller peak around 5000 can also be seen. The kurtosis of 1.72 is misleading as it centers actually around the mean at 5311 points. However,

this is only a secondary peak. The Jarque-Bera statistic indeed confirms at more than 99 percent confidence that the level series is non normal. The double-peakedness indicates that a transformation of the series is advisable for it to be useful in further analysis: the mean shifts over time. Or, expressed the other way around, the FTSE 100 series is not mean reverting. However, one is not surprised by this finding, as it is common for equity time series to be biased upwards. Think about it: if they weren't, at least in the long term, there would be no reason for an investor to be *long* equities for a substantial time period.

UK yield curve The UK yield curve and corresponding distribution is shown on figure 4.3 on page 136. One notes that the volatility has considerably augmented since the credit crisis in 2007. And since the end of 2008 one notes a widening gap between short and long term yields. This is, at least partly, due to decisive open market operations of the Bank of England in an attempt to stabilize the ailing British economy: The reference interest rates have been lowered and the market has been flooded with short term liquidity. The yield curve also shows two peaks around 0.0 and 0.5, although these are not very clear. One can also see that the tails are fat as indicated by the high kurtosis of 3.52.

US Dollar to Great British Pound One notes an appreciation of GBP relative to USD in a very long trend starting in mid 2001 and only sharply reversing in 2008, see figure 4.4 on page 137. Such long lasting trends are typical for exchange rates, see [333]. The ailing British economy and recovery hopes on the US side are mainly responsible for the drop between mid 2008 and the beginning of 2009. It is important to note that here, too, the sample contains up and down phases. Thus the model should not overfit to particular trend. However, a model trained on data up to 2008 and using the rest of the interval for testing might be ill-equipped to handle the downturn. A look at the distribution of returns is not very conclusive: the distribution is multi-modal with several not very clear peaks.

The kurtosis of 1.68 is the second smallest in the entire dataset and highlights a typical feature of exchange rates: they tend to move in ranges, see again [333]. This can be explained by the argument that exchange rates have a *fair value* which is determined by a basket of goods. If the exchange rate deviates too much, arbitrage becomes possible.

The arbitrage argument is a weak one as tariffs and shipping costs often prevent arbitrage opportunities. But in the long run the arbitrage argument holds. Another argument in favor of the range behavior is that central banks will tend to intervene if the exchange rate moves too far away from what is *perceived* as acceptable.

Gold Bullion Figure 4.5 on page 138 shows the level series and distribution. The distribution is typical for a commodity, see [200], p. 97: most observations are clustered around relatively low values, here in the range of USD 250–300. However, the consistent trend from 1999–2009 with only minor corrections is atypical. It is, partly, due to a depreciation of the US Dollar against other major currencies and highlights the role of gold as *reserve currency*. As stated previously the skewness of 0.82 indicating a stronger right tail is quite remarkable. In so far as modeling is concerned One expects some problems when using the correction of 2008–2009 as out-of-sample data. Clearly, also, the mean of the gold series is not constant over time. It is even questionable if one can sensibly speak of a mean when considering this series. The series will therefore require some kind of transformation before modeling.

Let the author conclude the descriptive statistics for the level series with some notes on other remarkable values in table 4.7 on the next page. The USD|JPY exchange rate series features a Jarque-Bera statistic of only 9.93, low skewness and low kurtosis. However, this is not sufficient to allow for an above 1 percent change of the series being normal. One also notes that the NASDAQ 100 series has a high kurtosis of 5.80 and the highest Jarque-Bera statistic of 2221.82. As this series is notably *different* from the others one might experience some problems when trying to model it multivariately using the other series.

It is interesting that the S&P 500 index has a *low* kurtosis of 2.20 and a Jarque-Bera statistic of 109.93 in the same range as the other time series as its constituents are also equities from NASDAQ. Indeed, non-technology components have a higher weight in the S&P 500. A detailed analysis is left for further research. One also notices a high kurtosis of 4.33 paired with low skewness of 1.36 and a high Jarque-Bera statistic of 998.10 for the Baltic Exchange Dry Index. One can attribute this to the commodity like type of the BDI.

To prevent a possible misunderstanding let the author note that *non normality* and *fat tails* are not at all linked. Non normality is measured by the Jarque Bera statistic which takes into account skewness *and* kurtosis. Fat tails are a familiar

way of designating *high* kurtosis. Already [237] notes that financial time series are generally non normal. Fat tailedness is however only a general feature of *returns*. An analysis of the return series is the topic of the following section.

Table 4.7: Descriptive statistics for level series. One notes several interesting things here: most importantly the p-value of the Jarque-Bera statistic is always 0.0. This signifies that *all* level series are non normal at a confidence level greater than 99 percent. Further, most series exhibit slight skewness and significant kurtosis. This is partly due to the fact that most series are at least bimodal and in some cases multimodal. Fat tails are indicated by high kurtosis. However, *level* series are not generally fat tailed. It is only when looking at return series, see table 4.8 on page 143, that fat tails become a general phenomenon. This will be discussed in the next section.

Mnemonic	Min Max	Median Mean	SD	Skewness Kurtosis	Jarque-Bera p-value
FTSE100	3287.04	5363.01	894.69	-0.18	192.31
	6930.2	5311.37		1.72	0.000
DAXINDX	2202.96	5231.35	1434.29	0.06	106.87
	8105.69	5334.98		2.02	0.000
FRCAC40	2403.04	4504.66	1059.60	0.12	119.08
	6922.33	4520.55		1.98	0.000
FTSEMIB	12620.57	32967.06	7825.97	-0.05	52.55
	50108.56	32959.11		2.31	0.000
DJES50I	1809.976	3544.58	862.12	0.25	102.73
	5464.43	3510.30		2.17	0.000
SPCOMP	676.53	1213.55	198.57	-0.31	109.93
	1565.15	1201.34		2.20	0.000
NASA100	804.65	1589.55	729.03	1.77	2221.82
	4704.73	1799.11		5.80	0.000
JAPDOWA	7054.98	12312.75	3281.71	0.25	155.61
	20833.21	13054.47		1.91	0.000
KORCOMP	468.76	910.27	398.72	0.69	238.90
	2064.85	1036.65		2.44	0.000

Continued on next page

Mnemonic	Min Max	Median Mean	SD	Skewness Kurtosis	Jarque-Bera p-value
BBGBP12	1.5209	5.04	1.12	-0.76	303.39
	6.8877	5.04		3.71	0.000
ECEUR3M	0.995	3.31	1.07	0.12	165.40
	5.35	3.27		1.79	0.000
UKyc	-2.1714	-0.16	0.88	0.55	163.85
	3.0911	-0.06		3.52	0.000
GERyc	-1.588	1.13	0.90	-0.44	146.14
	2.5546	0.97		2.25	0.000
FRyc	-1.2935	1.20	0.88	-0.35	125.84
	2.886	1.07		2.18	0.000
ITyc	-0.7669	1.40	0.86	-0.14	59.66
	3.4899	1.30		2.31	0.000
USyc	-1.4041	0.90	1.43	0.16	210.27
	3.6635	1.07		1.65	0.000
JAPyc	-0.2221	1.26	0.40	-0.81	305.13
	1.946	1.17		3.39	0.000
USDOLLR	1.3669	1.68	0.20	0.10	192.84
	2.1082	1.70		1.68	0.000
SWISFUS	0.5494	0.78	0.11	-0.07	106.63
	1.0139	0.75		2.02	0.000
USEURSP	0.8287	1.21	0.20	0.01	101.03
	1.5979	1.17		2.04	0.000
JAPAYEUSD	87.63	112.34	8.69	-0.15	9.93
	134.83	112.72		2.97	0.007
GOLDBLN	252.85	409.14	219.49	0.82	341.05
	1011.6	487.47		2.35	0.000
OILBREN	17.15	37.99	26.02	1.28	902.90
	145.61	47.49		4.33	0.000
NYFECRB	183.49	274.73	99.92	0.86	324.08
	614.57	305.03		2.92	0.000
BALTICF	663.0	2391.00	2455.96	1.36	998.10
	11793.0	3285.73		4.33	0.000

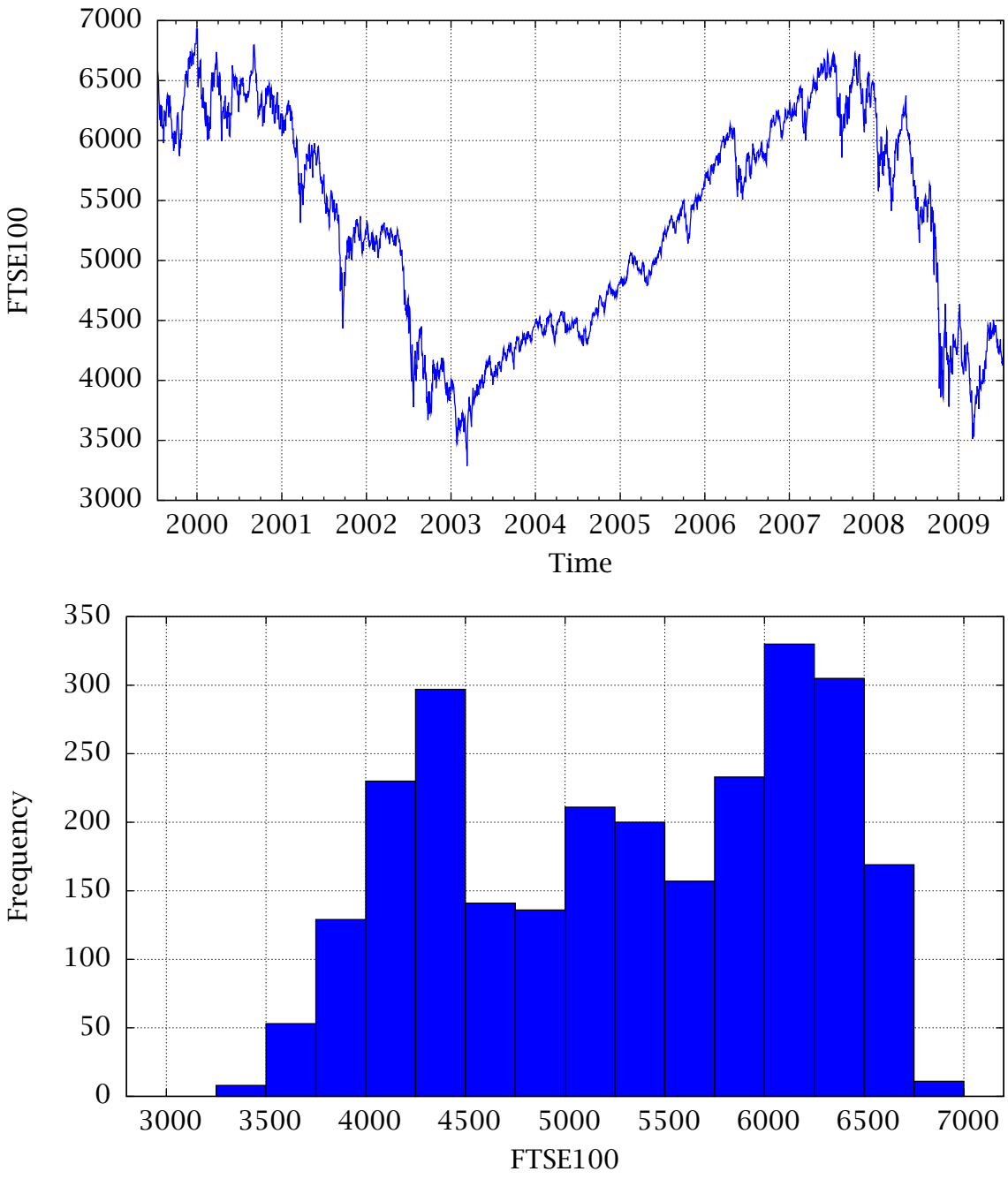


Figure 4.2: The FTSE 100 equity index from July 1999 to July 2009 and corresponding distribution. Note especially the two peaks of the distribution around 4250 and 6000. The sample shows a typical business cycle of seven years from peak to peak in the years 2000–2007. It captures the bear market of 2000, which marks the collapse of the new economy and the remarkable bull market starting in 2003. The bear market of the credit crisis starting mid 2007 can also be seen.

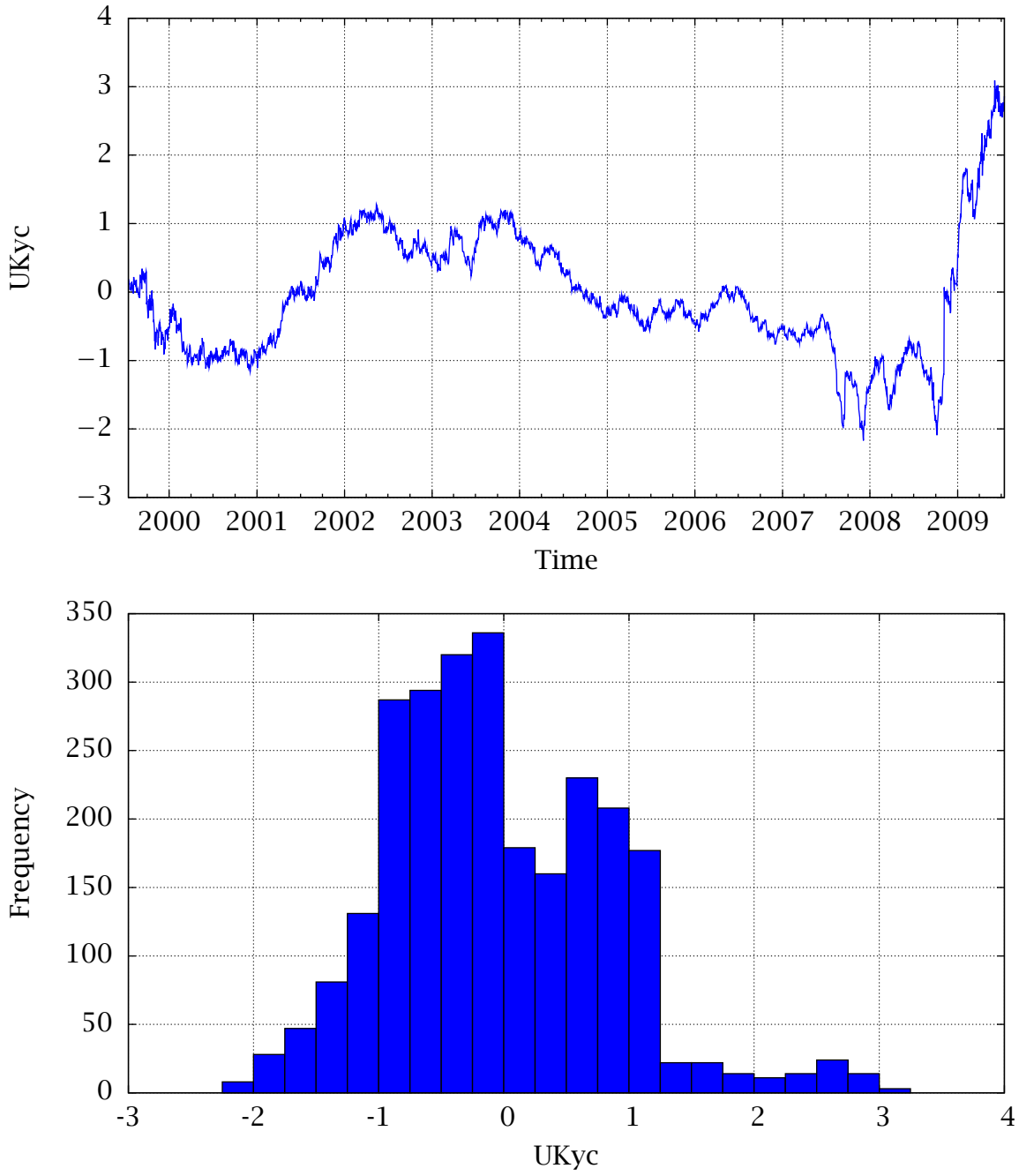


Figure 4.3: The UK yield curve and corresponding distribution. Note the positive skewness.

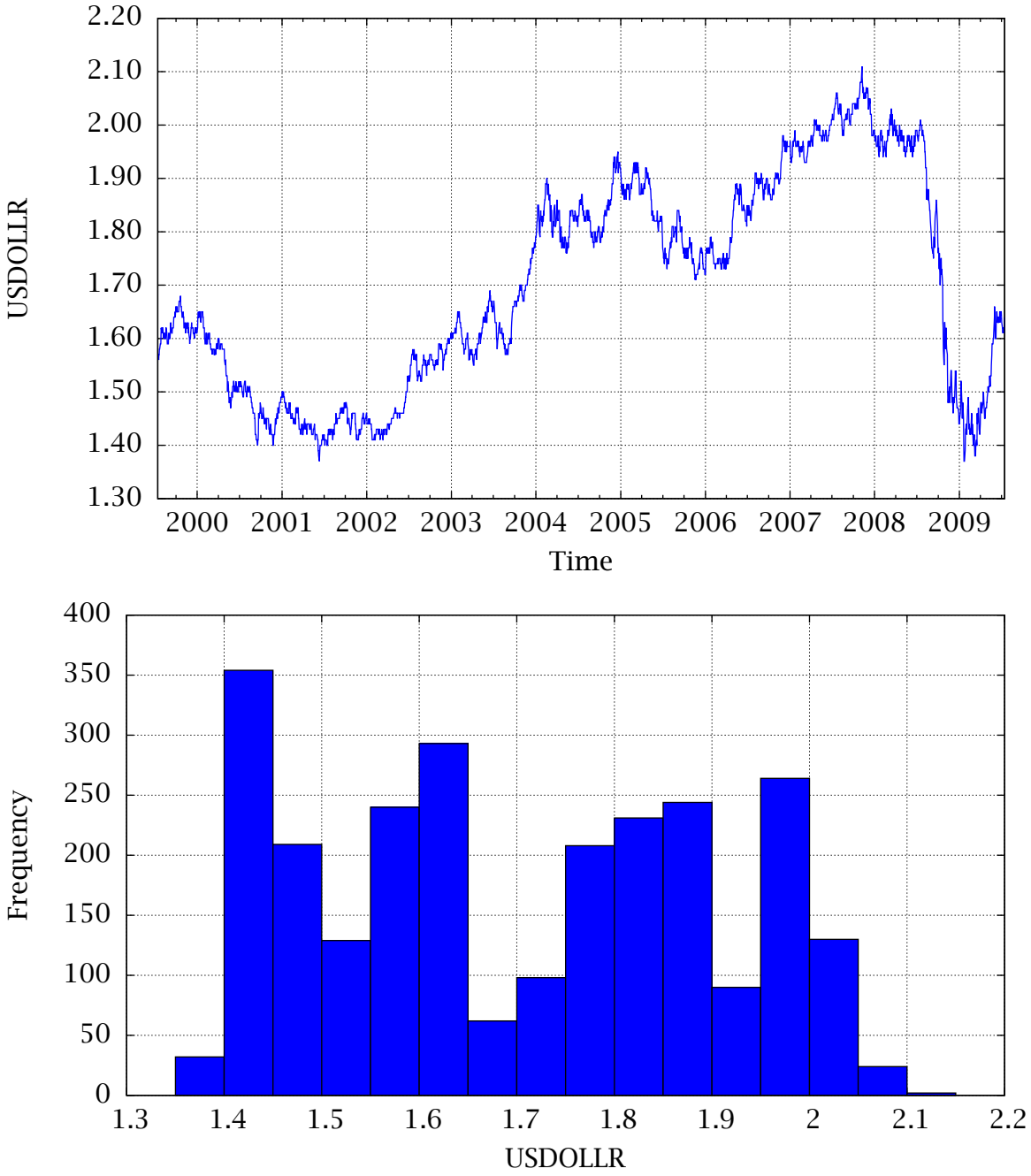


Figure 4.4: The GBP|USD exchange rate (Datastream USDOLLR), and distribution. From a modelers point of view the increased volatility at the end of the period necessitates special attention.

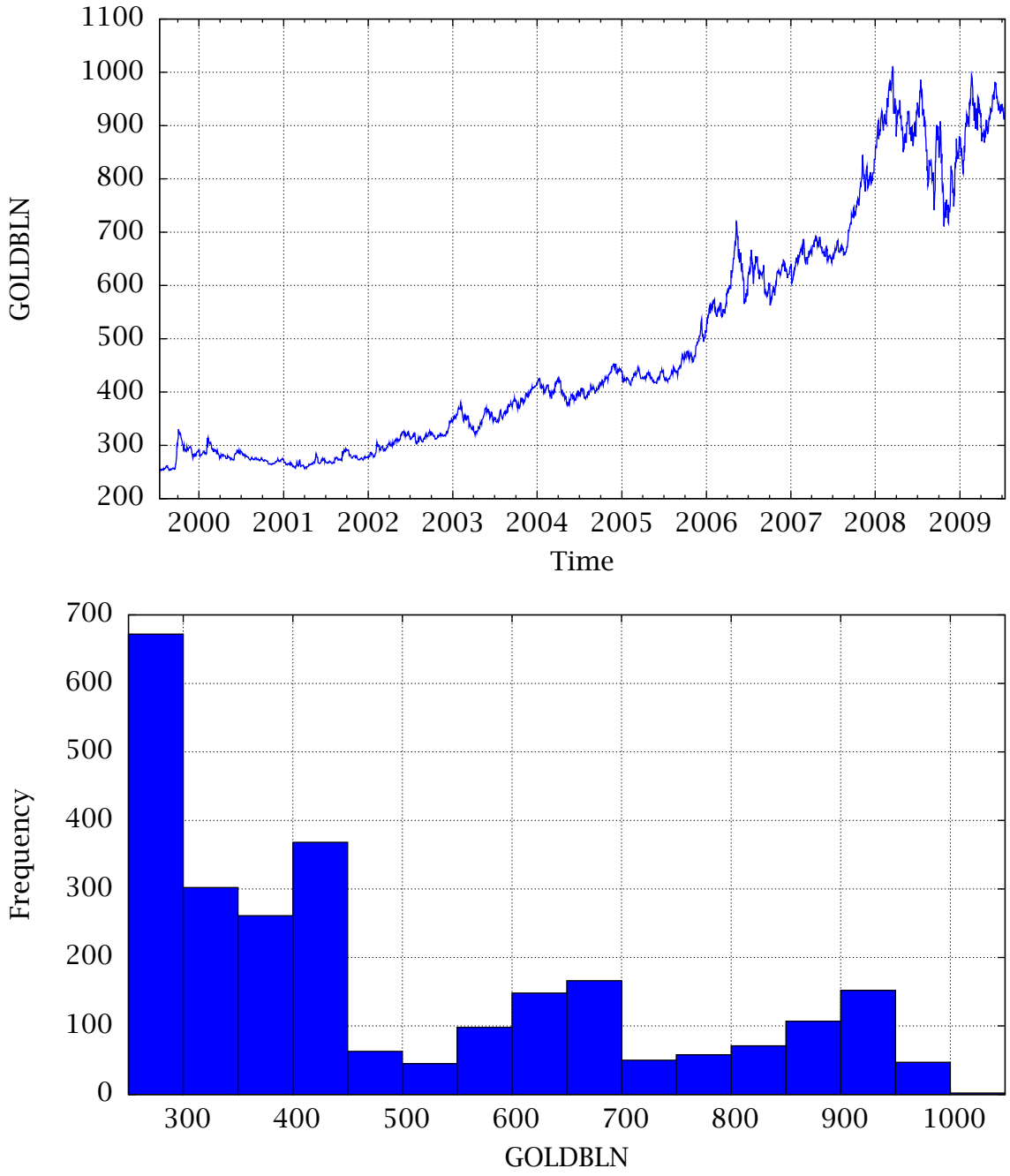


Figure 4.5: The Gold Bullion price, quoted in London in USD, price and distribution.

4.5.3 Data Transformation and Further Analysis

As one has seen the time series need to be transformed appropriately, especially because of means shifting over time. A commonly used transformation is to compute *rates of return*, see, e.g., [99]. One does this in the following way:

$$r_t := \frac{x_t}{x_{t-1}} - 1 \tag{4.1}$$

where x_t is the level series. This gives us returns centered around 0. Alternatively one can also take logarithmic returns defined as

$$r_t := \log\left(\frac{x_t}{x_{t-1}}\right) = \log x_t - \log x_{t-1} = \operatorname{diff}(\log X)_t \tag{4.2}$$

where diff represents the lagged differences operator

$$\operatorname{diff}(\cdot) := (\cdot)_t - (\cdot)_{t-1}.$$

Equation 4.1 offers a perfectly symmetric view of the returns where a gain of 10 percent is represented by $r_t = 0.1$ and loss of 10 percent is represented by $r_t = -0.1$. On the other hand, equation 4.2 skews the returns in that excessive gains are damped and excessive losses are emphasized, because of $\lim_{x\to 0}\log(x) = -\infty$. Where the logarithmic returns might be more appropriate to quantify *risk aversion* one choses the simple form of rate of returns, like in, e.g., [99]. However, both forms do not differ much if one considers typical daily returns which generally are in the range of 20% — and this would already be an extreme case, see also table 4.8. Figure 4.6 on the following page compares both return functions and one can immediately see that both functions are very similar in the typical range. Note that neither transformation function prevents *outliers* from biasing the dataset. If needed, one applies a *damping* transformation to the time series.

The yield curves deserve a special treatment. Because yield differences can also become negative it is not sensible to transform them into rates of returns. To account for changes in the yield curve one simply uses lagged differences, i.e.

$$r_t^{\mathrm{yc}} = x_t - x_{t-1}.$$

One sees on table 4.8 and in the following analysis that this transformation is sufficient to produce a mean of zero and to ensure stationarity.

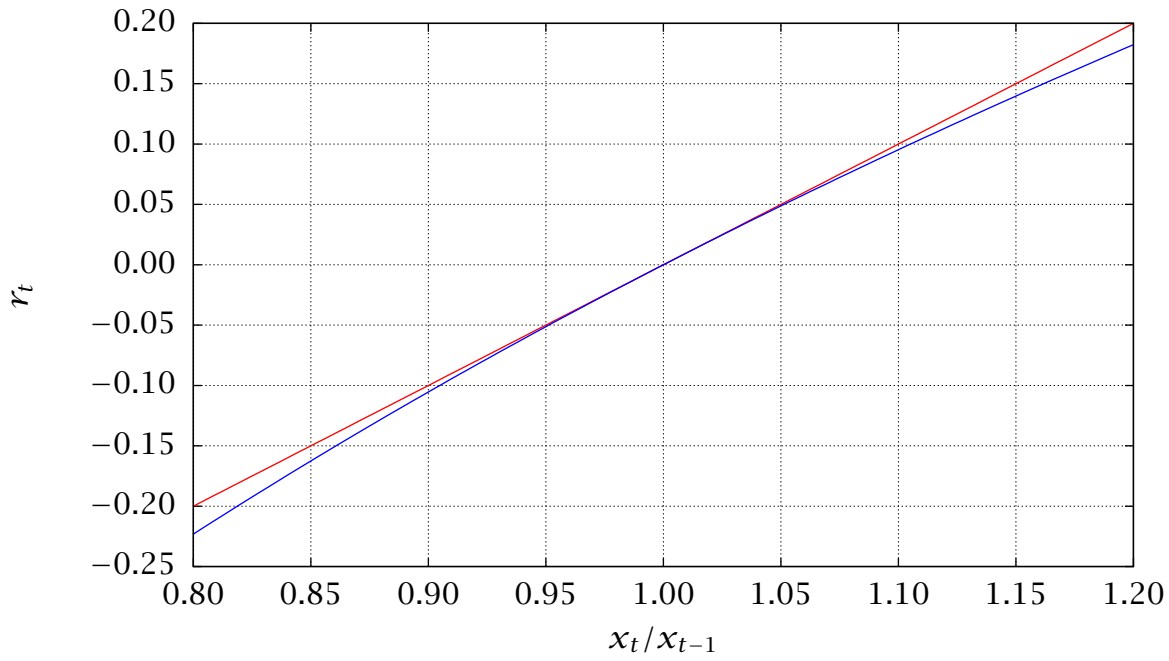


Figure 4.6: Comparison of rate of return functions. Simple returns in red, logarithmic returns in blue. Note how logarithmic returns damp gains and emphasize losses, whereas simple returns are symmetric. One can also see that both return functions do not differ by much in the typical case. Table 4.8 on page 143 shows that on a ten years span daily returns are always in the 20 percent range and even seldom exceed the 15 percent range. One uses simple rates of returns for further analysis.

Again, the author will now analyze in detail the return series for the four assets described in the previous section. All numbers can be found in table 4.8 on page 143 and on figures 4.7–4.10.

FTSE 100 Index Figure 4.7 on page 145 shows the return series and corresponding distribution. One notes a unimodal distribution with a mean at zero, slight skewness and very high kurtosis. This is indicative of fat tails. A high Jarque-Bera statistic confirms that the return series is non normal. One finds these facts confirmed in [148]. On the returns plot one notes the considerable volatility at the end of 2008. One can also verify visually the presence of volatility clusters as described in [237]. Volatility clusters are a feature of financial time series: empirically days with high volatility tend to be followed by more days with high volatility and vice versa.

UK yield curve The plot of daily differences on figure 4.8 on page 146 highlights an important challenge for model builders. Except for one peak at the end of 2008 which caused a shift in the yield curve of more than 1.2 percentage points, i.e., more than 120 basis points, the other peaks are noticeably smaller. As one has to scale the data appropriately the question emerges: how should one treat this peak? It is clearly not an outlier because the peak is confirmed by other data sources and can be explained by an intervention of the Bank of England. Had one reserved the years 2008 and 2009 for out of sample testing the peak would have most probably saturated the model.

One possibility is to *cap* the peak, e.g., at 0.6. However, an alteration of the data should always be judiciously pondered. The author chooses not to change anything and to accept the peak as it is. This, on the other hand, incurs the risk of loosing significance in the other values which are dwarfed by the peak. The distribution is slightly skewed, features a very high kurtosis and is definitely non normal as per the Jarque Bera statistic. The distribution plot shows a concentration around the mean of zero. The occurrence of extreme values is low. But when they do happen the deviations from the mean are remarkable. Extreme peaks can not be noticed in the distribution plot because of scaling.

US Dollar to Great British Pound Figure 4.4 on page 137 shows a remarkably even distribution with a skewness of zero. The returns, however, exhibit the typical volatility clusters, see again [237]. Correspondingly one observes fat tails as indicated by a kurtosis of 8.38 and a non normal distribution. The same challenge as above occurs when looking at the years 2008 and 2009. Where the returns had been mostly clamped to a range of 2 percent one can find an increased volatility where daily returns fluctuate even more than 4 percent. Again, as I'm conducting a long term analysis the author does not artificially modify the return series.

Gold Bullion The Gold Bullion return series plotted on figure 4.5 on page 138 is special in that high volatility occurs all over the time series, not only in 2008 and 2009. E.g., one notes significant volatility in 1999 and the beginning of 2000, then again in mid 2001. A quieter period follows with volatility rising again in mid 2006. One may attribute the recurrent volatility spikes to the role of Gold as a commodity. Cyclic behavior of returns is therefore not surprising, see also [54]. The series exhibits low skewness and high kurtosis.

Summarizing one sees in table 4.8 that *all* time series have a mean of zero. This is remarkable because the author didn't specially design the transformation to produce a mean of zero. However, a warning is appropriate: especially when looking at equity series one should be aware that a mean of zero is not guaranteed. Indeed, a simple analysis shows that in the very long run, over, say, fifty or more years, equity return series have a mean greater than zero. It is simply another form of saying that a buy and hold strategy has a strong chance to work out when the investment horizon is sufficiently long. The mean depends crucially on the period under investigation. If one had only chosen, e.g., a bull market phase one would have had a positive mean of the returns. A mean of zero is important when training neural networks. If the mean is another number than all weights will have to move in the same direction, see [176], p. 146. This is bad, because it *reduces* the effect of having several independent weights.

One also sees that *all* return series show a high kurtosis. The transformed interest rates and yield curve series have especially high kurtosis. This can be explained by a clustering around levels. Often, interest rates move almost in lockstep with some even value set by the central bank. For example, intervals of 25 or 50 basis points are current when setting target rates. This will cause short term rates to shift quickly when policy changes. Long term interest rates are not so affected by short term policy and will not change or change slowly. Therefore one observes a movement in discrete levels. This effect is visible quite clearly in *level* series on figure 4.3 on page 136. One can observe clustering of the yield curve around -1, 0 and 1 percent respectively. All return and difference series are *fat tailed*. The Jarque Bera statistic confirms non normality at the 99 percent confidence level for *all* series. Both features are common in financial time series, see [148].

Thinking at the future use of the data in a neural network one notes that the transformation serves to *harmonize* the time series. This is important to avoid domination of some time series by others, see [176], p. 144. Where one had levels at different order of magnitude, low or high skewness and kurtosis, one now has a data set with very similar parameters. Minima and maxima are at approximately 15 percent for return series and of the order of unity for difference series. Medians and means are zero up to two significant digits, although not exactly as the casual minus signs in table 4.8 indicate. The second moment, the standard deviation, is harmonized to the interval $0.1 \ldots 0.8$. Skewness and kurtosis still vary, but kurtosis is uniformly much greater than 3 thus always to be considered as *high*.

Table 4.8: Descriptive statistics for return and lagged difference series. The yield curve series is calculated using one day lagged difference; all other series are calculated using simple rates of returns. The transformation to return and difference series serves to *harmonize* the data. This is important to facilitate training for future neural network use. Minima and maxima are of the same order of magnitude. Median and mean are uniformly almost zero, though not exactly as the casual minus sign indicates. The transformed series feature a small standard deviation and very high kurtosis. All transformed series are *fat tailed*. The Jarque-Bera statistic confirms that all series are non normal at the 99 percent confidence level as indicated by the corresponding p-value which is uniformly zero up to three significant digits. These features are common in financial time series.

Mnemonic	Min Max	Median Mean	SD	Skewness Kurtosis	Jarque-Bera p-value
FTSE100	-0.08849274	0.00	0.01	0.04	4585.59
	0.09838771	-0.00		9.49	0.000
DAXINDX	-0.08492269	0.00	0.02	0.21	2496.79
	0.1140195	0.00		7.77	0.000
FRCAC40	-0.09036819	0.00	0.02	0.20	3292.27
	0.1117617	-0.00		8.49	0.000
FTSEMIB	-0.08238858	0.00	0.01	0.11	4577.44
	0.114905	-0.00		9.49	0.000
DJES50I	-0.07879843	0.00	0.02	0.16	2481.17
	0.1100186	-0.00		7.77	0.000
SPCOMP	-0.0903498	0.00	0.01	0.12	7271.25
	0.1158004	-0.00		11.17	0.000
NASA100	-0.1051949	0.00	0.02	0.46	2951.31
	0.187714	0.00		8.13	0.000
JAPDOWA	-0.1140637	0.00	0.02	-0.10	5350.46
	0.141503	-0.00		10.01	0.000
KORCOMP	-0.1201879	0.00	0.02	-0.32	1792.19
	0.1194567	0.00		7.01	0.000

Continued on next page

Mnemonic	Min Max	Median Mean	SD	Skewness Kurtosis	Jarque-Bera p-value
BBGBP12	-0.1905788	-0.00	0.01	-4.86	1648238.73
	0.057722	-0.00		125.75	0.000
ECEUR3M	-0.14	0.00	0.01	0.21	179864.60
	0.1627907	-0.00		43.67	0.000
UKyc	-0.6134	-0.00	0.07	2.21	224631.98
	1.2751	0.00		48.24	0.000
GERyc	-0.3781	-0.00	0.05	0.24	2029.41
	0.3672	0.00		7.29	0.000
FRyc	-0.4209	-0.00	0.05	0.08	2795.06
	0.3602	0.00		8.07	0.000
ITyc	-0.3926	-0.00	0.05	0.51	2888.93
	0.3667	0.00		8.05	0.000
USyc	-0.7181	-0.00	0.08	0.19	15501.22
	0.7939	0.00		14.94	0.000
JAPyc	-0.3688	0.00	0.04	0.05	6344.93
	0.2904	-0.00		10.64	0.000
USDOLLR	-0.03842446	0.00	0.01	0.00	3145.20
	0.04579271	0.00		8.38	0.000
SWISFUS	-0.02848304	0.00	0.01	0.22	346.90
	0.04322024	0.00		4.73	0.000
USEURSP	-0.03770979	0.00	0.01	0.28	1188.70
	0.04729214	0.00		6.26	0.000
JAPAYEUSD	-0.04505151	0.00	0.01	-0.55	1465.73
	0.02567126	-0.00		6.51	0.000
GOLDBLN	-0.06894229	0.00	0.01	0.08	3353.14
	0.07661333	0.00		8.55	0.000
OILBREN	-0.1274272	0.00	0.02	0.11	707.16
	0.1445876	0.00		5.54	0.000
NYFECRB	-0.05728754	0.00	0.01	-0.32	2448.47
	0.05152885	0.00		7.70	0.000
BALTICF	-0.1126648	0.00	0.02	0.21	15114.31
	0.1463415	0.00		14.78	0.000

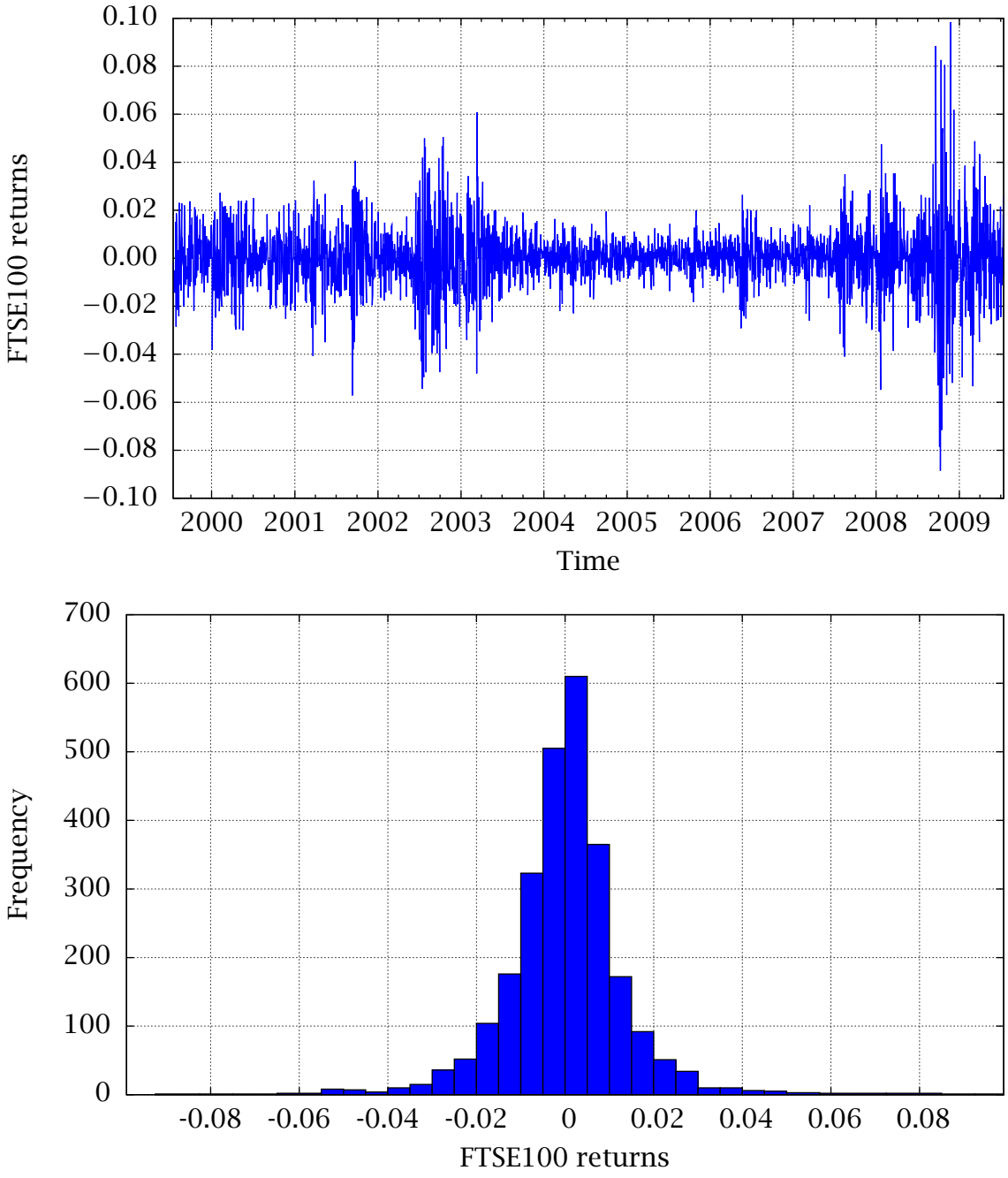


Figure 4.7: The FTSE 100 equity index returns from July 1999 to July 2009 and corresponding distribution.

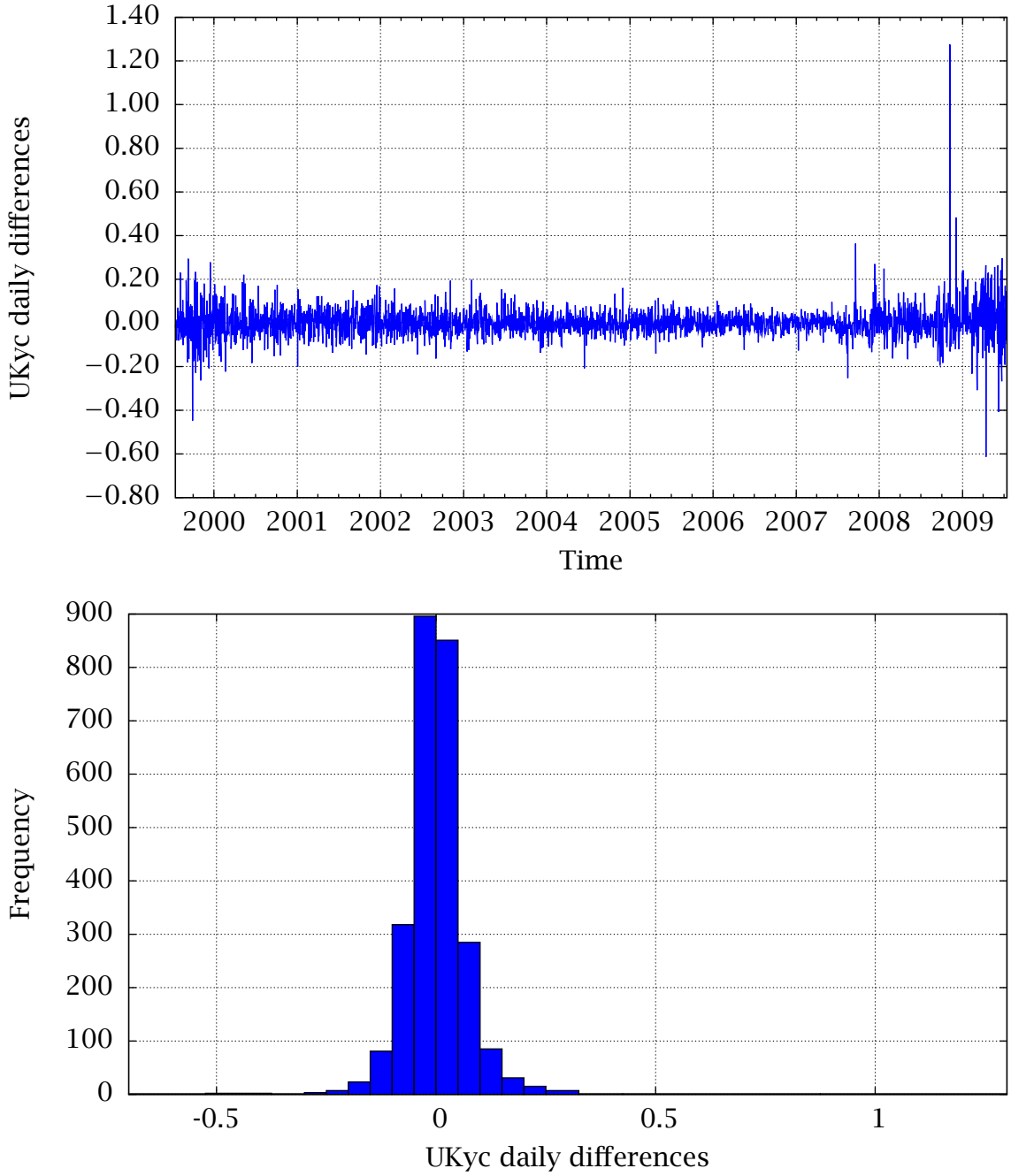


Figure 4.8: The UK yield curve daily differences and corresponding distribution.

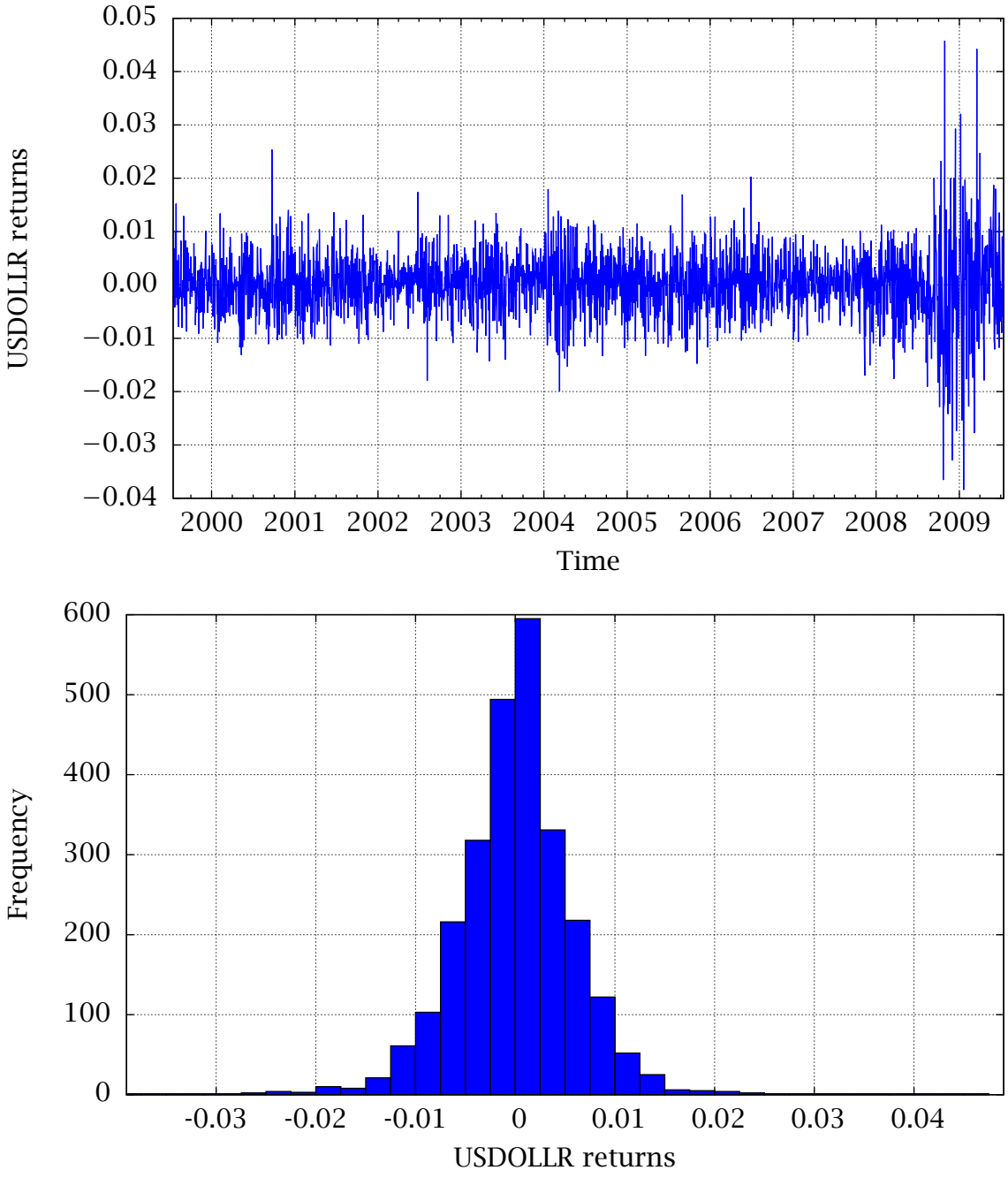


Figure 4.9: The GBP|USD exchange rate returns (Datastream USDOLLR), and distribution.

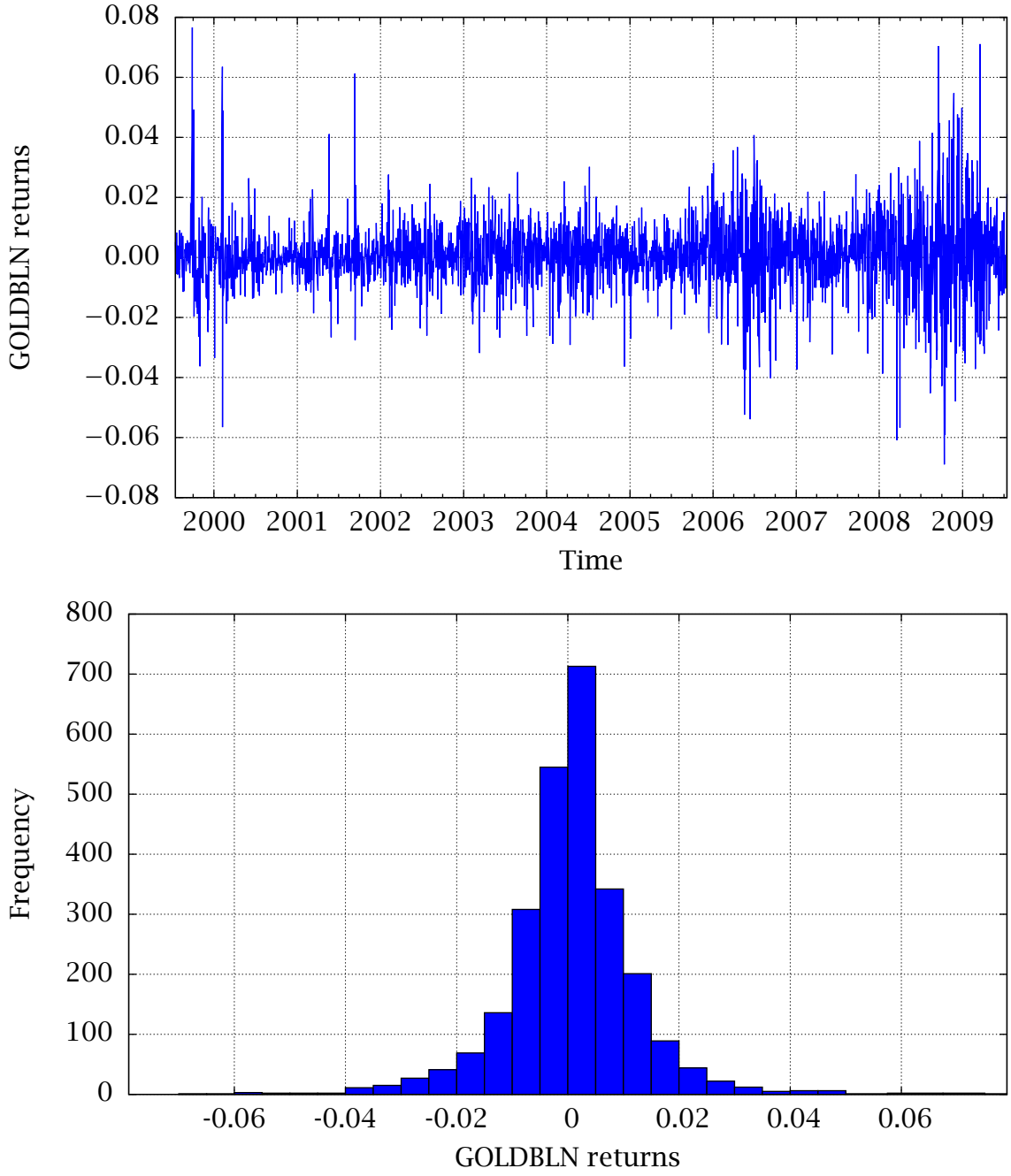


Figure 4.10: The Gold Bullion daily returns and distribution.

4.5.4 Stationarity and Autocorrelation Analysis

One analyzes whether the transformed series are stationary. I.e., one answers the question, if one can reasonably assume the probability distribution to be constant over time. Especially one wants to know, if mean and variance are unchanged. The augmented Dickey-Fuller (ADF) statistic, tests for the presence of a unit root, see [121]. The ADF statistic is a number smaller than zero. The smaller it is, the less likely the time series is to have a unit root. A unit root series is non-stationary. Therefore, if one can reject the hypothesis of a unit root one has a stationary process. Similarly the Phillips-Perron test is used to test for stationarity, see [267]. Table 4.9 on the next page shows that all return series are stationary at the 99 % significance level for both tests. One requires stationarity, especially for training of neural networks, to avoid having to move the weights all together in one direction during training.

Additionally one is interested in discovering autocorrelation in the data. Autocorrelation signifies that lags of the time series are correlated with one another. Periodic signals are, e.g., strongly autocorrelated. For the present analysis one is interested in autocorrelation because one can hope that the model will exploit it. The advantage of having a model with a memory is that it can — in principle — reproduce almost every kind of periodic signal. One analyzes two kinds of autocorrelation:

- standard autocorrelation, or acf, which measures correlation between lags r_t and r_{t-n}.
- partial autocorrelation, or pacf, which measures correlation between lags r_t and r_{t-n}, but additionally removes all correlation for lags in between, $r_{t-1}, \ldots, r_{t-n+1}$.

Partial autocorrelation indicates the pure autocorrelation between two lags. The Ljung-Box statistic provides a measure to test for autocorrelation, [230]. The higher the resulting χ^2 value the higher the probability that lags are indeed correlated. The corresponding p-value indicates the probability that the lags are *uncorrelated*. Table 4.10 on page 151 shows autocorrelation functions up to lag 10 and the corresponding Ljung-Box statistic for every lag. One notices several effects:

- As a general rule both autocorrelations decrease with increasing lags. This is quite intuitive: one expects *older* values to have *less* impact on today. An

Table 4.9: Stationarity analysis for returns: all transformed return series are stationary. This is confirmed by both the augmented Dickey-Fuller (ADF) test statistic and the Phillips-Perron (PP) test statistic at the 99 % significance level. One therefore assumes that the distribution of returns does not change over time for the sample in question. Mean and variance are constant, a useful property, especially when using this data as basis for neural network training. Most other statistical models also require stationarity.

Mnemonic	ADF	p-value	PP	p-value
FTSE100	-14.51	0.010	-55.08	0.010
DAXINDX	-13.66	0.010	-53.27	0.010
FRCAC40	-14.16	0.010	-54.00	0.010
FTSEMIB	-13.17	0.010	-52.08	0.010
DJES50I	-14.14	0.010	-53.98	0.010
SPCOMP	-13.82	0.010	-56.52	0.010
NASA100	-12.89	0.010	-55.84	0.010
JAPDOWA	-13.79	0.010	-51.94	0.010
KORCOMP	-14.46	0.010	-50.23	0.010
BBGBP12	-10.15	0.010	-44.59	0.010
ECEUR3M	-9.11	0.010	-69.35	0.010
UKyc	-13.68	0.010	-58.84	0.010
GERyc	-13.17	0.010	-54.00	0.010
FRyc	-13.10	0.010	-56.06	0.010
ITyc	-12.79	0.010	-53.65	0.010
USyc	-11.52	0.010	-54.66	0.010
JAPyc	-13.35	0.010	-59.54	0.010
USDOLLR	-13.78	0.010	-47.98	0.010
SWISFUS	-13.74	0.010	-54.53	0.010
USEURSP	-13.36	0.010	-49.85	0.010
JAPAYEUSD	-12.97	0.010	-51.98	0.010
GOLDBLN	-14.35	0.010	-51.16	0.010
OILBREN	-12.44	0.010	-51.81	0.010
NYFECRB	-13.81	0.010	-50.28	0.010
BALTICF	-9.14	0.010	-14.13	0.010

exception is data which includes some kind of cycle. In this case there can be some lags in the past which exhibit much higher autocorrelation than recent lags. The only time series which shows such a feature is the Baltic Dry Index. With pacf(7) = 0.12 being much higher than the surrounding pacf one finds a week kind of cycling behavior.

- One cannot verify the hypothesis that the lags are uncorrelated. This is a promising sign for future model building.
- Generally correlation is confirmed at the 99 percent confidence level. There are however exceptions: FTSE MIB, Nikkei, Kospi, EUR|USD, Gold Bullion, oil, and CRB index show high p-values, mostly for recent lags. Although they never reach even the 95 percent confidence level for confirming a random relation one should be aware that the inclusion of these lags in a purely autoregressive model might not be successful.

To summarize the analysis shows that the data is suitable for model building with neural networks: it is stationary and the present autocorrelation should help during the training phase.

Table 4.10: Autocorrelation analysis for the dataset. acf and pacf represent the appropriately lagged autocorrelation and partial autocorrelation function. The χ^2 statistic and corresponding p-value result from the Ljung-Box test. Low p-values indicate that the null hypothesis that the lags are *uncorrelated* cannot be verified.

lag	1	2	3	4	5	6	7	8	9	10
										FTSE100
acf	-0.06	-0.06	-0.09	0.09	-0.06	-0.05	0.03	0.07	-0.01	-0.02
pacf	-0.06	-0.06	-0.10	0.08	-0.06	-0.05	0.03	0.05	0.00	-0.01
χ^2	10.55	20.13	41.70	64.74	75.08	80.98	82.92	94.82	94.98	96.22
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										DAXINDX
acf	-0.04	-0.01	-0.03	0.04	-0.05	-0.01	-0.00	0.03	-0.01	-0.01
pacf	-0.04	-0.01	-0.03	0.04	-0.04	-0.02	-0.00	0.03	-0.01	-0.01
χ^2	4.51	4.85	6.87	11.76	17.57	18.09	18.10	20.83	21.20	21.63
p-val.	0.03	0.09	0.08	0.02	0.00	0.01	0.01	0.01	0.01	0.02

Continued on next page

lag	1	2	3	4	5	6	7	8	9	10
										FRCAC40
acf	-0.05	-0.04	-0.07	0.06	-0.07	-0.04	0.01	0.06	-0.04	-0.02
pacf	-0.05	-0.04	-0.07	0.05	-0.07	-0.04	0.01	0.04	-0.03	-0.02
χ^2	5.57	9.83	22.58	30.70	42.93	46.20	46.36	54.37	58.43	59.51
p-val.	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										FTSEMIB
acf	-0.02	-0.01	-0.03	0.12	-0.08	-0.01	0.00	0.08	-0.03	-0.01
pacf	-0.02	-0.01	-0.03	0.11	-0.08	-0.01	0.01	0.06	-0.01	-0.01
χ^2	1.18	1.41	4.29	38.93	56.52	56.76	56.78	73.52	76.41	76.74
p-val.	0.28	0.49	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										DJES50I
acf	-0.05	-0.03	-0.07	0.06	-0.06	-0.03	-0.00	0.06	-0.03	-0.01
pacf	-0.05	-0.04	-0.07	0.06	-0.06	-0.04	-0.01	0.05	-0.03	-0.01
χ^2	6.15	9.13	20.48	31.30	41.94	44.71	44.77	54.36	57.46	57.84
p-val.	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										SPCOMP
acf	-0.09	-0.07	0.03	-0.01	-0.03	-0.01	-0.03	0.04	-0.02	0.02
pacf	-0.09	-0.08	0.01	-0.01	-0.03	-0.02	-0.04	0.03	-0.02	0.02
χ^2	21.07	34.02	36.03	36.35	38.79	38.94	41.94	46.50	47.74	48.44
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										NASA100
acf	-0.08	-0.06	-0.01	0.00	-0.02	-0.01	0.02	0.01	-0.01	-0.00
pacf	-0.08	-0.07	-0.02	-0.01	-0.03	-0.02	0.01	0.01	-0.01	-0.00
χ^2	16.35	27.18	27.26	27.26	28.75	29.06	29.65	30.07	30.50	30.50
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										JAPDOWA
acf	-0.01	-0.06	-0.03	0.01	0.02	-0.03	0.00	-0.01	-0.04	-0.01
pacf	-0.01	-0.06	-0.04	0.01	0.01	-0.03	0.01	-0.01	-0.04	-0.01
χ^2	0.45	8.85	12.03	12.31	13.23	16.05	16.09	16.26	19.83	19.93
p-val.	0.50	0.01	0.01	0.02	0.02	0.01	0.02	0.04	0.02	0.03

Continued on next page

lag	1	2	3	4	5	6	7	8	9	10
										KORCOMP
acf	0.02	-0.03	-0.00	-0.02	-0.03	-0.01	-0.02	0.01	-0.03	-0.00
pacf	0.02	-0.03	-0.00	-0.02	-0.03	-0.01	-0.02	0.01	-0.03	-0.00
χ^2	0.77	2.66	2.70	3.31	5.44	5.61	6.25	6.55	8.32	8.32
p-val.	0.38	0.26	0.44	0.51	0.37	0.47	0.51	0.59	0.50	0.60
										BBGBP12
acf	0.18	0.12	0.11	0.12	0.06	0.10	0.07	0.07	0.05	0.03
pacf	0.18	0.09	0.07	0.08	0.02	0.07	0.03	0.03	0.01	-0.01
χ^2	88.91	126.80	155.63	191.50	201.84	227.67	241.96	254.86	261.51	264.12
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										ECEUR3M
acf	-0.34	0.18	-0.13	0.17	-0.02	0.10	-0.05	0.04	0.14	-0.01
pacf	-0.34	0.08	-0.06	0.11	0.09	0.11	0.02	-0.00	0.17	0.06
χ^2	293.73	379.30	424.32	497.21	497.88	526.29	533.07	536.67	587.83	588.04
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										UKyc
acf	-0.14	-0.01	0.04	0.03	0.04	-0.03	0.00	0.01	-0.00	0.05
pacf	-0.14	-0.03	0.03	0.04	0.05	-0.02	-0.01	0.01	0.00	0.05
χ^2	54.70	54.90	58.23	60.13	63.76	66.71	66.72	67.19	67.19	73.92
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										GERyc
acf	-0.06	0.03	0.02	0.03	0.00	0.03	0.01	0.00	0.02	-0.05
pacf	-0.06	0.03	0.02	0.03	0.01	0.03	0.02	0.00	0.02	-0.05
χ^2	8.24	10.31	10.94	12.87	12.92	16.04	16.54	16.58	18.15	23.86
p-val.	0.00	0.01	0.01	0.01	0.02	0.01	0.02	0.03	0.03	0.01
										FRyc
acf	-0.10	0.02	0.02	0.01	-0.01	0.06	0.01	0.02	0.01	-0.05
pacf	-0.10	0.01	0.02	0.02	-0.01	0.05	0.02	0.02	0.01	-0.05
χ^2	23.71	24.94	25.90	26.36	26.94	35.45	35.84	36.67	37.03	42.64
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Continued on next page

lag	1	2	3	4	5	6	7	8	9	10
										ITyc
acf	-0.05	0.04	0.01	0.03	0.01	0.03	0.00	0.02	0.02	-0.03
pacf	-0.05	0.04	0.02	0.03	0.01	0.03	0.00	0.01	0.02	-0.03
χ^2	5.99	10.10	10.44	12.84	12.92	15.84	15.84	16.44	17.31	19.61
p-val.	0.01	0.01	0.02	0.01	0.02	0.01	0.03	0.04	0.04	0.03
										USyc
acf	-0.08	-0.07	0.05	0.01	0.08	0.02	0.10	0.01	0.01	-0.01
pacf	-0.08	-0.07	0.04	0.02	0.09	0.04	0.11	0.02	0.03	-0.02
χ^2	15.05	26.66	32.17	32.64	49.56	50.99	75.76	75.90	76.48	76.74
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										JAPyc
acf	-0.14	-0.02	-0.05	0.04	0.01	-0.00	-0.03	-0.01	-0.01	-0.04
pacf	-0.14	-0.04	-0.06	0.03	0.02	0.00	-0.03	-0.02	-0.02	-0.05
χ^2	51.44	52.69	60.03	65.09	65.46	65.46	68.01	68.25	68.82	72.37
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
										USDOLLR
acf	0.06	0.00	-0.05	-0.00	-0.03	0.00	0.02	0.01	-0.03	-0.06
pacf	0.06	0.00	-0.05	0.00	-0.03	0.00	0.02	0.01	-0.03	-0.05
χ^2	9.37	9.41	14.90	14.91	16.86	16.88	18.25	18.57	21.16	29.42
p-val.	0.00	0.01	0.00	0.00	0.00	0.01	0.01	0.02	0.01	0.00
										SWISFUS
acf	-0.07	0.01	-0.01	0.01	0.02	0.01	0.00	0.03	-0.03	0.02
pacf	-0.07	0.00	-0.01	0.01	0.02	0.01	0.01	0.03	-0.03	0.02
χ^2	11.86	11.98	12.50	12.63	13.48	13.84	13.89	16.02	18.66	19.72
p-val.	0.00	0.00	0.01	0.01	0.02	0.03	0.05	0.04	0.03	0.03
										USEURSP
acf	0.02	-0.02	0.00	0.05	-0.01	-0.04	0.03	0.04	-0.04	-0.03
pacf	0.02	-0.02	0.00	0.05	-0.01	-0.04	0.03	0.04	-0.03	-0.02
χ^2	1.44	2.35	2.38	7.90	8.24	13.00	15.28	20.05	23.26	25.06
p-val.	0.23	0.31	0.50	0.10	0.14	0.04	0.03	0.01	0.01	0.01

Continued on next page

lag	1	2	3	4	5	6	7	8	9	10
										JAPAYEUSD
acf	-0.02	-0.02	-0.03	-0.00	0.00	-0.03	0.03	-0.01	0.02	0.01
pacf	-0.02	-0.02	-0.03	-0.00	-0.00	-0.03	0.03	-0.01	0.02	0.01
χ^2	0.78	1.53	3.59	3.59	3.59	5.23	8.03	8.32	9.15	9.40
p-val.	0.38	0.47	0.31	0.46	0.61	0.51	0.33	0.40	0.42	0.49
										GOLDBLN
acf	-0.00	0.01	0.01	0.03	0.04	-0.05	-0.03	0.00	-0.00	-0.02
pacf	-0.00	0.01	0.01	0.03	0.04	-0.05	-0.03	0.00	-0.00	-0.02
χ^2	0.02	0.44	1.00	2.72	6.31	11.99	14.11	14.15	14.19	15.54
p-val.	0.90	0.80	0.80	0.61	0.28	0.06	0.05	0.08	0.12	0.11
										OILBREN
acf	-0.01	0.00	0.02	0.02	-0.01	-0.05	0.03	-0.01	-0.01	0.02
pacf	-0.01	0.00	0.02	0.02	-0.01	-0.05	0.03	-0.01	-0.01	0.02
χ^2	0.58	0.59	2.13	3.00	3.12	8.62	11.62	11.72	12.13	12.81
p-val.	0.44	0.74	0.55	0.56	0.68	0.20	0.11	0.16	0.21	0.23
										NYFECRB
acf	0.02	0.00	0.04	0.03	-0.03	-0.03	-0.04	0.01	0.01	0.00
pacf	0.02	0.00	0.04	0.02	-0.03	-0.03	-0.04	0.01	0.01	0.01
χ^2	0.59	0.60	4.64	6.36	8.05	9.89	13.48	13.56	13.71	13.76
p-val.	0.44	0.74	0.20	0.17	0.15	0.13	0.06	0.09	0.13	0.18
										BALTICF
acf	0.83	0.62	0.43	0.29	0.20	0.15	0.15	0.18	0.22	0.22
pacf	0.83	-0.25	0.00	-0.00	0.01	0.04	0.12	0.06	0.02	-0.01
χ^2	1813.4	2806.3	3295.3	3522.8	3627.4	3684.9	3745.1	3834.5	3956.3	4086.1
p-val.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

4.6 Modeling Market Value at Risk

Value at risk is a crucial regulatory number. It influences the amount of capital a financial institution has to keep as margin of safety against adverse movements in the portfolio. Value at risk is typically used at the 99 percent confidence interval and a 10 day time span. If accurate, the 99 percent 10 day VaR indicates which loss will not be exceeded with a probability of 99 percent within the next 10 days, see [188]. Value at risk is the most common risk measure among financial institutions, traders and regulators alike, see also [278].

Here, the author will focus on finding a measure which tries to reproduce accurately the *real* encountered risk — not some conservative value. Conservative models like, e.g., historical simulation which the author will benchmark against, are necessary in their own right to provide a margin of safety, not least to make the regulator happy. Banks, however, are interested in the minimum value of the portfolio that will likely occur during the next 10 days. Think about it: which kind of model makes the most sense from the perspective of a financial institution involved in investment or trading? Is it a model which obliges the institution to hold exorbitant amount of risk capital in reserve? Or is it the model which adequately represents the portfolio over the next ten days? — At the peril that the occasional violation may occur.

From the latter point of view the story of modeling a *realistic* VaR boils down to forecasting the lowest portfolio value within the next ten days. A multi step forecast offers the additional advantage that one gets an impression of the portfolio moves not just the worst expected value. Using an expert topology gives us not only a *single* value or a single path for that matter but a *distribution* of values. One can therefore peek above the threshold, say 95 or 99 percent, and see how bad things might really get. This alleviates one of the deficiencies of standard VaR. It only answers the question: what will most likely *not* happen to the portfolio within the next ten days. It does not however answer the question: *if* the worst happens, how bad will it be? Note that this issue is addressed by conditional VaR, see [188]. However, simple VaR remains the most observed measure.

As one is training neural networks on historical data one will use *historical simulation* as benchmark. Historical simulation is a very simple yet often used and conservative way of estimating regulatory VaR, see [188]. It works as follows:

- Decide for a look back period l, here $l = 440$ days.

- Compute daily returns.
- For every day compute the forward worst 10 day returns.
- Order these returns by magnitude. Keep in mind: the best possible worst return is 100 percent of portfolio value as the present day is included in the computation.
- Take an appropriate percentile of the 10 day returns, e.g., the return at the worst 1 percent for a 99 percent VaR.

A simple and intuitive measure to compare value at risk models is the sum of squared deviations *SSD* from the forecasted value. This gives us an appreciation of how close one estimates the minimal value of the portfolio. From a regulatory point of view one might be tempted to only penalize *violations* of the VaR by the portfolio. However, one wants a measure of how closely the worst portfolio is modeled. If one assumes too low a value this costs a financial institution money in lost opportunities. It makes sense to choose a *symmetric* measure.

In the following the author presents a detailed analysis of modeled market VaR for three assets: FTSE 100, GBP|USD exchange rate and Gold Bullion. This time the author omits the UK yield curve because one would have to construct a derived portfolio measuring the effect of yield curve changes. This could be realized, e.g., with a swap. However it would unnecessarily complicate the exposure without adding something substantially important.

First look at the *distributions* of worst returns from the expert topology. These are shown on figures 4.11 on page 161 for the FTSE 100 index, figure 4.12 for GBP|USD exchange rate and figure 4.13 for the Gold Bullion. The figures show forecasted worst asset values for a time span of 10 days. The asset prices are rebased at 1.0 every day to allow comparisons. Obviously, the *worst* portfolio value can never be greater than 1.0, as one always include the present day in the computation: even if the portfolio rises continually during the next ten days the *worst* value is still 1.0. For every day the forecasts are reordered so that the worst is at position 1 and the best at position 300. Remember, one has an expert topology with 300 committee members. The three figures give us a feeling for the forecasts:

- The forecasted worst values are different among the assets under consideration: e.g., the GBP|USD exchange rate is forecasted to have better portfolio

values over 10 days than the other assets. The Gold Bullion on the other hand is forecasted to be more volatile.

- The worst forecasts are clustered in a small area of the tail of the distribution. This indicates that it is not appropriate to simply take the average of the forecasts. It is better to choose an appropriate percentile. In the following the author will use 1 percent, i.e., the author will discard the worst 3 networks of 300 networks.

- The forecasts are very irregular. It is advisable to smooth them. The analysis will also consider forecasts smoothed by the look forward period, i.e., a 10 day simple moving average.

FTSE 100 Index The author starts with an analysis of VaR for the short time span of 110 days. That is one quarter of the training and validation data and the typical validity period of a quantitative model. Figure 4.14 on page 162 shows the result for the FTSE 100 Index. The author explains this figure in detail as it is complex. The target is the bold blue line. This is an a posteriori calculation of the worst return that indeed occurred in the following 10 days. Deviations are measured against this target. Then one has the red line which shows the suggested expected portfolio worst return obtained by historical simulation over the past 440 days. This is the benchmark one tries to beat. Every VaR forecast that is closer to the realized values is considered better.

One notes that historical VaR stays almost constant during the whole time span of 110 days. A more flexible model should therefore be able to improve on this. The green line shows the forecasted VaR by taking the appropriate lowest percentile of all networks as described above. One sees that it roughly follows the target and is generally closer to it than historical VaR. However, as already mentioned the forecast is very jigged. The purple line is a smoothed version of the forecast and exhibits a nicer behavior. Especially, one avoids violations which occur using only the pure forecast but which do not occur with historical VaR.

From table 4.11 on page 167 one gathers that the deviation of smoothed VaR from the target is only about 77 percent of historical VaR. Figure 4.17 on page 164 shows the development of the cumulated squared deviations. At the beginning the values move closely together with historical VaR being even a little bit better. Around day 520 the lines decidedly separate and historical

VaR gets worse. Although the smoothed forecast looks better the difference compared to the pure forecast is not very important: both still move together. This is also reflected in figure 4.20 on page 165 which shows the mean SSD. Interestingly the model manages to beat historical VaR over the prolonged time span of more than 8 years, see table 4.11.

US Dollar to Great British Pound Figure 4.15 on page 163 shows VaR for GBP|USD exchange rate. One notes that VaR is mostly confined to a small band of less than 2 percent. Or, equivalently, the worst portfolio value stays above a normalized value of 0.98. During this time the forecast and the smoothed version manages well to model VaR and delivers closer estimates than historical VaR. On day 587 however, the forecast and smoothed model cause a small violation. Historical VaR on the other hand produces no violation. As one is considering a 99 percent confidence interval one is not surprised to find one violation in a time span of 110 days. Figure 4.18 on page 164 shows how well the smoothed forecast actually performs in terms of accuracy.

During the entire time span the smoothed forecasted VaR models the realized VaR more closely than historical VaR. Whereas the simple forecasts perform worse at the beginning until day 480 one sees that from that day on the accuracy gap between historical VaR on the one hand and forecasted VaR on the other hand widens considerably. The forecasts are able to better adapt to the smaller range of the GBP|USD exchange rate which begins on day 495. Figure 4.21 on page 166 confirms the findings: the mean deviation is significantly smaller for both forecast models. Table 4.11 shows that the mean SSD for the smoothed model is only

$$\frac{0.00070}{0.00107} = 0.65$$

or 65 percent of the historical simulation. For the entire time span of more than 8 years one gets a ratio of 71 percent.

Gold Bullion Figure 4.16 on page 163 compares the respective VaR. Almost always the smoothed forecast models VaR more accurately than historical simulation. One notes however two violations. The first right at the beginning, the second around day 482. One sees that historical VaR acts almost like a resistance for the smoothed forecast which manages to adapt accurately to the less volatile period starting around day 490.

Figures 4.19 on page 165 and 4.22 on page 166 show impressively how well the forecasts work. For the smoothed version one even has only 44 percent deviation compared to historical VaR. Interestingly, the model even manages to *improve* this ratio to 31 percent for the 8 year period. The Gold Bullion chart, figure 4.5 on page 138, shows the cause for this improvement. The gold price spikes in the first half of 2006 and then has a very volatile period starting in mid 2007. This marks the beginning of the credit crisis. Such volatile periods are the times where the forecasts really shine. Historical VaR produces über-conservative estimations then. However, one might be tempted to further optimize smoothing period and percentile of the distribution to avoid violations in the forecast model. This is not the scope of the analysis.

The analysis shows that an expert topology of shared layer perceptrons is capable of beating the benchmark VaR. The smoothed version of the forecast manages to do this *always* for the time span of 110 days. But even when considering the prolonged out of sample time span of more than 8 years table 4.11 on page 167 shows that the smoothed forecast performs worse than historical VaR only in 5 out of 17 cases. In all other cases the forecast is often considerably better.

Consider also that no additional optimization has been conducted. E.g., one might optimize the smoothing period or the percentile over the validation data. Doing this independently for each asset might considerably improve forecasts for individual assets. However, this is beyond the scope of the present work and would be against the spirit of a coherent market forecast: The author prefers to use a model which demonstrably works well for *all* assets without curve fitting to a specific one. In quantitative asset management there are, of course, good reasons to do just that. One then needs to recalibrate the parameters often. Compared to a forecast using a single multi layer perceptron — or any model which offers just *one* number — key benefits of the presented method are:

- The multi step forecasts offers a complete view on portfolio value path.
- A *single* model is used for different confidence levels. Once the expert topology is trained one gets every percentile of the distribution *for free*: just choose the point in the distribution that happens to be suitable.
- A *single* model is used for all assets. If it works simultaneously over a broad range of assets one will be more confident to use it. The shared layer perceptron works well for *all* inputs *by design*.

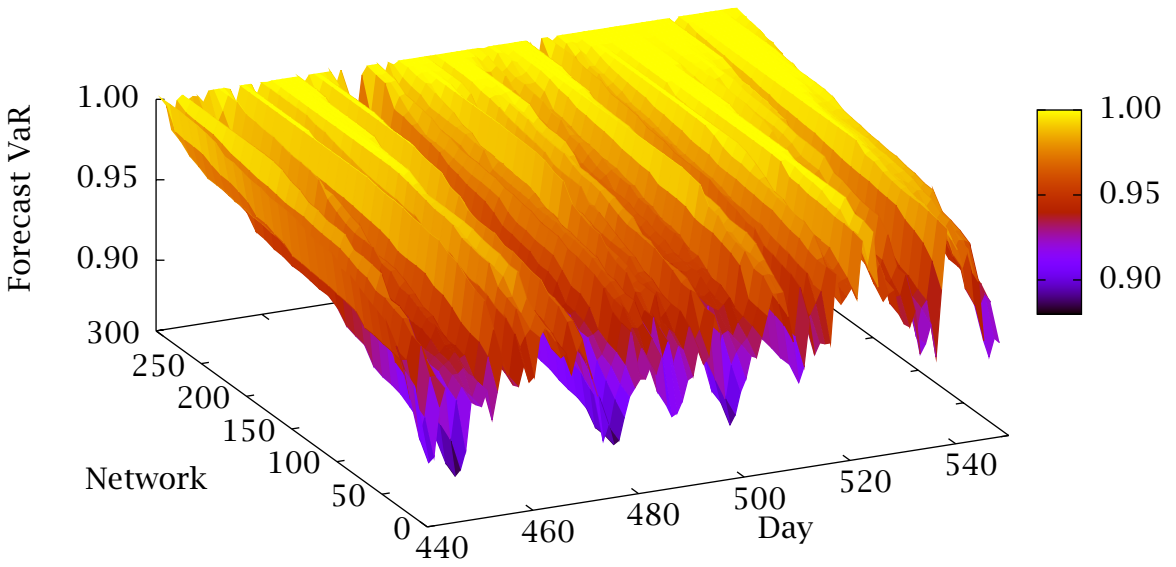


Figure 4.11: Worst 10 day returns for the FTSE 100, distribution of expert topology.

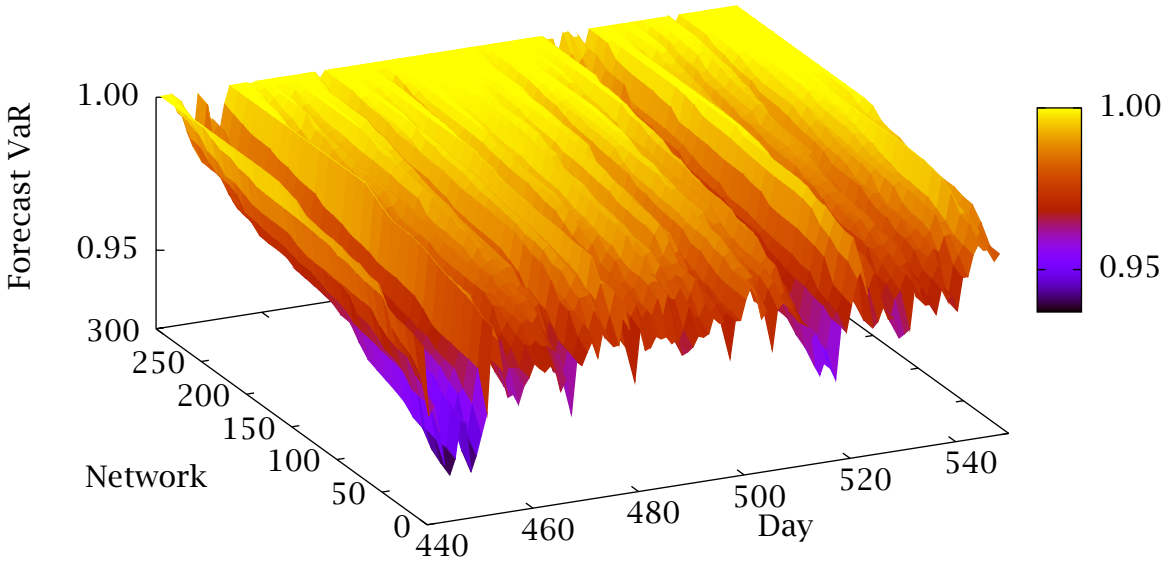


Figure 4.12: Worst 10 day returns for the GBP|USD exchange rate, distribution of expert topology.

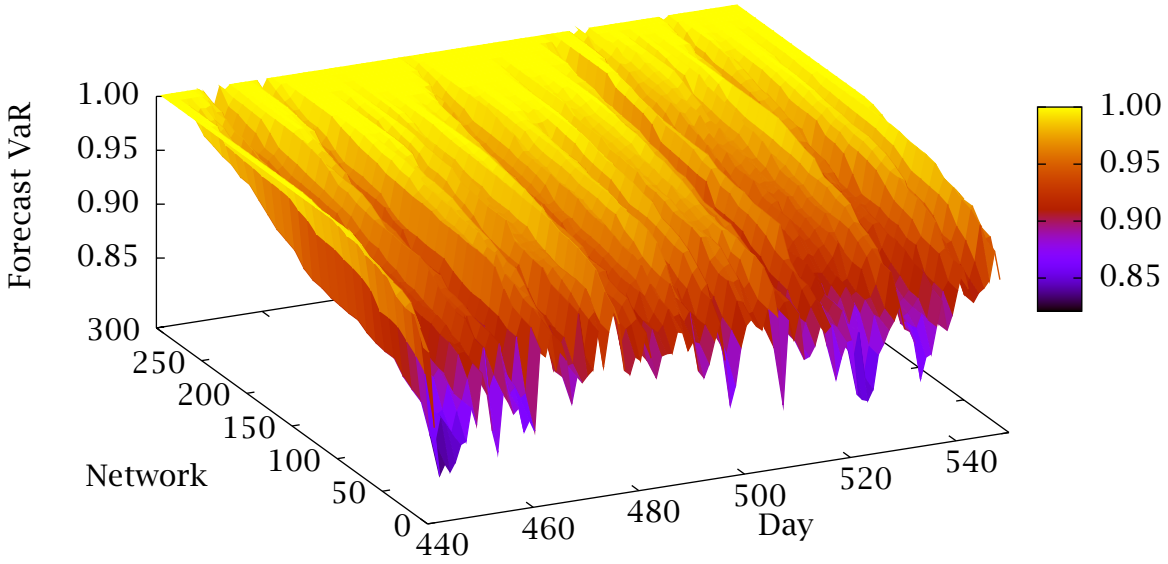


Figure 4.13: Worst 10 day returns for the Gold Bullion, distribution of expert topology.

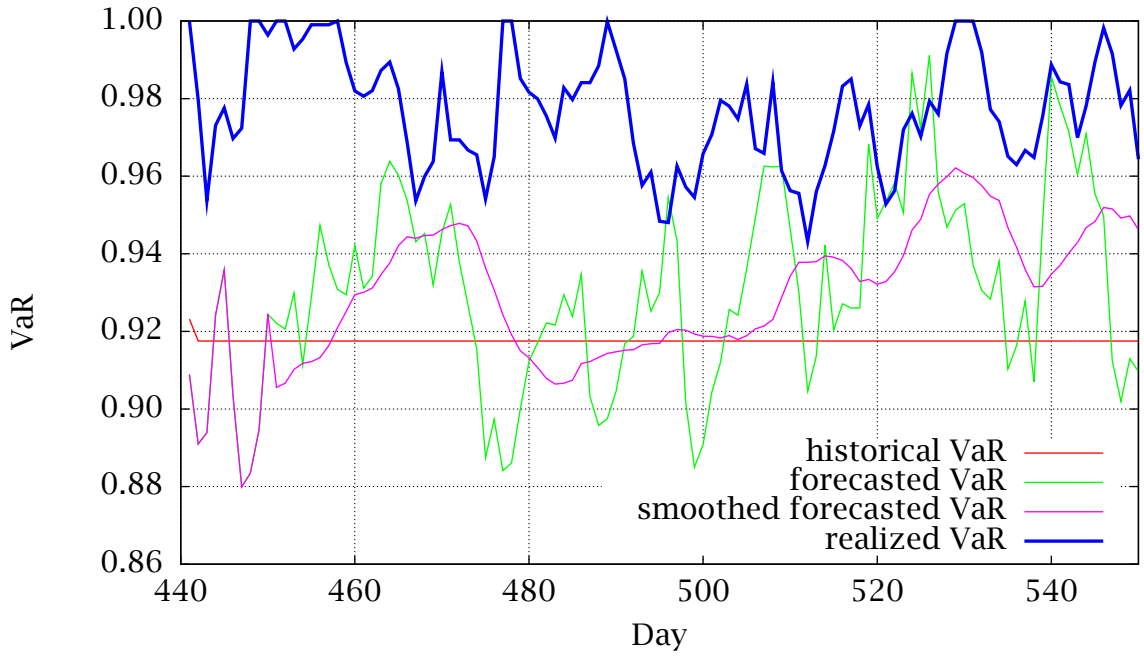


Figure 4.14: Comparison of historical, forecasted and smoothed forecasted VaR for the FTSE 100.

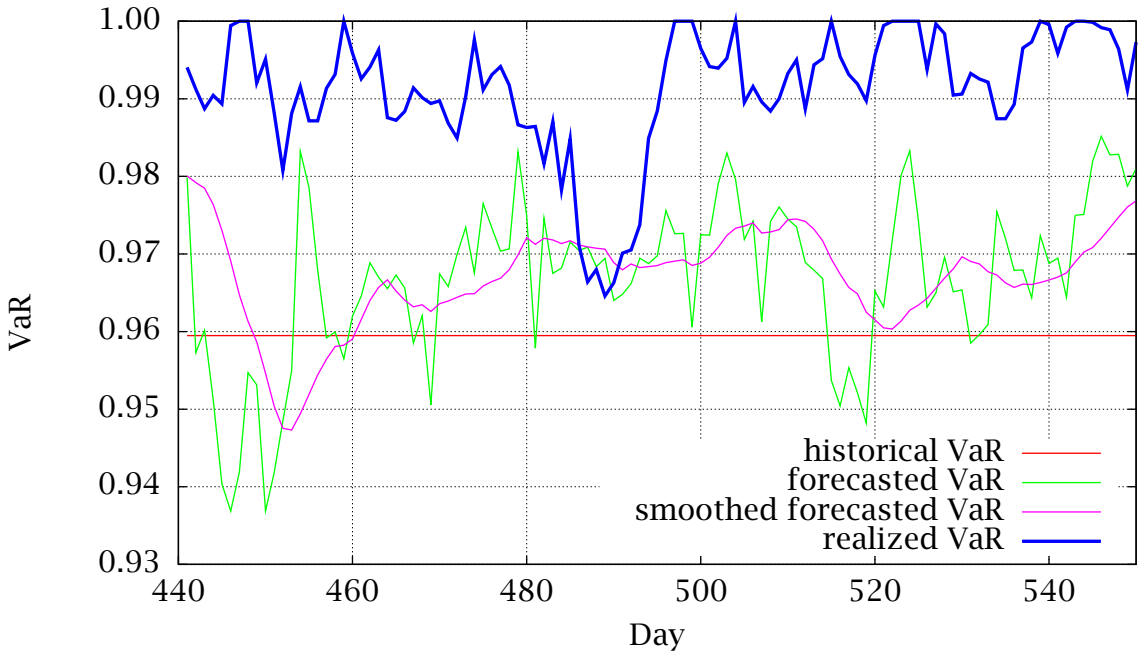


Figure 4.15: Comparison of historical, forecasted and smoothed forecasted VaR for the GBP|USD exchange rate.

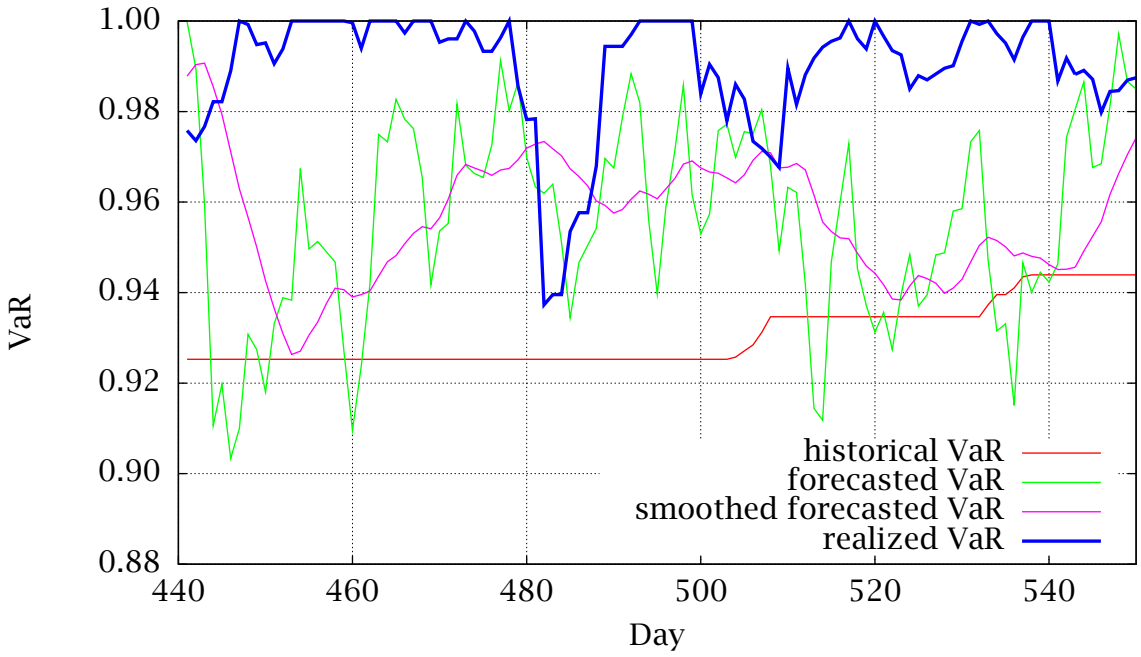


Figure 4.16: Comparison of historical, forecasted and smoothed forecasted VaR for the Gold Bullion.

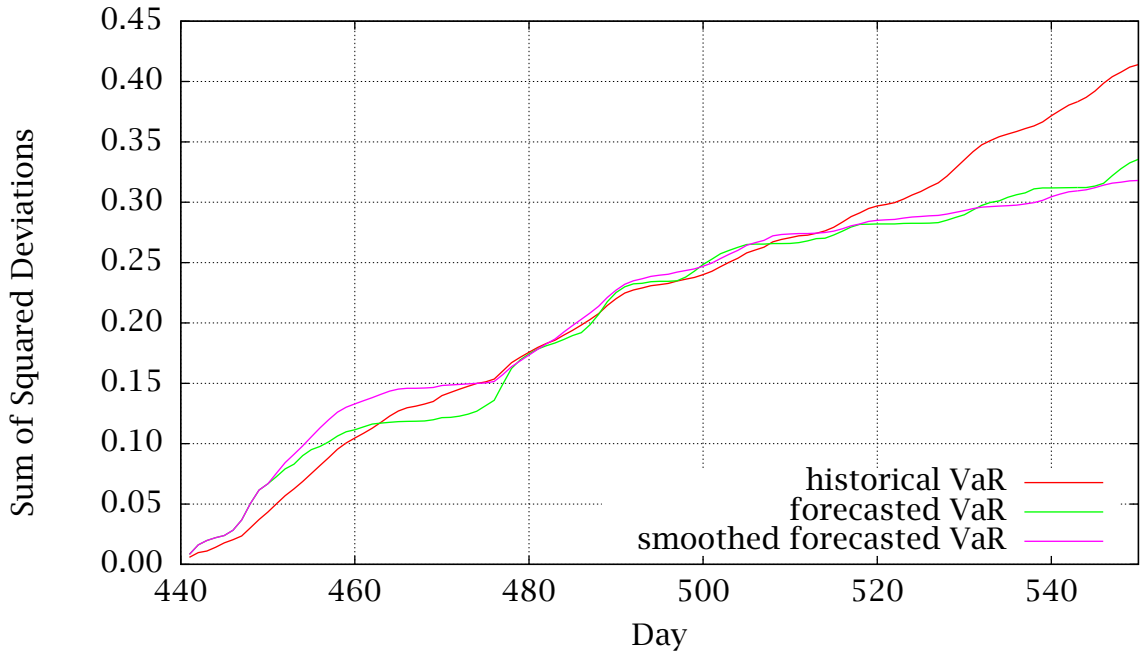


Figure 4.17: Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the FTSE 100.

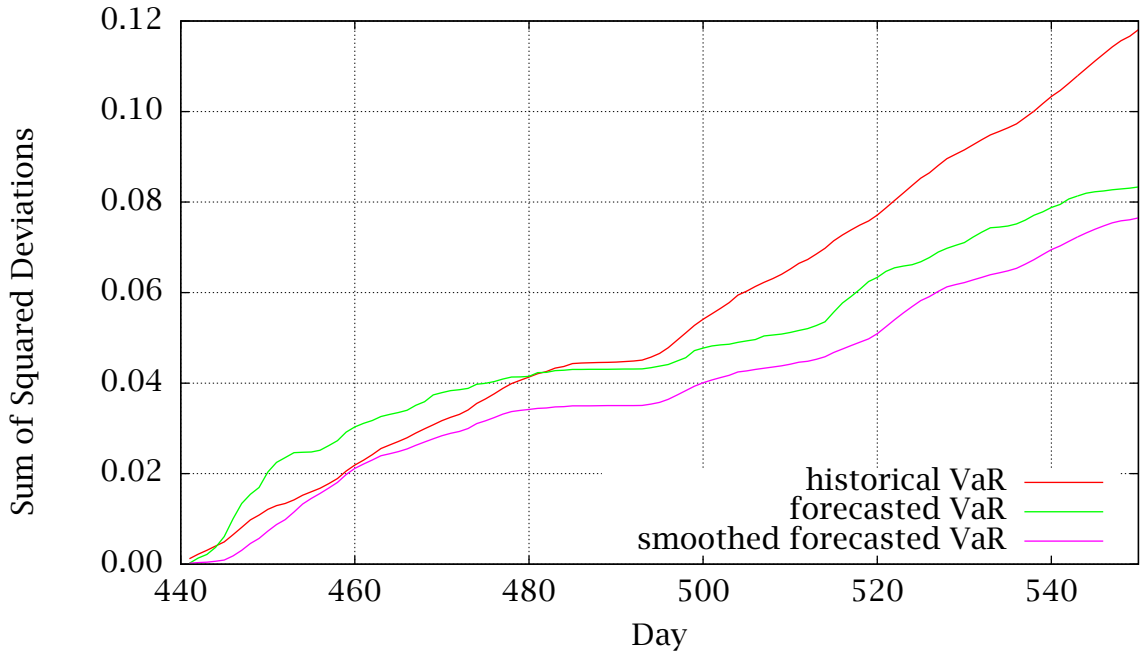


Figure 4.18: Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the GBP|USD exchange rate.

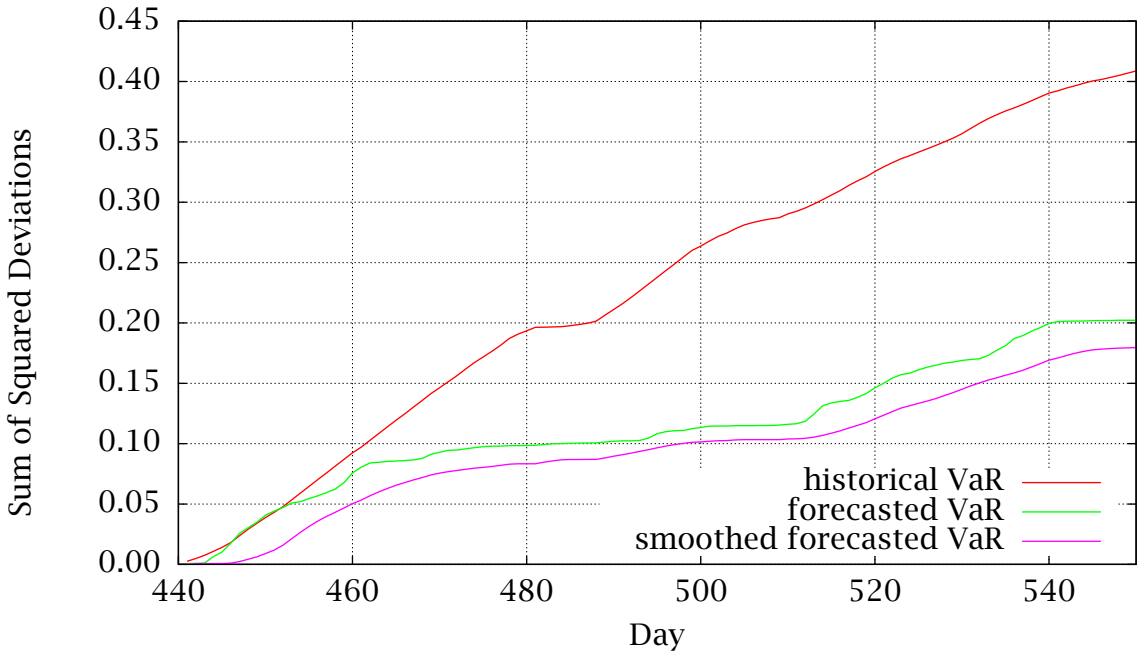


Figure 4.19: Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the Gold Bullion.

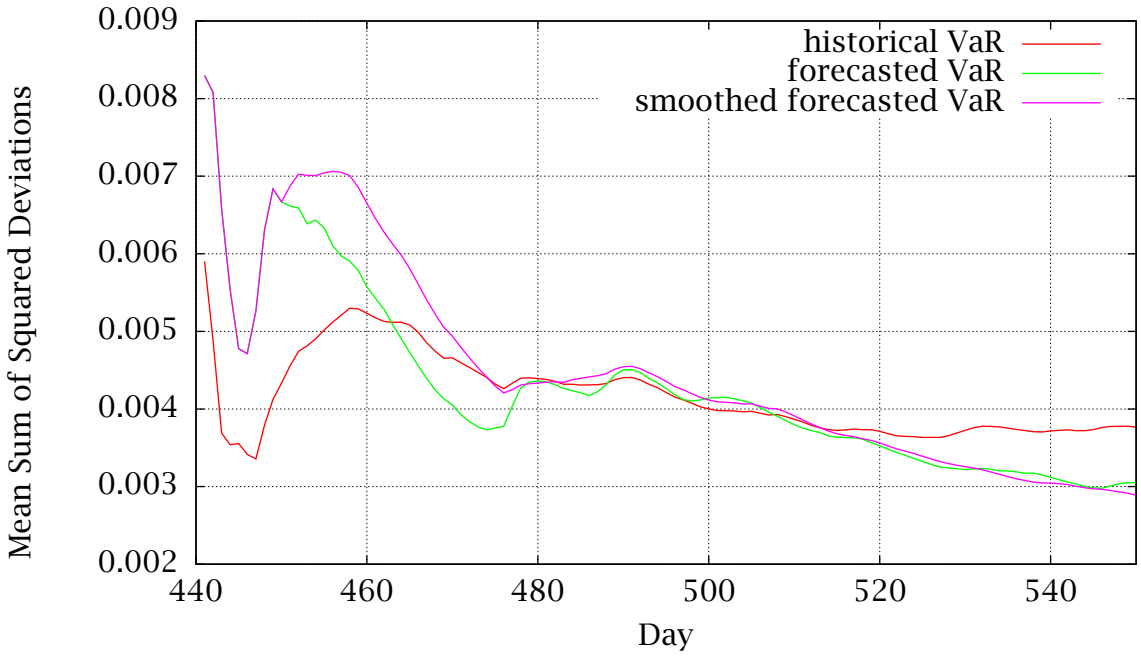


Figure 4.20: Mean Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the FTSE 100.

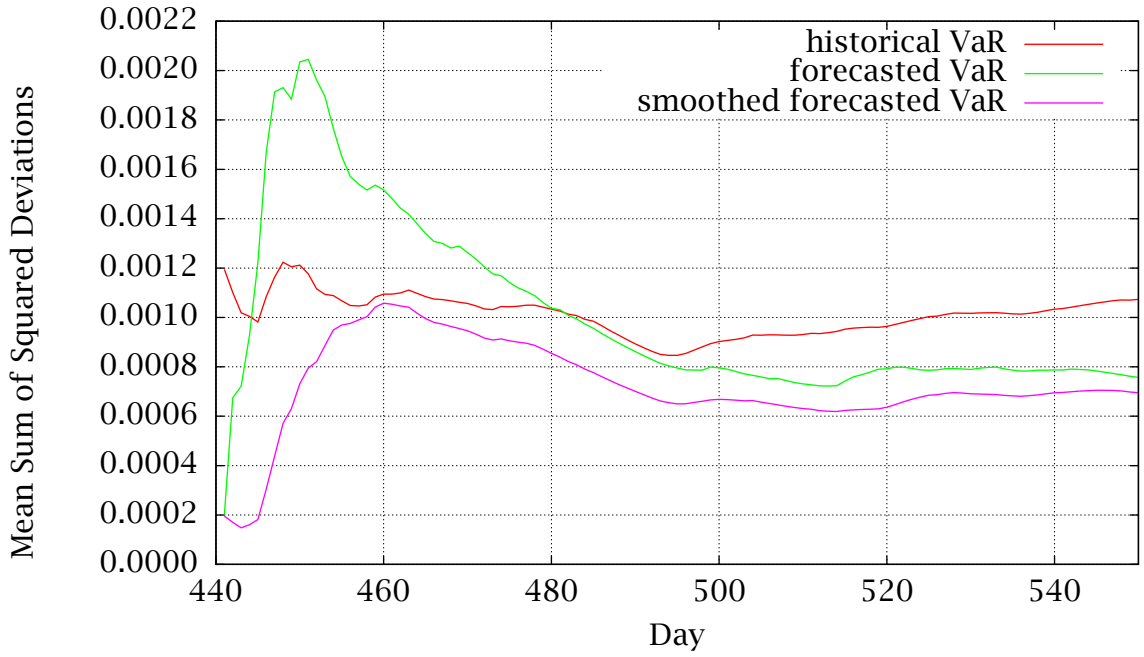


Figure 4.21: Mean Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the GBP|USD exchange rate.

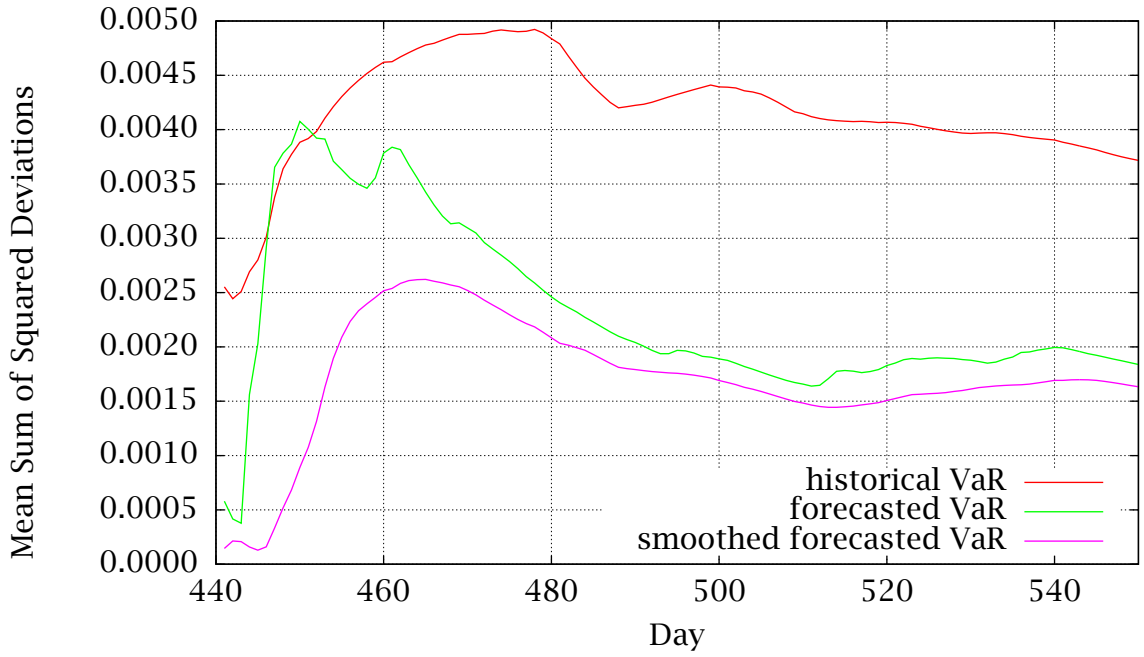


Figure 4.22: Mean Sum of Squared Deviations: Comparison of historical, forecasted and smoothed forecasted VaR for the Gold Bullion.

Mnemonic	time	SSD			mean SSD		
		hist	fc	smooth	hist	fc	smooth
FTSE100	110 days	0.414	0.336	0.318	0.00376	0.00305	0.00289
	8 years	17.546	9.053	8.083	0.00817	0.00421	0.00376
DAXINDX	110 days	0.594	0.550	0.473	0.00540	0.00500	0.00430
	8 years	30.215	13.089	11.733	0.01407	0.00609	0.00546
FRCAC40	110 days	0.389	0.420	0.352	0.00354	0.00382	0.00320
	8 years	23.680	10.487	9.030	0.01102	0.00488	0.00420
FTSEMIB	110 days	0.446	0.463	0.398	0.00405	0.00421	0.00361
	8 years	26.245	12.943	11.586	0.01222	0.00603	0.00539
DJES50I	110 days	0.362	0.104	0.081	0.00329	0.00095	0.00073
	8 years	24.213	3.784	3.084	0.01127	0.00176	0.00144
SPCOMP	110 days	0.645	0.544	0.520	0.00586	0.00495	0.00473
	8 years	14.941	12.158	10.988	0.00696	0.00566	0.00512
NASA100	110 days	3.210	2.241	2.217	0.02919	0.02038	0.02015
	8 years	25.722	75.732	70.908	0.01198	0.03526	0.03301
JAPDOWA	110 days	0.674	0.064	0.084	0.00613	0.00058	0.00076
	8 years	21.521	4.299	3.619	0.01002	0.00200	0.00168
KORCOMP	110 days	2.018	1.744	1.664	0.01835	0.01586	0.01513
	8 years	29.087	61.269	56.884	0.01354	0.02852	0.02648
USDOLLR	110 days	0.118	0.083	0.077	0.00107	0.00076	0.00070
	8 years	2.871	2.251	2.046	0.00134	0.00105	0.00095
SWISFUS	110 days	0.131	0.032	0.027	0.00119	0.00029	0.00024
	8 years	2.192	1.026	0.825	0.00102	0.00048	0.00038
USEURSP	110 days	0.176	0.174	0.156	0.00160	0.00158	0.00142
	8 years	2.723	5.642	5.163	0.00127	0.00263	0.00240
JAPAYEUSD	110 days	0.118	0.120	0.102	0.00108	0.00109	0.00093
	8 years	3.184	4.694	4.281	0.00148	0.00219	0.00199
GOLDBLN	110 days	0.409	0.202	0.180	0.00372	0.00184	0.00163
	8 years	10.690	3.927	3.321	0.00498	0.00183	0.00155
OILBREN	110 days	3.776	2.700	2.359	0.03433	0.02455	0.02145
	8 years	55.391	67.252	62.115	0.02579	0.03131	0.02892
NYFECRB	110 days	0.160	0.131	0.124	0.00145	0.00119	0.00112
	8 years	6.570	3.226	2.947	0.00306	0.00150	0.00137
BALTICF	110 days	0.439	0.170	0.159	0.00399	0.00155	0.00145
	8 years	79.787	9.257	8.629	0.03715	0.00431	0.00402

Table 4.11: VaR benchmark measures. Note that the smoothed forecast is always better than the historical simulation on 110 days. Forecasts that are worse than historical simulation are highlighted in red.
Legend: hist = historical, fc = forecasted, smooth = smoothed

4.7 Purchasing and Transaction Decision Support

In this example one will look at a typical situation of a corporate treasurer: on an ongoing basis some kind of transaction has to be conducted. This may be a regular monthly investment in equities for a pension plan or a fixed income placement. It might be a regular foreign exchange transaction to pay monthly costs in another currency. Or it could be the monthly supply of some commodity, like, e.g., fuel or metal.

Common to all these cases is that the treasurer has to choose an appropriate time for the transaction. This is, of course, the day on which the price is the most favorable. One wants to buy equities, foreign currencies and commodities at the *lowest* price within some time frame. And one also wants to place the money at the *highest* available long term rate. Here, a multi step forecast proves useful, because it gives us an idea of probable price movements.

In the following the author looks at a 20-day ahead multi step forecast for different assets. The benchmark to evaluate market timing is the *realized potential*, RP. RP is a number between 0 and 1, where 0 indicates that the transaction takes place at the *worst* possible moment. $RP = 1$ indicates a perfect fit, i.e., one gets the *best* possible price. To simplify further analysis the author will now assume that one is on the buy side and best is equal to lowest. Then one define the 20-day ahead realized potential at time t of a transaction as

$$RP_t(20) := 1 - \frac{p_t^{\text{realized}} - p_t^{\min}(20)}{p_t^{\max}(20) - p_t^{\min}(20)}. \tag{4.3}$$

$p_t^{\max}(20)$ and $p_t^{\min}(20)$ represent the maximum and minimum prices in the 20-day ahead window starting at time t. p_t^{realized} is the price realized when following the forecast at time t. Please note, that p_t^{realized} is *not* the forecasted price but the actual price at which it would have been possible to trade. Indeed, there is no obligation for the price to follow the forecast.

The author compares RP from the 20-day ahead forecast with RP resulting from buying on a fixed day in the month, i.e., buying always on the 1st, or on the 15th, and so on. For the analysis the author allows for a 440 day learning window, containing training and validation data.

One then uses the resulting neural network ensemble to make a 20-day ahead forecast and evaluate RP. Then one updates the network with new data from the

day and move one time step forward, forecast, evaluate RP and so on. This is a forecast with a 20-day rolling window.

To illustrate how the forecast works the author will have a look at two typical examples. As the dataset comprises 25 assets with 2609 days this makes an astonishing number of $25 \cdot 2609 = 65225$ possible forecasts. The author chooses the two examples, because they are graphically especially appealing. First, have a look at figure 4.23 on page 171 which shows a forecast for the Baltic Exchange Dry Index. The forecast starts at day 441, which is the first day on which the expert topology of neural networks has not been trained. Then, for the next 20 days until day 461, one gets level forecasts.

These level forecasts have been computed by a back transformation of the return forecasts. All forecasts are rebased at 1 to facilitate comparisons between different forecasts. The first observation for the BDI is that the networks follow the general tendency of the realized values. I.e., they first indicate prices to go down, then stay flat for several days and finally go up again slightly.

This has been forecasted *without* ever «seeing» the values between $t = 442$ and $t = 461$. If one wanted to buy dry bulk shipping capacity within the next 20 days: which would be the best day? Clearly, it is $t = 448$ where the BDI hits its low at 1423 points. The neural networks suggests a buy at $t = 456$ or 1428 points. Let the author emphasize important points:

- The neural networks avoid the high values at the beginning of the period under consideration.
- The suggested buy point is not very time sensitive: it is a good timing whether one buys one day earlier or later.
- The neural networks avoid the price rise at the end of the investigated period. For market timing this is very important.

Using concrete values for the BDI one has a high, $p_{441}^{\max}(20) = 1505$ points, right at the beginning of the period. The low occurs at $t = 448$ with $p_{441}^{\min}(20) = 1423$ points. One buys at $p_{441}^{\text{realized}} = 1428$ points. The realized potential for the BDI using equation 4.3 is therefore

$$RP_{441}(20) := 1 - \frac{p_{441}^{\text{realized}} - p_{441}^{\min}(20)}{p_{441}^{\max}(20) - p_{441}^{\min}(20)} = 1 - \frac{1428 - 1423}{1505 - 1423} = 0.94.$$

This means that one achieved 94 percent of the best possible price.

Figure 4.24 on page 172 shows the typical case of a forecast. One starts at $t = 451$ and forecast the price of the Gold Bullion for the next 20 days, i.e., until $t = 471$. In this case the networks suggest a buy at $t = 456$ which is actually not the absolute low but still close to it. One sees what happens, when the networks overshoot *at the beginning* of the forecast, here in the direction of low values: as the forecasts are based on *returns* rather than *levels*, small errors at the beginning tend to skew subsequent values.

This is a phenomenon only observeable in true multi step forecasts. In the present case the networks tend to exaggerate the low and only get the trend right again in the following. And this is substantial: a corporate treasurer basing his decision on the expert topology would still get a feel for where the gold price is headed, even without having an *exact* level forecast.

The treasurer would see that it makes sense to buy in 3 to 6 days time, because the gold price is expected to decrease. She would see that the exact timing decision is not very time sensitive, because the gold price should stay flat for some time — and it does. But she would also note, that the gold price will rise at the end of the period — and this is indeed what happens. One gets RP as

$$RP_{451}(20) := 1 - \frac{p_{451}^{\text{realized}} - p_{451}^{\min}(20)}{p_{451}^{\max}(20) - p_{451}^{\min}(20)} = 1 - \frac{258.30 - 257.05}{264.85 - 257.05} = 0.84.$$

One realizes 84 percent of the high-low span over 20 days.

The exemplary above results look promising. Two questions remain however which the author wants to address in the following:

- Do the forecasts outperform the benchmark? I.e., do the forecasts beat the typical strategy of a corporate treasurer whose primary goal is *not* to predict the financial markets. Or would a simple strategy of buying always at the same day in a four week cycle perform better? The latter strategy is currently often implemented.

- Are the results consistent over time and over all assets? Or does the model «age» and looses forecasting power? Do certain assets or certain type of asset classes perform better?

To answer the first question one looks at a near term forecast over the next 110 days and calculate the excess realized potential. I.e., one compares the realized potential of the neural networks with the strategy of buying always at some fixed

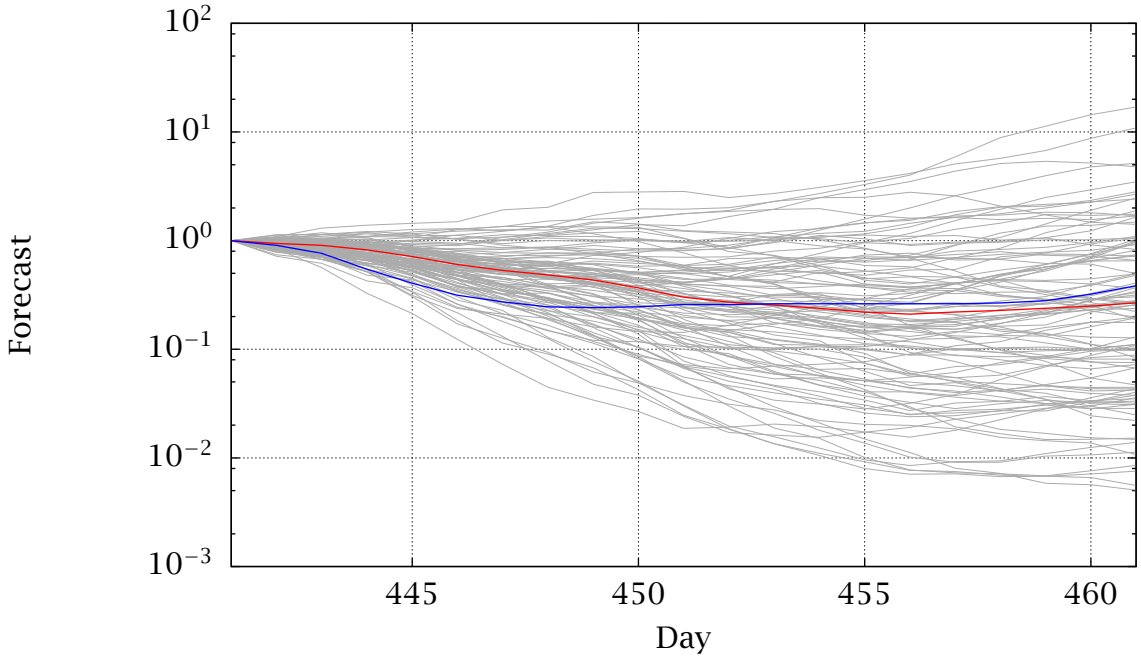


Figure 4.23: An almost ideal example of a 20-day forecast for the Baltic Dry Index. Realized values in blue, forecast in red. Additionally the gray lines show forecasts of all different networks. Note, how the networks gets the general tendency right: first down, then flat, then rising again a little bit. It also suggests a «buy» at a very sensible low point although the actual value of the low is not quite hit. Keep in mind, that this is a genuine 20-day ahead forecast: the network runs freely for 20 time steps *without* input of realized data. To allow comparisons with other forecasts the values are rebased at 1.

day in a four weeks — or 20 days — cycle. Formally one define the excess realized potential as

$$ERP = RP_{\text{neural}} - RP_{\text{fixed day}} \tag{4.4}$$

with RP from the two respective strategies. ERP is a value in the range $[-1 \ldots 1]$. -1 signals that the simple strategy was perfect but the neural network suggested to buy at the high — the worst possible case. 1 signals the inverse: the simple strategy performed worst and the neural network hit the low exactly. Clearly, one expects $ERP > 0$ for the forecasts to offer any added value. If one consistently had $ERP < 0$ one would be better off not using the forecasts. To get a performance measure for

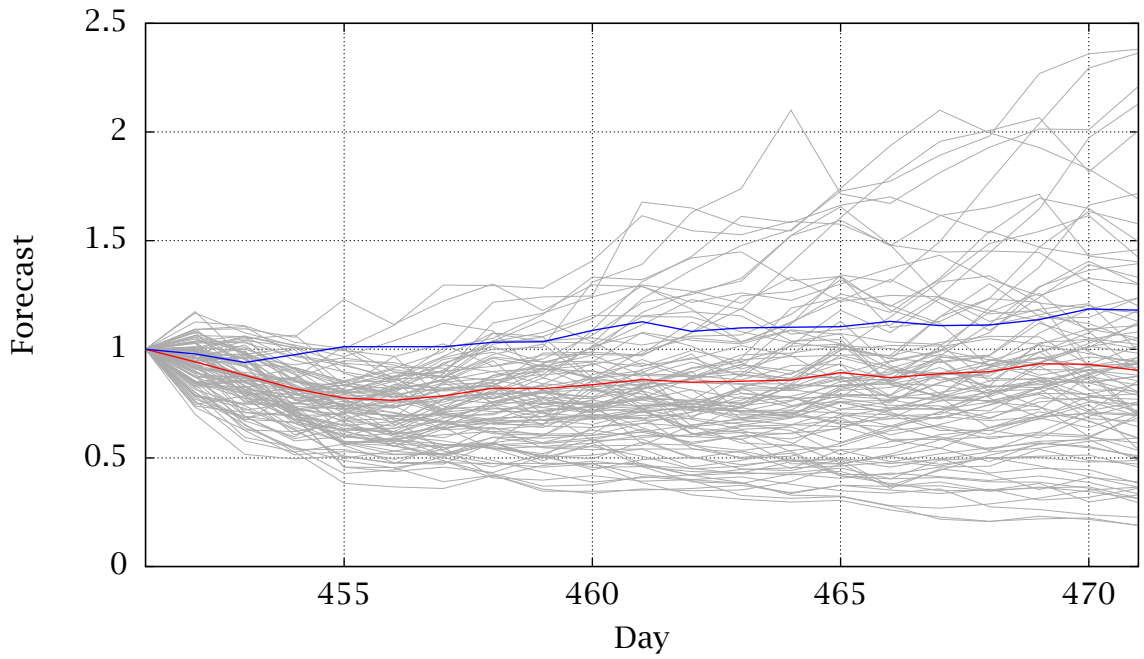


Figure 4.24: Forecast for the Gold Bullion showing a typical case. The network misses out on the absolute minimum but the suggested buying point on day 456 is not bad at all, considering that the Gold price continues to *rise* afterwards. Again, one notes that the network appropriately gets the general tendency right, but an overshooting in the first four days causes the values to be skewed. Realized values in blue, forecast in red. The gray lines show forecasts of all different networks.

different time spans one calculate the cumulated excess realized potential as

$$cERP = \sum_{t=t_{\min}}^{t_{\max}} ERP_t \tag{4.5}$$

where $t_{\min}$, $t_{\max}$ represent the time span of interest.

The results are shown in figure 4.25 on the facing page. To interpret this figure one should first consider that in every case *cERP* is positive. I.e., for near term forecasts the model *always* performs better than any fixed day strategy. This is true for every asset.

One also note that there are fixed days, mostly in the range 12–18, which show only relatively low *cERP*. A detailed analysis of this effect is beyond the scope of the present work. Suffice it to say that an *optimized* fixed day strategy can be a hard

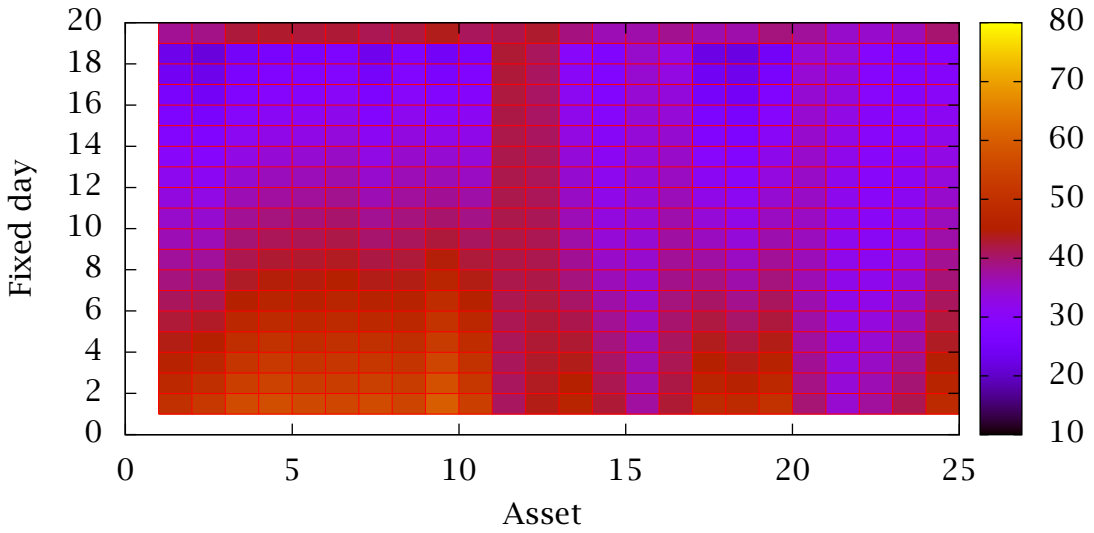


Figure 4.25: Cumulated yearly excess realized potential for a typical time span of 110 days and different fixed day strategies. This map is intended as overview, only. Please find the exact numbers in table 4.12 on the following page. Note, that for *every* asset and *every* day of the month the excess realized potential is positive. I.e., the forecasts add value consistently. See also figure 4.30 on page 184 for a comparison to an 8 year forecast.

benchmark to beat. This is already recognized by [153]. Because of, e.g., futures expiry certain days of a week and of a month will consistently exhibit certain return patterns. Conveniently optimized a strategy might simply exploit this although one can doubt if the pattern will *really* be persistent. Obvious patterns are generally exploited rapidly by arbitrageurs.

Table 4.12 on the next page allows a detailed analysis of *cERP* for the 110 day time span. One especially notes that even assets which are linearly relatively uncorrelated to the others feature significant *cERP*: Nikkei, Kospi, the Japanese yield curve, and the USD|JPY exchange rate are all not worse than other assets. One notes a general tendency for low and high *cERP* to cluster. I.e., one does not have high *cERP* followed by low *cERP* and so on but mostly similar *cERP* next to each other. The author will look at this in more detail below.

Table 4.12: Realized excess potential compared to different fixed day strategies for a typical model validity time stretch, here one quarter of training and validation data, i.e., 110 days. Note, that the realized excess potential is *always* positive. This means that for *every* asset and every possible day of the month strategy the neural network adds economic value. One is better off using the forecast.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									FTSE100
52.30	50.11	47.99	46.06	44.43	42.75	41.00	39.31	37.84	36.35
34.99	33.37	31.82	30.21	28.73	27.24	25.88	24.47	23.05	54.77
									DAXINDX
50.47	48.02	45.88	43.95	42.30	40.70	39.13	37.54	36.07	34.54
32.98	31.27	29.72	28.30	26.83	25.42	24.18	22.88	21.50	53.33
									FRCAC40
56.91	53.92	51.04	48.61	46.22	43.96	41.73	39.71	37.82	35.91
34.09	32.05	30.09	28.23	26.47	24.84	23.32	21.69	20.06	60.34
									FTSEMIB
58.57	56.28	54.29	52.42	50.72	48.90	46.97	45.09	43.54	41.89
40.27	38.55	36.95	35.49	34.10	32.68	31.45	30.01	28.42	60.82
									DJES50I
58.29	55.43	52.75	50.35	48.23	46.21	44.19	42.21	40.34	38.50
36.68	34.57	32.63	30.74	28.91	27.09	25.26	23.34	21.35	61.46
									SPCOMP
56.17	54.05	52.29	50.77	49.39	47.94	46.51	45.07	43.67	42.11
40.57	39.00	37.55	35.99	34.44	32.93	31.48	29.98	28.44	58.42
									NASA100
57.09	54.81	52.77	50.89	49.25	47.50	45.59	43.68	41.72	39.83
38.02	36.10	34.32	32.58	30.86	29.13	27.36	25.73	24.01	59.58
									JAPDOWA
57.96	55.48	53.25	51.07	48.77	46.54	44.37	42.22	40.30	38.51
36.57	34.63	32.75	30.87	28.90	26.88	24.92	22.97	21.06	60.74

Continued on next page

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									KORCOMP
54.10	52.64	51.17	49.56	48.33	47.01	45.73	44.43	43.09	41.58
40.25	39.03	37.92	36.78	35.59	34.42	33.30	32.14	31.18	55.76
									BBGBP12
68.45	64.91	61.57	58.58	55.71	52.81	49.89	46.96	44.04	40.96
37.93	35.12	32.48	29.78	27.10	24.47	21.75	18.94	16.14	72.13
									ECEUR3M
41.22	41.09	40.85	40.72	40.55	40.33	40.08	39.74	39.22	38.59
37.69	37.10	36.50	36.10	35.84	35.58	35.19	34.69	34.29	41.06
									UKyc
40.05	40.49	40.84	41.45	41.84	42.52	43.21	43.73	44.21	44.86
45.54	46.25	46.83	47.24	47.79	48.53	49.51	50.46	51.33	39.60
									GERyc
48.58	47.01	45.67	44.52	43.24	42.09	40.80	39.59	38.50	37.67
36.68	35.87	35.09	34.37	33.56	32.54	31.80	31.19	30.78	50.29
									FRyc
45.55	44.36	43.20	42.26	41.12	40.02	38.82	37.72	36.54	35.51
34.51	33.70	32.97	32.14	31.35	30.47	29.82	29.14	28.79	46.91
									ITyc
40.97	39.60	38.47	37.39	36.22	35.11	34.16	33.03	32.23	31.45
30.65	29.98	29.29	28.74	28.17	27.69	27.30	26.98	26.61	42.61
									USyc
34.62	34.67	34.78	34.98	35.27	35.35	35.56	35.92	36.57	37.03
37.56	38.10	38.56	39.06	39.61	40.35	41.25	42.21	43.17	34.88
									JAPyc
51.78	49.85	48.58	47.13	45.60	43.96	42.27	40.64	38.86	37.02
35.12	33.32	31.83	30.10	28.62	26.83	25.06	23.40	21.70	53.58
									USDOLLR
47.29	45.31	43.48	41.86	40.28	38.62	37.07	35.50	33.98	32.42
31.07	29.83	28.72	27.64	26.45	25.41	24.33	23.24	21.82	49.52

Continued on next page

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									SWISFUS
51.61	48.95	46.46	44.09	41.98	40.06	38.25	36.58	34.95	33.51
32.25	31.12	29.77	28.21	26.80	25.27	23.77	22.35	21.04	54.91
									USEURSP
51.51	49.48	47.55	45.80	44.39	43.06	41.77	40.68	39.60	38.50
37.53	36.47	35.36	34.06	32.82	31.40	29.93	28.51	27.31	53.92
									JAPAYEUSD
28.93	29.03	29.37	29.73	30.13	30.58	30.87	31.49	32.02	32.38
32.89	33.55	34.32	35.30	36.19	37.28	38.42	39.59	40.84	28.93
									GOLDBLN
40.62	39.08	37.91	36.66	35.56	34.50	33.61	32.61	31.74	30.89
30.41	29.75	29.38	29.07	28.75	28.44	27.94	27.26	26.64	42.53
									OILBREN
36.18	34.69	33.46	32.48	31.78	31.08	30.54	30.17	30.14	30.06
30.25	30.48	30.58	30.64	30.89	30.99	31.20	31.21	31.26	38.03
									NYFECRB
48.32	46.30	44.66	43.03	41.51	40.09	38.58	37.00	35.40	34.00
32.65	31.49	30.50	29.53	28.59	27.75	26.68	25.54	24.39	50.74
									BALTICF
50.64	48.30	46.58	45.19	44.00	42.88	41.80	40.73	39.67	38.64
37.66	36.70	35.81	34.98	34.21	33.45	32.63	31.74	30.90	53.87

The author now addresses the second question of robustness and model consistency over time. For this the author first has a detailed look at four selected assets from every category:

FTSE 100 Index Figure 4.26 on page 180 shows the cumulative excess realized potential for the index. The figure covers a time span of more than eight years into the future during which the model is not retrained. One notes that at the beginning of the forecast, starting with day 441, the *cERP* monotonically increases for *every* fixed day strategy. This is, of course, consistent with the previous analysis which showed very good results for a 110 day time span. The more one advances into the future, however, the more mixed the results get: for fixed day strategies on days 1–10 the results stay positive. On other

fixed days one periodically notes negative *cERP*. This indicates that up to this point in time, e.g., the valley around day 1000, a fixed day strategy would have performed better. Periods where the *cERP* rises again follow. Interestingly one sees that towards the end of the period under investigation there is a general tendency for *cERP* to rise again. To understand this phenomenon one looks back at the FTSE 100 level series chart, figure 4.2 on page 135. one notes that the training and validation period comprises the end of the bull market and beginning bear market in 1999–2001. Viewed altogether one would qualify the training and validation period as *sideways* market where there is no clear tendency in the returns: positive and negative returns alternate.

This is comparable to the situation at the end of the out-of-sample period 2008–2009. One notices a bottom forming in the prolonged bear period at the beginning of 2009 and then a moderate rise again. But the prices seem to move in the range between 3500 and 4500 points. This is about the same range as the prices at the beginning of the training, between 6000 and 7000 points. It is comforting to see that the networks manage to *remember* these situations. However, the networks seem to have problems with trending markets. This is especially noticeable when one considers the bear period of 2001–2003. During that period, a consistent downtrend, it would almost always have been better to buy as late as possible, because prices were falling. One clearly sees the downward bump that ensues for *cERP*.

Lastly, one notes an overall good *cERP* for fixed day strategies at day 20. It would be very interesting to analyze this effect in more detail. As a preliminary hypothesis one can attribute this to the fact that day 20 corresponds to the end of the month. Generally, at the end, of the month several transactions occur, e.g., from institutional or government investors which are not necessarily related to some *economic* reason. In the case of institutionals there is often some kind of window-dressing going on. In the case of governments, transactions are sometimes deliberately announced to stabilize markets. If any, one may draw the conclusion, that a fixed day strategy at the end of the month is unattractive compared to the neural network. [153] offers more insights related to fixed day effects.

Detailed *cERP* numbers for the entire forecasting period are shown in table 4.13 on page 185. One sees in the table that the worst *cERP* is at −26.02 whereas the best is realized at 134.39. This result is very satisfying, as one

would have incurred a small performance penalty in only six cases. But in the other fourteen cases one observes significant gains.

UK yield curve *cERP* values for the yield curve are on figure 4.27 on page 181. One see that at any given day the *cERP* stays positive. One still sees the effect of a strong increase at the beginning of the forecasting period, indicating a well adapted model. Overall the *cERP* increases almost monotonically, with the same tendency of lower *cERP* towards the end of the month. An explanation for the very good performance of the neural network is, that the UK gilt market is not perceived as being very efficient. This holds true for most government bond markets, see [333]. As already mentioned, government entities will not have trading revenues as their primary objective when interacting with their bond market. This gives quantitative models a certain edge which the neural network is capable to capture.

What is the economic value in finding a low point in the yield curve? One recalls that a low point in the yield curve signifies that long term interest rates are relatively low compared to short term interest rates. Be careful: the long term rate does not have to be *lower* than the short term rate. It can be, though. In any case a low point is an attractive time for borrowing long term funds and simultaneously lending short term funds. A corporate treasurer with the duty to ensure long term funding for a company on an ongoing basis should choose a low point in the yield curve when securing long term funds and — at the same time — invest excess money at short term rates. This transaction is *only* sensible when both components are considered: long term borrowing and short term lending. If only one component is needed it is better to develop a model to predict low points in either the long or short term rate series.

The inverse problem, finding *maxima* of the yield curve, is the typical day-to-day challenge for a bank. It is an aspect of one of the core reasons why banks are actually in existence: term transformation. Typically, clients want to borrow money for the long term, i.e., the bank *lends* to them. On the other hand, banks refinance themselves short term — generally by paying interest to other clients who hold money in their account. I.e., the bank *borrows* from them. This operation is low risk and often low margin and that's why banks obviously want to time their borrowing and lending decisions to capture more of the interest rate differential. As an aside: The author has always been

surprised when talking to bankers by the ubiquitous wish to get an accurate view of the yield curve. Most do not want models to forecast something fancy like stocks or currencies — just the yield curve! And that's why the author is especially glad that yield curve timing works well with neural networks.

GBP|USD exchange rate Figure 4.28 on page 181 shows *cERP* for the currency pair. Again, one finds the typical increase at the beginning of the period. However, the day of the week effect is much less pronounced than for the FTSE or the UK yield curve. *cERP* rises and falls mostly depending on the time period not on the fixed day. One notes that towards the end of the period the overall good *cERP* falls dramatically. There is still significant *cERP* realized as can be seen in 4.13 on page 185 but the very good values of the year 2007 are unfortunately lost. Nevertheless one gathers from the table that the worst case is $cERP = -18.47$ whereas the best case is more than five times that, $cERP = 102.93$.

How can one explain the performance loss just towards the end of the out-of-sample period? A look at the GBP|USD level series is quite revealing, see figure 4.4 on page 137. One notes that the networks are trained on a sideways period followed by a mild downtrend. Starting mid 2008 one notices a very abrupt downtrend followed by an equally sharp reversion. Clearly, the neural network has never seen such kind of returns and fails to perform. It manages, however, to catch up slightly in the following uptrend. Looking at the corresponding level figure 4.9 on page 147 one notices another problem. As this is a true out-of-sample test data has been scaled appropriately to the training and validation period. The end of the out-of-sample period shows however more extreme returns. Of course, the scaling includes a safety margin. But nevertheless the inputs are saturated and therefore the network is *in principle* not able to represent these values adequately. This is a risk which one is always confronted with. It can be mitigated by retraining more often and adapting the safety margin to current market conditions. However, the scope of the present work is to analyze model robustness during long time periods.

Gold Bullion One finds *cERP* for the Gold Bullion on figure 4.29 on page 182. One notes a very good *cERP* performance always positive at every point. Interestingly the fixed day of the month effect is reversed compared to the FTSE 100 performance: better *cERP* is realized towards the end of the month. One

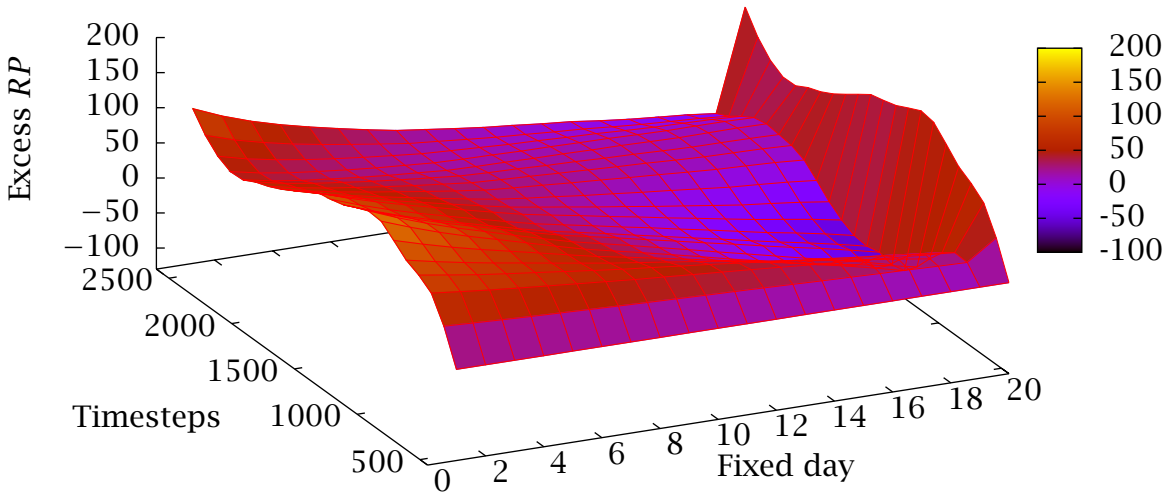


Figure 4.26: Excess realized potential for the FTSE 100 index.

may attribute this to the fact that gold is not only used for speculative purposes but also as reserve currency and as real commodity. There are some inefficiencies resulting from market participants who are obliged — or feel obliged — to buy or sell. Inefficiencies in the precious metals markets are also acknowledged in [116, 242]. Interestingly these inefficiencies are quite persistent as [116] is a recent study from 2007 and [242] was already published in 1994. The present work extends the analysis up to mid 2009 and confirms the findings for gold.

An application for an ongoing gold market timing model is its use in the industry. As gold is an excellent electric conductor and does not oxidate in air or water it is used for electrical connections and contacts where costs of failure are high. This includes, e.g., air and space crafts.

Figure 4.30 on page 184 presents cumulative realized excess potential for all assets and all fixed day strategies, the detailed corresponding values are shown in table 4.13 on page 185. Generally one notes from the figure that often inferior performance occurs clustered at the end of the month.

This includes the FTSE 100, Dow Jones Euro Stoxx, Nikkei, German, French, Italian, United States and Japanese yield curve, GBP|USD, USD|SFR, USD|JPY. A few as-

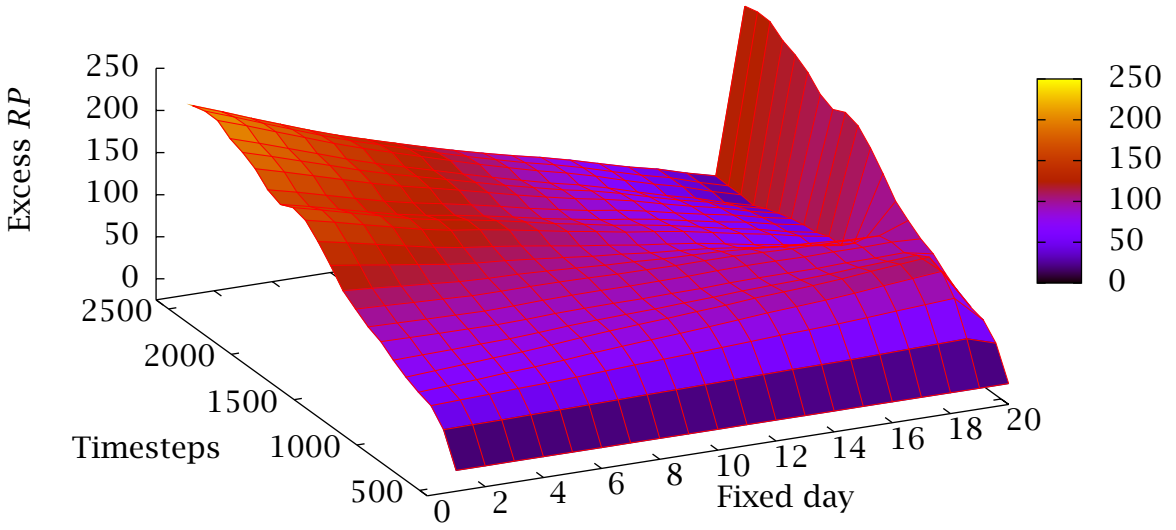


Figure 4.27: Excess realized potential for the UK yield curve.

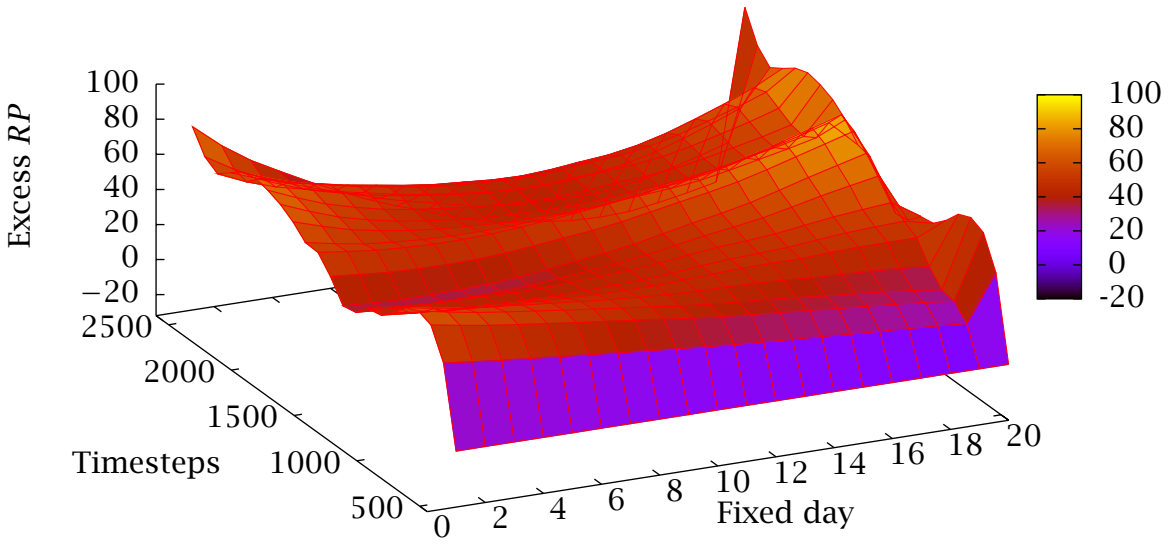


Figure 4.28: Excess realized potential for the GBP|USD exchange rate.

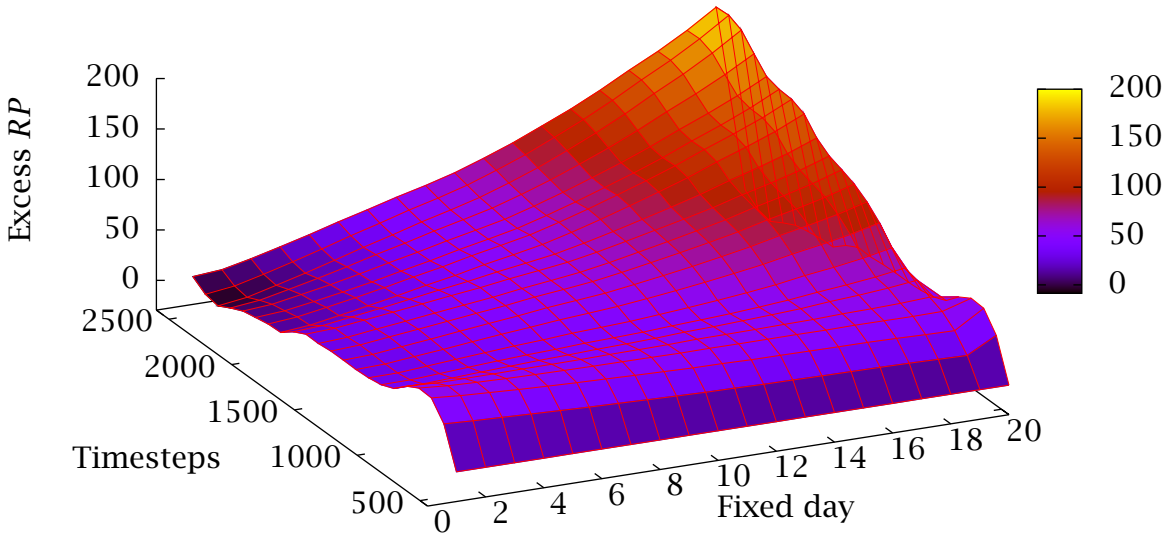


Figure 4.29: Excess realized potential for the Gold Bullion.

sets show inferior performance at the beginning of the month: Kospi, EUR|USD and — only to some extent — Brent oil. Several assets are very good performers without any negative values: DAX 30, CAC 40, FTSE MIB, S&P500, NASDAQ 100, 12 months LIBOR and 3 months EURIBOR, UK yield curve, Gold Bullion, CRB Index and Baltic Exchange Dry Index.

One notes that from the Western equity indices only FTSE 100 performs inferiorly. On day 19 one also has a small negative value for the Dow Jones Euro Stoxx. The other well known indices all show overwhelmingly good performance. The two indices from Asia Pacific, however, are not so convincing. Still, the best performing days are at least twice as good as the wort performing, and the underperforming days are a minority.

Nevertheless, if one wanted to improve performance on Asia Pacific one should include more relevant time series. Asia Pacific performance is also hurt by time zone aspects: the data from Asia Pacific is first to arrive, around 11.00 GMT. It can influence the ongoing trading session in Europe and the opening of the US session at 15.00 GMT. However, when US trading closes around 22.00 GMT there is a *gap* of 13 hours until the next Asian close. One can attenuate this problem by including additional data.

Considering interest rates one notes that interest rate differences are easier to forecast than yield curve shifts. Still, in every case the good forecasts outperform the bad. Further analysis if market interaction of other governments is different compared to the United Kingdom could lead to interesting results. For now, its very satisfying to see such good performance for short term interest rates: 3 and 12 months.

One notes that currencies are difficult to predict in the long term. Foreign exchange is notorious for having very little exploitable inefficiencies, see [333]. This work confirms the findings. Especially EUR|USD underperforms in a majority of cases. Very interestingly, USD|JPY performs surprisingly well. One only has four negative days and the worst performance of -20.21 is dwarfed by the best performance of 228.87 *cERP*. According to [333], USD|JPY is the second most actively traded currency pair.

Commodities perform very satisfactorily. One only has a small underperformance for oil on five days. All other values are positive. One especially notes consistently high values for the CRB index and the Baltic Exchange Dry Index. As commodities *always* fulfill some real economic purpose inefficiencies are not uncommon. Especially the less traded commodities are prone to these, see, e.g., [105, 226]. This is even more true for the BDI: prices are always *real* in the sense that they are always tied to existing or soon to exist capacity. The Baltic Exchange is a market where real capacities are traded by those who need them and those who have them. As demand and supply is inelastic and likely to stay so one is not surprised to see that the network can exploit the inefficiencies of this market. Additionally, an electronically tradeable version of the BDI future is only available since June 2008 through Imarex. It remains to be seen if this market attracts enough quantitative strategies to void returns.

To wrap-up this section on purchasing decision support the author emphasizes that results are very satisfying. For the short term forecast of 110 days the neural network model *always* beats the benchmark. It shows very consistent performance as illustrated by figure 4.25 on page 173. What is even more satisfying is that a *single* expert topology of neural networks shows robust performance on a timespan of more than eight years. Training and validation data only spans two years. Although performance is not always positive for all assets there are still assets, especially commodities and Western equity indices, where performance is much better than the benchmark. One may ask: when performance is negative why shouldn't

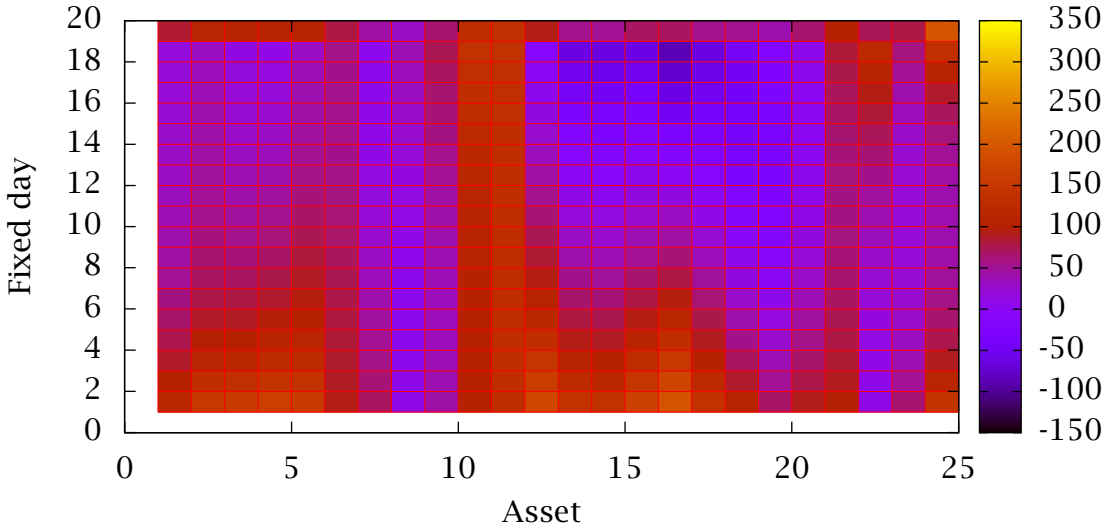


Figure 4.30: Cumulated excess realized potential for the entire time span of 8 years and different fixed day strategies. This map is intended to give the reader an overview, only. The exact numbers can be found in table 4.13 on the next page. What the reader should take from this figure is that the model is remarkably robust. Several assets still show positive excess realized potential, although the model has only been trained and validated on the first 2 years. Note that, although for some assets and for some days negative excess potential is shown, this is only with the benefit of *hindsight*. One couldn't have possibly known *before*, which day of the month would lead to the best results.

a corporate treasurer always take the best performing fixed day strategy? It is tentative to do that. But remember that the comparison is biased towards the fixed day strategy. One chooses — or more appropriately: trains — neural networks on a timespan of almost two years and select them *a priori*. This means one lets them run untouched for eight years. Selecting «the best» fixed day strategy is however an *ex post* decision. Only after eight years can one tell which strategy has performed best. It is only with the *benefit of hindsight* that one could choose an appropriate fixed day strategy. one concludes that multi step neural network forecasts do indeed add economic value to purchasing decisions.

Table 4.13: Realized excess potential compared to different fixed day strategies for the *entire* dataset, i.e., 8 years. Negative excess potential is highlighted in red. With the benefit of hindsight one can tell for which fixed day strategies the neural network would actually have performed worse in a time span of 8 years. But note how remarkably robust the neural network nevertheless is. E.g., several equity indices and the Gold Bullion still produce positive excess realized potential, although the neural network has been trained and validated only with the first 2 years of data and runs freely for the next 8 years.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									FTSE100
111.00	92.27	76.87	64.14	53.17	43.61	34.61	26.34	20.66	14.91
9.58	5.57	0.96	-3.15	-8.71	-13.69	-17.42	-21.10	-26.02	134.39
									DAXINDX
144.60	125.03	110.31	97.48	87.58	79.34	72.03	67.44	65.08	62.33
60.00	59.76	59.28	59.23	58.89	58.76	58.19	58.75	58.87	171.78
									FRCAC40
183.26	154.38	131.24	112.37	96.96	84.60	73.13	63.90	56.66	49.39
43.14	38.81	33.63	28.78	23.78	18.12	13.41	9.27	4.03	217.46
									FTSEMIB
158.28	134.49	116.87	103.49	91.94	82.18	72.95	64.37	58.57	51.58
45.82	41.65	37.28	33.22	29.67	25.71	23.61	21.69	17.69	186.44
									DJES50I
195.22	168.40	147.02	128.31	113.41	100.64	88.71	78.41	69.79	60.48
51.56	44.59	37.90	31.74	25.27	18.46	11.91	5.79	-2.51	227.96
									SPCOMP
133.05	123.21	115.34	109.23	104.27	99.56	95.50	91.26	86.28	80.97
76.02	72.26	69.49	67.18	66.07	65.19	64.04	63.32	60.52	143.11
									NASA100
60.26	58.83	58.21	57.75	58.38	57.53	54.72	53.11	49.89	47.54
44.63	42.18	40.45	40.47	41.38	41.52	43.56	47.27	49.24	63.02

Continued on next page

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									JAPDOWA
85.56	71.52	59.34	49.55	39.22	29.96	20.96	12.69	6.13	0.03
-6.31	-12.00	-17.13	-21.15	-24.61	-28.60	-32.72	-36.43	-40.31	104.02
									KORCOMP
-59.98	-54.25	-47.47	-40.67	-32.50	-21.67	-11.55	-0.87	9.95	19.91
30.61	42.21	54.52	66.38	78.00	89.68	99.58	109.62	120.51	-64.68
									BBGBP12
151.92	135.78	122.79	111.81	102.43	93.32	84.01	75.67	68.51	62.15
56.83	52.41	48.80	44.70	41.13	37.18	33.52	31.62	31.42	169.54
									ECEUR3M
60.75	71.17	81.35	92.15	102.24	113.21	124.98	135.24	145.55	154.84
164.59	174.58	185.00	195.68	206.83	218.40	229.35	242.22	255.06	48.72
									UKyc
204.84	192.26	179.03	166.34	153.72	142.10	131.70	121.64	111.72	102.42
93.87	85.59	78.10	69.55	60.07	50.82	42.97	33.08	24.70	220.06
									GERyc
165.30	145.30	127.25	112.21	96.84	82.73	68.75	57.05	44.91	33.25
21.62	10.97	0.70	-9.26	-20.23	-31.61	-42.85	-52.48	-60.54	187.36
									FRyc
137.27	119.42	102.71	88.55	74.49	61.88	49.09	37.95	26.63	15.38
4.48	-5.27	-14.11	-23.07	-33.58	-44.17	-54.75	-64.38	-72.16	157.03
									ITyc
157.75	138.29	120.68	104.89	88.72	74.22	60.17	47.87	35.70	23.23
10.80	0.14	-9.41	-18.63	-29.86	-39.82	-49.43	-57.31	-64.48	180.08
									USyc
197.38	174.25	151.92	132.74	115.57	98.87	82.33	67.03	53.52	41.05
29.69	18.05	6.38	-6.08	-18.42	-31.02	-43.18	-54.56	-64.91	222.15
									JAPyc
213.97	190.41	169.44	149.26	131.01	111.48	92.75	73.52	53.84	34.53
14.83	-4.03	-23.43	-41.79	-58.89	-75.70	-91.53	-107.21	-122.11	241.42

Continued on next page

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
									USDOLLR
82.28	67.36	55.12	45.14	35.47	26.73	18.20	11.00	4.71	-2.14
-7.68	-11.78	-14.63	-17.35	-18.47	-18.34	-16.28	-13.60	-10.32	102.93
									SWISFUS
153.70	125.01	100.23	78.13	59.34	40.96	24.55	7.90	-7.65	-21.21
-32.54	-42.92	-50.70	-58.52	-64.24	-68.25	-73.48	-75.96	-74.94	188.63
									USEURSP
-1.09	-9.07	-14.62	-16.80	-18.26	-19.21	-21.55	-22.02	-22.19	-21.46
-18.13	-14.91	-10.17	-7.20	-1.48	6.36	14.82	24.27	35.04	11.50
									JAPAYEUSD
200.40	175.96	156.69	137.57	119.78	102.89	85.87	71.28	58.28	46.23
35.87	26.13	17.42	9.57	2.01	-5.48	-10.71	-15.23	-20.21	228.87
									GOLDBLN
9.10	10.98	16.95	23.88	31.18	39.09	46.58	54.42	61.41	69.40
79.32	89.95	101.94	114.05	127.79	143.03	157.28	173.48	191.90	9.52
									OILBREN
13.36	6.11	1.04	-1.69	-2.16	-2.60	-3.09	-2.17	0.24	3.22
6.35	10.34	14.93	19.83	26.26	34.58	43.90	53.04	65.12	22.10
									NYFECRB
127.60	104.79	86.11	71.96	63.21	56.53	50.95	45.08	40.95	37.26
33.94	33.12	31.84	33.23	35.72	40.17	46.38	54.27	65.91	159.50
									BALTICF
193.88	148.25	110.71	92.73	73.91	62.89	51.94	45.21	41.33	41.51
43.79	50.54	59.97	73.94	91.66	114.53	143.50	182.59	237.73	322.78

4.8 Investment Decision Support

Literature review already shows that an overwhelming number of published neural network applications in finance deal with investment and trading decision support. The reason for this is that the results from different studies are easily comparable and intuitive to grasp: everybody will agree that a higher annualized return is better provided that incurred risk is adequate. Once one has devised an investment strategy it is straightforward to calculate risk measures like, e.g., sharpe ratio and maximum drawdown, and to compare the results with every other strategy. Although the emphasis of the model is not particularly based on forecasting correct sign changes it is interesting to see, how shared layer perceptrons perform. Following two examples which use all features of the model, i.e., multi step forecasts and distribution, one now analyzes a typical quantitative investment approach.

Here the focus lies on finding an *adequate* investment strategy. This is in its most basic form equivalent to correctly forecasting the sign of tomorrow's rate of return. The author benchmarks the neural network against two other technical strategy, see also [110]. The first is the naive strategy, which simply states

$$r_{t+1} = r_t,$$

i.e., tomorrow's return equals today's. If today's price already reflects all available information then the best bet is, that the return remains unchanged. In an efficient market the naive strategy should be a tough benchmark to beat. This also provides us with an efficiency measure. The excess return, if any, of the neural network over the naive strategy indicates how efficient or rather inefficient markets are.

The second strategy is a typical technical strategy using moving averages. The n-day moving average is

$$MA_n(t) = \frac{1}{n} \sum_{i=t-n+1}^{t} X_i$$

where X_i is the *level* — not the return. One now looks at two moving averages of different periods. The rule is to buy when the shorter moving average moves through the longer one from *below*. One sells when the reverse happens: i.e., the shorter moving average intersects the longer one from above. This strategy is abbreviated MA in table 4.14 on page 193 which presents an overview of all results.

The basic neural network strategy is straightforward: one takes the forecasted rate of return, r_{t+1}. If $r_{t+1} > 0$ one goes or stays long. If $r_{t+1} < 0$ one goes or stays

short. Should $r_{t+1} = 0$ one would go or stay flat but that never happens. The basic neural network strategy is abbreviated NN.

This strategy is tentatively augmented by a threshold filter. If the forecast is not *decisive* enough one goes or stays flat. For the subsequent analysis one only trades, if $|r_{t+1}| > 0.005$. This is mostly intended to avoid jigsaw and trading when the probability to cover transaction costs is only low, see [110]. This strategy is abbreviated NNthr. One should expect this strategy to produce better risk measures. To be specific, it should improve sharpe ratio and maximum drawdown.

Finally, one would like to trade only if one expects the trend to go on. The multi step forecast offers the possibility to select only those trades where

$$\operatorname{sgn} r_{t+1} = \operatorname{sgn} r_{t+2}.$$

I.e., one wants the 2-day ahead forecast to go into the same direction as the 1-day ahead forecast. This double forecast strategy is abbreviated NNdbl. Again, one expects this strategy to show improved risk measures over the basic strategy.

Results for all assets and strategies are shown in table 4.14 on page 193. From an investing point of view one is first interested in the annualized return,

$$r_a = 252 \cdot \frac{1}{T} \sum_{t=1}^{T} r_t$$

for all relevant times. Of course, one excludes the training and validation data, i.e., the first 440 days. r_a^{tc} refers to annualized return with transaction costs subtracted. Here, 0.033 percent are taken for a round-trip. Transaction costs, of course, differ in various markets. But these costs are realistic for a market maker or a financial institution, see, e.g., [110].

The sharpe ratio is a risk adjusted measure of return. It compares annualized return with incurred volatility:

$$\text{sharpe} = \frac{r_a}{\sigma_a}$$

where the annualized volatility is given by

$$\sigma_a = \sqrt{252} \cdot \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (r_t - \overline{r})^2}.$$

The sharpe ratio may be interpreted as follows: a sharpe ratio inferior to zero sig-

nifies a negative return. A sharpe ratio greater than zero signifies positive returns with higher sharpe ratios being better.

Finally the maximum Drawdown, max DD, indicates which is the worst incurred loss before reaching new highs in equity. It is the smallest value of cumulated returns r_t which ever happens during the life of the strategy. The author highly recommends to have a look at further risk measures in [98], p. 33, from which the author takes the presented methodology of risk measures.

Additionally for information purpose the result table also shows the number of trades the strategy produces. A lower number of trades is generally preferred because of the transaction cost penalty. Each trade also involves the risk of slippage. However, with the advance of all-electronic trading, transaction costs are getting lower.

The information contained in the results table 4.14 on page 193 is very dense. For this reason the author starts with a general overview of salient features:

- In most cases the basic neural network strategy delivers more consistent and stable results when comparing the shorter *and* longer timespan. It does not necessarily beat the benchmark strategies on the shorter timespan but generally beats them on the longer timespan.

- The naive strategy achieves impressive results in the short timespan. One may attribute this to the inherent ability of this strategy to exploit trends — in this case the consistent downtrend following the collapse of the new economy which can be seen in most assets. However, the neural network strategy either follows as a close second best or even beats the naive strategy.

- The moving average strategy is often disappointing, both for shorter and longer timespan. It too, like the naive strategy, should work well in trending markets. But one sees that the strategy trades seldom. Trading activity is often only one fifth of that of the other strategies. While this is good in respect to transaction costs a closer look reveals that the moving average strategy simply misses out on important successful trades.

- Performance of the filtered neural network strategies compared to the basic neural network strategy is mixed. While they manage to reduce volatility the improvement in sharpe ratio, if any, if often only marginal. As trading activity if reduced to 60 percent compared to the basic strategy the filtered strategies,

like the moving average strategy, miss out on too many good trades. On the longer timespan, however, the filtered strategies often perform better.

Let us now look in detail at strategy results for asset classes and individual assets. In the case of equity indices the naive strategy starts with very impressive results for the short timespan on FTSE 100, DAX 30, CAC 40, FTSE MIB, Dow Jones, NASDAQ, and Kospi. It is however beaten by the neural network strategy in the case of NASDAQ and Kospi. In the other cases neural networks match the performance. Interestingly, the neural network strategies perform much better for S&P and Nikkei. In both cases the naive strategy fails to perform at all. The author stresses that this is surprising because both indices are perceived to represent very different markets.

Analyzing the longer timespan one notes very bad results for the naive strategy and bad results for the moving average. This is due to the fact that the longer timespan of 8 years encompasses very different market types which the simple strategies fail to adapt to. The neural networks, although not retrained, manage to at least break even in most cases. While this is, of course, not satisfactory one should keep in mind that in quantitative trading models are recalibrated regularly to catch up with different market dynamics. However, the point of the present analysis is to look at the flexibility of neural networks. And from this point of view the author deems it a succès d'estime that the networks manage to perform at all. One also notes that before transaction costs neural networks feature quite impressive returns even on the longer timespan for Nikkei and Kospi.

Comparing filtered and unfiltered neural network strategies one sees the threshold strategy sometimes leads to small improvements. This is especially the case for the FTSE 100 on the shorter timespan. In other cases returns are reduced. It is only with the benefit of hindsight that one would have been able to chose the better of the two strategies. The double neural network strategy which only invests on forecasted two-day moves generally produces inferior results than the basic strategy. However, it sometimes improves on the threshold strategy, e.g., for the NASDAQ. This improvement takes place with *significantly* fewer trades, e.g., 32 instead of 51. The incurred maximum drawdown is also reduced. Although this fails to improve the sharpe ratio the calmer trading style of the double strategy has its own merits. One concludes that it might be preferred by less active investors.

Performance on interest rates and yield curves is mixed — for all time spans and strategies. To be more specific: Neural networks decidedly outperform on 12

months LIBOR and the UK yield curve. The naive strategy delivers good short term results on the German and French yield curves. Moving averages perform well on US and Japanese yield curves. However, 3 months EURIBOR fails to be forecasted accurately by all strategies. Still, moving averages and neural networks realize marginal profits before transaction costs on the longer timespan.

Interestingly filtered neural network strategies manage to improve the performance on the *longer* timespan. E.g. the threshold strategy delivers satisfactory results on 3 months EURIBOR, the German, US and Japanese yield curve. Keep in mind that performance here is not measured in percent of invested capital but simply reflects basis points. The *real* performance depends on the kind of product one chooses to actually realize a view on interest rates.

Performance on currencies is clearly balanced towards neural networks. They outperform impressively on SFR|USD, EUR|USD and USD|JPY and come as a close second for GBP|USD. The naive strategy also achieves satisfactory performance on GBP|USD, SFR|USD and USD|JPY. It fails to perform completely on EUR|USD. On the longer timespan performance is low, sometimes slightly positive sometimes slightly negative for all strategies.

Commodities perform with mixed results. On the Gold Bullion the naive strategy slightly outperforms neural networks for the short time span. On the other hand all neural network strategies manage to produce positive returns before transaction costs on the longer timespan. In this case the filtered strategies clearly add value by reducing the number of trades significantly. Short term directional oil forecasts are decidedly — at least in the example — not for neural networks. They accumulate losses while moving averages impressively outperform. However, the filtered threshold strategy manages to produce positive returns before transaction costs on the longer timespan. The CRB index is forecasted best by the double neural network strategy in the short term. However, in the long term, the naive strategy slightly outperforms. Both strategies, which emphasize trends in their most basic form, gain from cyclical commodity markets which are indexed by the CRB. Finally the Baltic Dry Index performs very well for all strategies and all time frames. Obviously all strategies exploit the consistent trends in the index. From the point of view of quantitative investment the BDI might be an interesting asset to look at.

One may summarize the analysis as follows:

- By the consistency of their returns on short *and* long time frames neural networks are especially suitable for forecasting financial time series.

- However, simple technical strategies might prove lucrative, too. One sees that these simple strategies do *not* produce consistent returns in the long term.
- A very interesting application — possibly with neural networks as it is a pattern recognition problem — is therefore to determine *which* strategy to use. This is clearly beyond the scope of this work but further research on this topic should prove fascinating.
- One can clearly recommend the use of neural networks for forecasting equity indices. They consistently produce very satisfying and often best returns for all indices. This is not the case for the naive and moving average strategy.

Table 4.14: Comparison of different strategies. The naive and moving average or MA strategy are the benchmark strategies. NN refers to a basic neural network strategy, whereas NNthr uses a threshold filter. This avoids trades with low expected returns. The strategy NNdbl only trades if a *double* forecast, i.e., two consecutive days, promises to show identical sign of the return. r_a refers to annualized return, r_a^{tc} to annualized return including transaction costs. The first line of a strategy refers to a time span of 110 days, the second line is the result of using a single model for more than 8 years. Note that, although the NN strategy is seldom best, it shows *consistent* returns for most asset classes.

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						FTSE100
naive	0.437	0.421	2.55	-0.034	0.171	49
	-0.129	-0.494	-0.61	-0.870	0.213	1106
MA	-0.364	-0.368	-2.12	-0.154	0.172	13
	-0.057	-0.139	-0.27	-0.539	0.215	249
NN	0.124	0.106	0.71	-0.133	0.175	54
	0.047	-0.247	0.22	-0.438	0.215	891
NNthr	0.155	0.140	1.17	-0.100	0.132	44
	-0.047	-0.288	-0.26	-0.655	0.176	731
NNdbl	0.117	0.105	1.03	-0.078	0.113	35
	0.003	-0.181	0.02	-0.519	0.164	558

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						DAXINDX
naive	0.491	0.475	2.42	-0.065	0.203	49
	-0.127	-0.487	-0.48	-0.839	0.265	1091
MA	0.007	0.003	0.03	-0.104	0.206	13
	0.119	0.054	0.44	-0.372	0.269	197
NN	0.278	0.260	1.34	-0.151	0.207	53
	0.011	-0.315	0.04	-0.502	0.269	987
NNthr	0.111	0.095	0.65	-0.142	0.171	47
	-0.025	-0.315	-0.11	-0.569	0.234	881
NNdbl	0.204	0.193	1.39	-0.072	0.147	34
	-0.047	-0.240	-0.23	-0.595	0.199	586
						FRCAC40
naive	0.262	0.244	1.34	-0.087	0.196	54
	-0.130	-0.504	-0.52	-0.875	0.250	1134
MA	-0.028	-0.031	-0.14	-0.121	0.200	9
	-0.011	-0.078	-0.04	-0.566	0.252	205
NN	0.176	0.156	0.88	-0.117	0.200	61
	-0.002	-0.321	-0.01	-0.592	0.252	965
NNthr	0.181	0.166	1.09	-0.113	0.167	45
	0.039	-0.247	0.19	-0.375	0.210	867
NNdbl	0.022	0.012	0.17	-0.103	0.133	33
	-0.051	-0.234	-0.28	-0.626	0.184	556
						FTSEMIB
naive	0.276	0.258	1.71	-0.096	0.161	53
	-0.105	-0.474	-0.46	-0.840	0.230	1117
MA	-0.393	-0.400	-2.40	-0.170	0.164	21
	0.091	0.017	0.39	-0.437	0.233	223
NN	0.249	0.231	1.55	-0.062	0.161	57
	0.053	-0.277	0.23	-0.516	0.233	1002
NNthr	0.244	0.226	1.94	-0.046	0.126	53
	-0.004	-0.287	-0.02	-0.596	0.197	857
NNdbl	0.036	0.026	0.31	-0.094	0.113	28
	-0.030	-0.220	-0.17	-0.569	0.174	576

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						DJES50I
naive	0.409	0.390	2.08	-0.076	0.197	55
	-0.162	-0.533	-0.64	-0.902	0.254	1124
MA	0.020	0.016	0.10	-0.093	0.201	13
	-0.012	-0.088	-0.05	-0.640	0.257	229
NN	0.318	0.298	1.60	-0.119	0.199	61
	0.113	-0.212	0.44	-0.363	0.256	985
NNthr	0.173	0.158	1.04	-0.077	0.167	46
	-0.019	-0.308	-0.09	-0.661	0.220	875
NNdbl	0.129	0.118	0.93	-0.083	0.138	32
	0.003	-0.179	0.01	-0.398	0.189	550
						SPCOMP
naive	-0.085	-0.105	-0.45	-0.124	0.191	58
	-0.306	-0.693	-1.43	-0.946	0.214	1171
MA	0.051	0.047	0.26	-0.074	0.192	13
	-0.010	-0.080	-0.05	-0.619	0.220	213
NN	0.405	0.388	2.12	-0.076	0.191	50
	0.004	-0.325	0.02	-0.478	0.220	995
NNthr	0.146	0.133	1.11	-0.046	0.131	42
	-0.008	-0.293	-0.05	-0.444	0.184	863
NNdbl	0.239	0.228	1.70	-0.057	0.140	32
	0.005	-0.171	0.03	-0.303	0.159	535
						NASA100
naive	0.370	0.350	0.74	-0.189	0.503	60
	-0.178	-0.549	-0.60	-0.924	0.294	1126
MA	-0.196	-0.201	-0.38	-0.321	0.513	13
	0.001	-0.080	0.00	-0.550	0.303	244
NN	0.512	0.494	1.00	-0.237	0.512	54
	-0.119	-0.473	-0.39	-0.824	0.303	1073
NNthr	0.101	0.084	0.23	-0.244	0.443	51
	-0.130	-0.475	-0.47	-0.798	0.279	1044
NNdbl	0.173	0.163	0.44	-0.227	0.393	32
	-0.063	-0.237	-0.30	-0.574	0.209	529

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						JAPDOWA
naive	-0.376	-0.394	-1.50	-0.250	0.251	56
	-0.084	-0.446	-0.35	-0.646	0.242	1097
MA	-0.046	-0.050	-0.18	-0.166	0.261	10
	-0.018	-0.090	-0.07	-0.560	0.257	217
NN	1.107	1.090	4.39	-0.089	0.252	50
	0.211	-0.136	0.82	-0.448	0.256	1053
NNthr	1.034	1.018	4.97	-0.050	0.208	47
	0.203	-0.082	0.95	-0.364	0.213	864
NNdbl	0.524	0.513	2.81	-0.070	0.187	33
	0.027	-0.167	0.15	-0.413	0.179	589
						KORCOMP
naive	0.155	0.136	0.62	-0.144	0.249	58
	0.118	-0.228	0.46	-0.463	0.257	1049
MA	0.289	0.287	1.14	-0.114	0.254	6
	0.142	0.076	0.53	-0.355	0.266	201
NN	0.480	0.467	1.91	-0.085	0.252	41
	0.201	-0.091	0.76	-0.416	0.266	885
NNthr	0.407	0.394	1.66	-0.092	0.244	39
	0.129	-0.161	0.53	-0.474	0.244	880
NNdbl	0.182	0.172	0.91	-0.085	0.200	29
	0.142	-0.036	0.68	-0.510	0.208	540
						BBGBP12
naive	-0.027	-0.044	-0.23	-0.068	0.122	51
	0.272	-0.019	2.02	-0.264	0.135	883
MA	0.259	0.256	2.12	-0.039	0.122	9
	0.277	0.227	2.03	-0.273	0.136	151
NN	0.157	0.142	1.28	-0.068	0.123	47
	0.069	-0.221	0.50	-0.202	0.137	877
NNthr	-0.037	-0.039	-1.13	-0.022	0.033	7
	-0.006	-0.055	-0.17	-0.111	0.033	148
NNdbl	0.283	0.274	2.57	-0.029	0.110	26
	0.093	-0.090	0.81	-0.128	0.114	554

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						ECEUR3M
naive	-0.116	-0.134	-1.05	-0.129	0.111	54
	-0.133	-0.396	-0.89	-0.879	0.149	795
MA	-0.035	-0.037	-0.28	-0.060	0.124	5
	0.134	0.074	0.74	-0.671	0.181	181
NN	-0.191	-0.207	-1.54	-0.184	0.124	47
	0.057	-0.237	0.32	-0.318	0.181	893
NNthr	-0.040	-0.054	-0.38	-0.104	0.106	44
	0.064	-0.193	0.45	-0.232	0.143	779
NNdbl	-0.213	-0.221	-2.05	-0.145	0.104	26
	-0.004	-0.164	-0.03	-0.406	0.131	484
						UKyc
naive	-3.285	-3.305	-3.78	-0.842	0.869	62
	-1.125	-1.511	-1.07	-1.000	1.056	1171
MA	-2.244	-2.251	-2.55	-0.838	0.881	21
	-0.316	-0.406	-0.30	-1.000	1.061	270
NN	1.419	1.399	1.59	-0.436	0.893	61
	0.435	0.056	0.41	-1.374	1.061	1148
NNthr	1.814	1.795	2.11	-0.336	0.859	59
	0.386	0.011	0.37	-1.302	1.041	1137
NNdbl	1.537	1.528	2.61	-0.220	0.589	29
	0.359	0.165	0.45	-2.374	0.796	587
						GERyc
naive	0.885	0.868	1.35	-0.252	0.656	51
	-0.391	-0.751	-0.51	-0.999	0.771	1093
MA	-0.174	-0.178	-0.26	-0.437	0.657	13
	0.367	0.297	0.48	-0.869	0.772	212
NN	-0.127	-0.147	-0.19	-0.464	0.658	59
	0.003	-0.306	0.00	-0.987	0.772	937
NNthr	-0.119	-0.138	-0.19	-0.487	0.643	58
	0.153	-0.154	0.21	-0.952	0.745	930
NNdbl	-0.033	-0.042	-0.07	-0.260	0.457	25
	0.047	-0.124	0.08	-0.801	0.561	520

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						FRyc
naive	0.548	0.531	0.81	-0.326	0.680	51
	-0.370	-0.732	-0.48	-0.999	0.764	1096
MA	-0.255	-0.259	-0.38	-0.422	0.679	11
	0.262	0.186	0.34	-0.952	0.765	230
NN	-1.087	-1.106	-1.61	-0.567	0.677	57
	-0.126	-0.447	-0.16	-0.986	0.765	973
NNthr	-0.309	-0.328	-0.52	-0.399	0.599	56
	-0.126	-0.448	-0.17	-0.982	0.733	971
NNdbl	-0.848	-0.858	-1.71	-0.403	0.496	29
	-0.116	-0.301	-0.21	-0.947	0.542	560
						ITyc
naive	-0.452	-0.473	-0.71	-0.572	0.639	65
	-0.169	-0.533	-0.23	-0.997	0.743	1101
MA	-0.777	-0.784	-1.21	-0.484	0.641	21
	0.308	0.233	0.41	-0.903	0.749	229
NN	-1.069	-1.088	-1.67	-0.604	0.640	59
	-0.069	-0.364	-0.09	-0.996	0.749	895
NNthr	-1.159	-1.177	-1.87	-0.627	0.619	56
	0.082	-0.209	0.11	-0.988	0.722	883
NNdbl	-0.481	-0.490	-1.03	-0.353	0.466	26
	-0.036	-0.201	-0.06	-0.947	0.562	501
						USyc
naive	-2.252	-2.272	-1.97	-0.791	1.144	61
	-0.850	-1.213	-0.66	-1.000	1.279	1100
MA	0.210	0.204	0.18	-0.816	1.172	18
	0.647	0.579	0.50	-0.989	1.287	208
NN	0.126	0.107	0.11	-0.655	1.168	57
	-0.048	-0.392	-0.04	-1.000	1.288	1043
NNthr	-0.871	-0.889	-0.79	-0.671	1.097	54
	0.208	-0.135	0.17	-1.000	1.246	1040
NNdbl	0.041	0.030	0.05	-0.392	0.800	32
	0.159	-0.026	0.17	-0.997	0.928	559

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						JAPyc
naive	0.048	0.030	0.10	-0.244	0.476	55
	-0.655	-1.027	-1.03	-1.000	0.635	1127
MA	0.336	0.334	0.66	-0.231	0.512	8
	-0.283	-0.366	-0.44	-0.991	0.649	251
NN	-0.431	-0.452	-0.84	-0.306	0.511	62
	-0.015	-0.366	-0.02	-0.951	0.649	1064
NNthr	-0.002	-0.023	-0.00	-0.264	0.478	61
	0.160	-0.189	0.26	-0.877	0.618	1059
NNdbl	-0.632	-0.640	-1.87	-0.286	0.337	23
	0.033	-0.132	0.07	-0.873	0.483	501
						USDOLLR
naive	0.074	0.058	1.02	-0.031	0.072	48
	0.057	-0.289	0.58	-0.282	0.097	1046
MA	-0.158	-0.164	-2.20	-0.090	0.072	18
	-0.007	-0.095	-0.07	-0.264	0.098	268
NN	0.014	-0.002	0.19	-0.049	0.073	49
	0.010	-0.310	0.11	-0.312	0.098	970
NNthr	0.043	0.039	2.28	-0.008	0.019	12
	0.005	-0.106	0.10	-0.111	0.048	335
NNdbl	-0.023	-0.033	-0.46	-0.022	0.050	30
	0.032	-0.155	0.42	-0.183	0.075	567
						SWISFUS
naive	0.124	0.106	1.32	-0.038	0.094	53
	-0.071	-0.455	-0.65	-0.546	0.109	1165
MA	0.005	0.001	0.05	-0.078	0.098	12
	-0.062	-0.156	-0.56	-0.485	0.111	285
NN	0.144	0.125	1.47	-0.037	0.098	57
	-0.043	-0.368	-0.39	-0.461	0.111	984
NNthr	0.088	0.075	1.46	-0.040	0.060	39
	-0.032	-0.254	-0.45	-0.296	0.072	671
NNdbl	0.019	0.010	0.33	-0.038	0.059	29
	-0.033	-0.210	-0.42	-0.306	0.079	537

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						USEURSP
naive	-0.019	-0.036	-0.18	-0.071	0.105	52
	0.002	-0.362	0.02	-0.253	0.099	1102
MA	-0.051	-0.056	-0.48	-0.093	0.106	16
	0.026	-0.050	0.26	-0.212	0.100	229
NN	0.168	0.155	1.60	-0.033	0.105	39
	-0.011	-0.307	-0.11	-0.375	0.100	898
NNthr	0.080	0.071	1.32	-0.021	0.061	29
	-0.028	-0.225	-0.41	-0.346	0.067	598
NNdbl	0.207	0.199	2.42	-0.023	0.085	25
	0.001	-0.172	0.02	-0.243	0.077	524
						JAPAYEUSD
naive	0.069	0.050	0.73	-0.036	0.095	58
	0.036	-0.325	0.35	-0.230	0.103	1093
MA	-0.056	-0.060	-0.58	-0.084	0.096	12
	0.014	-0.058	0.13	-0.192	0.104	219
NN	0.138	0.122	1.44	-0.073	0.096	50
	0.045	-0.272	0.43	-0.146	0.104	961
NNthr	0.137	0.128	2.43	-0.030	0.056	27
	0.036	-0.155	0.50	-0.166	0.072	579
NNdbl	0.092	0.084	1.22	-0.054	0.075	24
	0.001	-0.171	0.01	-0.189	0.079	520
						GOLDBLN
naive	0.107	0.091	0.96	-0.055	0.112	50
	-0.031	-0.395	-0.17	-0.433	0.183	1104
MA	-0.117	-0.121	-1.00	-0.141	0.116	14
	0.008	-0.062	0.04	-0.384	0.190	211
NN	0.064	0.048	0.55	-0.084	0.116	47
	0.008	-0.289	0.04	-0.544	0.190	900
NNthr	-0.005	-0.017	-0.05	-0.060	0.097	38
	0.072	-0.151	0.51	-0.390	0.140	675
NNdbl	0.169	0.159	2.54	-0.020	0.067	31
	0.065	-0.127	0.43	-0.316	0.150	581

Continued on next page

strategy	r_a	r_a^{tc}	sharpe	max DD	volatility	trades
						OILBREN
naive	0.068	0.049	0.27	-0.121	0.256	58
	-0.083	-0.436	-0.23	-0.882	0.362	1069
MA	0.447	0.444	1.72	-0.118	0.261	10
	0.140	0.066	0.38	-0.637	0.369	226
NN	-0.560	-0.579	-2.15	-0.250	0.260	57
	0.071	-0.270	0.19	-0.748	0.369	1034
NNthr	-0.411	-0.431	-1.69	-0.203	0.243	56
	0.120	-0.210	0.36	-0.593	0.336	1001
NNdbl	-0.122	-0.132	-0.63	-0.136	0.195	31
	0.048	-0.136	0.18	-0.447	0.265	557
						NYFECRB
naive	0.011	-0.007	0.14	-0.038	0.082	55
	0.083	-0.261	0.60	-0.202	0.139	1044
MA	0.083	0.080	0.99	-0.038	0.084	9
	0.071	-0.000	0.50	-0.227	0.144	217
NN	0.062	0.044	0.74	-0.054	0.084	55
	0.007	-0.292	0.05	-0.349	0.144	906
NNthr	0.031	0.021	0.64	-0.022	0.049	31
	0.033	-0.164	0.35	-0.132	0.095	598
NNdbl	0.097	0.087	1.48	-0.028	0.065	30
	0.026	-0.169	0.23	-0.234	0.114	591
						BALTICF
naive	1.343	1.337	16.12	-0.004	0.083	16
	2.474	2.371	10.54	-0.072	0.235	312
MA	0.960	0.959	9.39	-0.037	0.102	5
	1.507	1.485	5.62	-0.196	0.268	68
NN	1.126	1.121	11.79	-0.028	0.096	15
	1.876	1.773	7.26	-0.206	0.259	312
NNthr	0.805	0.801	9.16	-0.022	0.088	12
	1.686	1.599	6.99	-0.177	0.241	264
NNdbl	1.081	1.075	11.68	-0.022	0.093	14
	2.054	1.953	8.48	-0.107	0.242	304

4.9 Summary

This chapter provides a detailed analysis of a diverse portfolio of assets. Examples exploit the three key benefits of shared layer perceptrons:

- The ability to provide a market forecast, i.e., several assets are forecasted with the *same* model and at the *same* time.
- Multi step forecasts offer the possibility to generate a view on the probable path of assets or portfolios.
- The expert topology leads to a *distribution* of forecasts which can be conveniently used, e.g., to assess risk.

The chapter analyzes assets from different instrument classes: important equity indices, interest rates and yield curves, currency exchange rates, and commodities. The complete dataset includes a timespan of 10 years: from July 1999 to July 2009. The inclusion of more exotic assets like the Korean Kospi or the Baltic Dry Index for freight rates proves insightful and also very attractive regarding performance. The analysis at the beginning shows that the level series are non stationary. An appropriate transformation into rates of return provides stationarity. The Jarque Bera statistic confirms that all return series are not normally distributed. Autocorrelation is generally low. The dataset is representative of different market situations: the boom and bust of the new economy, the bull market until 2007, the credit crisis and the subsequent recovery. The analysis does not suffer from being biased towards a specific kind of market situation.

The first two examples are unique in the sense that they benefit especially from the shared layer perceptron model. The author does *not* claim that these examples would be impossible with other modeling techniques — but building the model would be considerably more work and considerably less intuitive!

The first example models market value at risk. The idea is to provide a more adequate risk measure than standard techniques like, e.g., historical simulation. With the shared layer perceptron it is possible to model an asset value 10 days ahead. Then, one can choose an appropriate percentile of the distribution of experts. The shared layer perceptron manages to *always* utilize less capital than the historical simulation for every asset on a timespan of 110 days. On the longer time frame of more than 8 years the shared layer perceptrons still outperforms for 12 out of 17 assets considered in the example. Please note, that the shared layer perceptron is

not retrained. I.e., a single model manages to catch the dynamics very well. The author concludes that neural networks add value when modeling the risk of a portfolio. However, care should be taken to assess possible violation of regulatory risk measures.

The second example features a typical but complex purchasing decision problem: one has to buy some supply, some asset, regularly. In the example the author uses a monthly basis. Obviously, one wants to buy for the lowest price achievable in that month. Where the first example deals with accurately forecasting the lowest *level* of a portfolio this second example involves forecasting the *time* when this lowest level will occur. Again, shared layer perceptrons beat *every* fixed day strategy on a 110 day timespan. And on the longer timespan of 8 years shared layer perceptrons still beat fixed day strategies in the overwhelming majority of the cases. And keep in mind, that the networks do not produce any especially bad realized potential. One can therefore be confident of the robustness of the model.

The third example is more typical for neural network applications: forecasting the direction of change of financial time series. This works very well compared to benchmark technical strategies for equity indices and currency exchange rates. Results on interest rates and commodities are mixed but still overall satisfying. In some cases filtered neural network strategies improve the results. Especially the filter strategy which takes into account the sign of the return on two consecutive days produces consistent, but lower, performance with significantly reduced trading activity: on average it produces only one or two trades per week.

The author will not hide that not all potential of shared layer perceptrons has been exploited. Further research will address the following questions:

- How do the models perform when the networks are regularly retrained? Is it sensible to even retrain *continuously*?
- How far can one extend multi step forecasts? Can one enhance multi step forecasts when using additional information from another time step width, like a weekly model?
- What does the distribution of returns tell us about the probability of the realization of a single forecast? Does the information of distribution width and height provide additional value?
- How dependent is the model on portfolio choice? Would performance change when including more or less assets?

5 Conclusions and Outlook

5.1 Introduction

This chapter wraps up this book. The reader finds a summary of results in the following section. But the chapter is more than that. Section 5.3 provides a critical review of parts that are still missing. The author highlights further research areas in section 5.4. Note that much of the content of this section directly draws from the critical assessment section. That is quite natural: incomplete parts are likely to induce more research. Section 5.5 provides concise recommendations for decision makers and implementers. This is not intended as executive summary. The reader will find this summary *at the beginning* of this book on page 18. Finally, the last section concludes this book with some thoughts.

5.2 Summary of Results

My research question is «Can advanced neural networks provide sustainable and economic competitive edge in today's financial markets?» Instead of merely repeating the end of chapter summaries the author will follow his research question and highlight some keywords. The following bold headlines come from the research question.

advanced neural networks

This book presents and analyzes in depth the shared layer perceptron topology. This topology models a memory enabled dynamic system evolving in discrete time steps. It's key feature is that it models several time series *at once*. Unobserveable states of the world are approximated through *hidden states*. One forecasted step, i.e., output serves as input for the next forecasting step. Multi step forecasts are

possible. One can see an unfolded shared layer perceptron as a multi layer perceptron with the same number of hidden layers as there are history time steps. However, the main difference is that each layer *shares* the same weights. The shared layer perceptron is more manageable than a standard multi layer perceptron of same depth.

Generally, training neural networks necessitates the first partial derivatives with respect to the weights. One can calculate this derivatives efficiently using reverse accumulation and matrix algorithms. The shared layer perceptron uses the optimization algorithm implemented in the neurosimulator FAUN, i.e., an efficient sequential quadratic programming method.

sustainable

The model is robust in two ways. First, it works well over a broad range of financial assets. The author demonstrates that for all three applications results on equities, interest rates, currency exchange rates, and commodities, are good or very good. This holds compared to benchmark strategies. But the model also produces very satisfactory results on an absolute scale. In the author's opinion the crux of most otherwise presented models is that they are only trained for *a single market*. While the results indicate good performance in this market they say nothing about other markets or even only other assets in the same market. The author can't emphasize this enough: One will trust an overall good model more than a model fitted — or overfitted? — to a single case.

Second, the shared layer perceptron obviously manages to catch the dynamics of shifting markets well. The author demonstrates this by comparing forecast results for a period of 110 days with a period of more than 8 years. Not unsurprisingly results on the longer time span are noticeably worse than on the shorter time span — but not abysmally so! Where benchmark strategies on the longer time span show significant variance in performance the shared layer perceptron's performance degrades gracefully. Again: this should strengthen trust in the quality of the model. One will not believe that model performance is due to a statistical fluke.

economic

Training the shared layer perceptron is possible without specialized hardware. One doesn't need massively parallel high performance computers. Instead, the grid

computing client puts spare computing capacity to work. In an organization where user workstations are available anyway it is trivial to install the client. Using wake up and shut down mechanisms controls power consumption efficiently.

The grid computing version of FAUN produces good speedups of more than 95 percent. This is in line with what the MPI and PVM versions achieve. However, using the grid computing client does not require any special setup on the host computer or in the network. Installing and using the client is very economical: as long as the client reaches the server it will work.

FAUN is also prepared for fine grained parallelization. Although this does not scale as well for more than four processes it is very promising for usage on graphics cards.

competitive edge

The author shows that the model generally outperforms benchmark strategies. This holds especially for modeling market value and risk and the transaction decision support. Using the shared layer perceptron topology *adds* value. The shared layer perceptron also offers applications, like multi step forecasts, which are not easy to model otherwise.

today's financial markets

The author shows that his model works well on past *and* recent data, even without retraining. The applications the author presents cover a broad range of what today's financial markets require.

Modeling market value at risk is a topic of risk management. For a financial institution it is of utmost importance to have a model that provides a *realistic* assessment of portfolio value over the next few days. Standard computation methods like, e.g., historical simulation are often too conservative in their assumptions. Holding back capital unnecessarily leads to lost opportunities.

Corporate treasurers face periodic purchasing decisions: they purchase an asset — tangible or intangible — on a regular basis, say, every month. The challenge is to buy this asset for the lowest price in the purchasing period. The shared layer perceptron easily models portfolio course over 20 days. This allows to select the entry point with good accuracy.

The third applications consists in day to day investment decisions: should one

buy — or rather sell — an asset. Here, the *sign* of forecasted returns is of importance.

5.3 Critical Assessment

Although the author is personally satisfied with the results of this book there are several aspects which warrant a critical review. These may be just simple questions that could be answered with relative ease given time. Or they may be unknowns which could shake the foundations of what the author presents. The author first assesses every main chapter individually. Then the author gives a comprehensive assessment of critical points linking all chapters.

5.3.1 Grid Computing

The performance of fine grained parallelization on multi processor computers is disappointing. While the author expected some decrease in speedup due to high latency the author is surprised to see such an amount. Clearly, more research is warranted here. The author is convinced that programming the algorithm «closer to the metal» with an in depth knowledge of processor architecture would increase the result considerably. Also, OpenMP does not allow very fine tuned control on how exactly it parallelizes computations. Again, programming more closely to the underlying hardware should improve speedup.

Fine grained parallelization is a *par excellence* topic for programming on the graphics card. The author tried this and the emulated results look promising. However, the author hadn't a powerful graphics card available. Graphics card programming would also have introduced a whole new layer of complexity. The author decided to avoid this for the present book. However, using standard graphics hardware for accelerating computation matches well with the general tenor of this book. It is unfortunate that the author didn't have more time to further inquire in this direction.

The coarse grid computing client is still not as flexible as the author would like it. The necessary general functions like file transfer, system calls, wake up, shutdown, etc. are indeed generically usable. But the entire product is still very much tied to FAUN. On the one hand there is no need to provide superfluous flexibility. On the other hand the author would have been glad to publish the program as a universal

grid computing client. One *can* use it like this. But it would be nice to have more examples.

It is somewhat unsatisfactory that until now no automatism exists that evaluates how an additional computer in the grid influences speedup. As a general rule network bandwidth is more important than clock speed. But it would have been nice to analyze this in more detail than the scenarios. Here, the book lacks a systematic empirical test. Actually, the author would like a theoretic underpinning. But the author is not at all sure if this is realistic with constantly varying hardware.

Administration of the grid participants occurs via the command line. This is unproblematic for seasoned system administrators. But nevertheless the present program lacks a user friendly graphical interface for managing the grid, except for shutdown and wake up. At least the update functionality should be accessible using the web interface. This would help in spreading the work further.

The architecture does not use bandwidth very efficiently. It is obviously better to collect results *locally* and send them back as bundle. This introduces an additional layer of complexity and also makes management more difficult. However, there might be arrangements, like local clusters, where this would increase speedup. Using data compression would also improve bandwidth use. In principle this should not be difficult to implement. The distributed object model is layered. It is possible to insert an additional compression layer.

Another concern is that data is sent every time a new computation starts. This is not important for production runs, because training then lasts longer. But for test runs the client spends a comparatively high amount of time in sending the data. Avoiding this is easy to implement by using a hash, e.g., the md5 hash. The MPI version of FAUN already has the same inefficient bandwidth use. It would be good to resolve this once and for all. However, the author does not consider this a key problem and lacking time prevented an implementation.

Along the same lines as above goes another concern. For each new computation FAUN will rescale the data as necessary. This holds for all versions of FAUN. Normally, scaling does not consume significant time and for the present book this was not a problem. However, several months ago the author worked with a dataset of several GB. With this dataset scaling takes three minutes. It would be nice to avoid this unnecessary waiting time. Again, using a hash would solve this easily.

Security is also an issue. The client uses ssl to secure communication which is good. However, the client stores training and validation data, and results unen-

crypted on the local hosts. It is therefore possible for users with the appropriate rights to *view* and *possibly modify* the data. This is a systemic problem, because at some stage data has to decrypted. One could modify FAUN such that it reads data directly from memory and not from file. But even then unencrypted data *is* in memory. This makes it slightly more difficult for the casual attacker to get access. One has to be aware of the fact that whoever owns the computer may look at data. One could slightly mitigate this by taking two separate virtual machines: one for the grid client and one for the local user. But the basic problem remains.

5.3.2 Neural Networks

The saying goes that building and training neural networks is «more an art than a science». While the author likes to be an artist — who wouldn't — this raises serious concerns about *what* the neural networks do and *how* they do it. The shared layer perceptron topology mitigates these concerns. But nevertheless for a given set of meta parameters one can only prove convergence *empirically*, on a case by case basis.

The essential question remains: which meta parameters are appropriate for the application at hand? The author found the parameters by informed experimentation — or bluntly put: by trial and error. This is far from satisfactory and not user-friendly at all. A unified theory about how to best choose the parameters will — probably — not emerge in the foreseeable future. At least one would like an automatism setting sensible parameters for the application. The FAUN project group has not been inactive in this respect. But the problem is really *difficult*. It necessitates incorporating prior knowledge about successfully trained neural networks. Then one needs to map this to a slightly different problem. One could call this a meta neural network for parameter setting.

As a related topic the author would have liked to analyze convergence in much more detail. It is satisfying to see such robust convergence on a financial time series dataset. However, a thorough analysis should include datasets from different sources. It should also vary parameters about a much broader range. And there are still other parameters which the author didn't even mention until now. But they could also influence convergence. These include:

- The choice of single versus double precision. The author found no significant difference during first tests. But, again, bigger datasets might well *need*

double precision.

- The line search tolerance. NPSOL allows to set a granularity for the line search. As high dimensional neural network functions can be very rugged choosing a smaller line search parameter might even influence convergence negatively. Experiments show no difference. But again, this is more anecdotal and not universal.

- Optimization options of the compiler. In this book the author only uses the Intel compiler on Intel systems. Except for debugging purposes the author uses the `-fast` option which provides sensible optimization without being overly aggressive. Using more aggressive parameters does not influence convergence in first experiments. However, I'd like to verify this in more detail.

- Different compilers. While I'm very satisfied with results from the Intel compiler, the author is very interested in trying the compiler from the Portland Group. This compiler now offers auto parallelization on Nvidia graphic cards. The question is, if the algorithm really translates to the GPU and how this affects convergence.

The shared layer perceptron offers, in principle, a white box — or at least gray box — approach because it uses only a *single shared* matrix. A look at the upper left $N \times N$ square should prove interesting. These numbers tell us how the immediately previous time step affects the immediately next time step. Due to sparsity most entries will be zeroes but the remaining could be informative. One could also extend the analysis to include *all* previous states that influence the present observeables. The question remains how to interpret the matrices for an expert topology.

5.3.3 Financial Applications

Every model faces a fundamental challenge: how will it stand the test of time? The author thinks to have demonstrated the robustness of the approach. However, the models have not been traded. When trading a model a host of catches can appear:

- The tradeable price deviates from the modeled price. A model may correctly forecast a price, for a certain day, and this price also appears. But it is not tradeable. One gets slippage or volume is not sufficient. This affects especially the transaction and investment decision support applications.

- Retraining takes too long. In a live application one would typically retrain the model when new data is available. One may not be able to do this in real time. For the shared layer perceptron the minimum time interval is 20 minutes. When using minute or tick data one would not be able to retrain for every new data. This must not be a serious disadvantage. But one has to be aware of it.

- Hardware fails. The model is entirely quantitative. There is no «guts» feeling in it — at least not in the forecast. One is heavily dependent on reliable hardware, which may not be available. Think about it: one needs a cluster, a redundant internet connection for data supply and an uninterruptible power supply. The shorter the time frame the more important are these three points.

- Software fails. One may not be able to transmit the model's decision in adequate time to the broker.

- Markets «fail». Obviously, no model can account for outside shocks. One may be caught in the wrong position with no possibility to get off.

All these are very valid concerns. One can only evaluate the importance of each by trading the model — with *real* money. Only with real money will one encounter realistic slippage and unfilled orders. Unfortunately the author didn't have the capacity to put together a live model for this book. As an aside: if *you* are interested in trying the model live, do tell the author by all means.

Related to the above concerns is the topic of money management in all three applications, but especially in the investment decision application. Normally, one would protect positions with a stop loss. This will — probably — prevent catastrophic drawdowns. But it will — probably — also prevent good trades. The author didn't analyze the effect of money management rules.

It is not at all clear what happens when the time scale of the data changes. Will the shared layer perceptron also work well for weekly data and for tick data. Only empirical tests can reveal what is likely to happen. The author didn't use other timescales than daily. Both — shorter and longer — timescales have their own set of problems associated with them:

- Longer timescales face increasing data sparsity. It is generally not a problem to get sufficiently long history, i.e., a few hundred time steps, for weekly data. However, the picture changes quickly when going to monthly or even yearly data. Most data of interest will only be easily available since the 1980s with

exceptions starting in the 1960s. One may be lucky and get — some — data since the inception of, e.g., the New York Stock Exchange in 1792 or predecessors of Frankfurter Wertpapierbörse in 1150, but that seems rather esoteric. In principle a shared layer perceptron of appropriate size should put all available data to good use. But that remains to be seen.

- Shorter timescales face increasingly erratic behavior. Studies on market microstructure generally agree that finding a suitable price may take up to one or two minutes, even in a liquid market. When feeding and training the shared layer perceptron with ultra high frequency data it may well be that the network merely learns noise. Using validation data will cause early stopping at high error levels and the information content of the network may be nil. This problem is not specific to the shared layer perceptron not even to neural network. But the ease of use of the shared layer perceptron may lead to abuse. In any case, only thorough empirical studies can demonstrate the behavior at short timescales.

When data is scarce the question arises: which amount of data is sufficient for successful training? Clearly, one will reduce the number of weights. But can we, e.g., use only 10 data points of yearly data and make sensible forecasts? The opposite question is also of interest: what is the highest sensible amount of time steps one can use? Adding more time steps will increase training time but should also produce better forecasts. However, there should be a region in which adding more data only marginally increases forecasting performance, if at all. This region will also be different at various timescales. The shared layer perceptron topology still misses a comprehensive analysis of this entire topic.

In this book the author uses daily data. Each new data point is considered to represent «one time step». Even daily data shows irregularities: one has weekends and holidays. So one is, in fact, not using a daily timescale but a — crude — volatility timescale where periods of no trading — hence no volatility — are left out. With higher frequency data volatility time scales gain in importance. Only currencies trade around the clock. Other assets pause during the night or even do not trade in the early morning and late afternoon. The author didn't analyze the effect of a volatility time scale at all. However, when using higher frequency data one almost certainly will find differences between, e.g., strictly hourly data and the same amount of data based on a volatility time scale. Not connected to neural networks

but yet very interesting is the following question: how can one construct a volatility time scale for *several* assets which trade around the clock in *different* timezones? To the best of the author's knowledge there still is a research deficit here. One notices that time zone considerations already affect the model on daily data. This will be much more accentuated when using higher frequency data.

In short: the evidence that the model works is only empirical. It is very unlikely that one will one day have a unified theory on how markets work. But in the meantime much more robustness tests of the shared layer perceptron are needed. One can neither *prove* that other classical models *work*. But they have stood the test of time. The author hopes that the shared layer perceptron, too, will be much used.

Another criticism of the work also holds: the analysis could have been even deeper. In the present form it does not extract all information that the shared layer perceptron provides. This concerns especially the *distribution* of returns. In first tests the author already noticed that distribution width varies among assets. What does this tell us about the probability of the realization of a path? Can one be more confident when distribution is narrow? And should one be wary if distribution is wide?

Related to this the author also does not exploit the potential of the *expert topology*. An interesting question remains: how much experts are enough? Also, the author does not provide a method — except training error — to select the experts. Possibilities include to select experts based on past performance. This allows to give experts with good out of sample results in the past a more important weight. It also creates a *third* type of memory: the — hidden – states are the short term memory, the weights the long term memory and the *history* of weight changes during retraining incorporates the *very long term* memory. The author does not know of any reference analyzing this aspect. But it is an obvious next step for the shared layer perceptron.

The author doesn't investigate the use of dummy variables. These variables like day of the week, day of the month, time of the day, and so on may play a significant role in improving forecast quality. Incorporating them does not even lead to increased training times, because the shared layer perceptron does not have to *forecast* them. It can simply extrapolate the time series from known rules. Also, the author doesn't research the effect of *derived* variables. Surely, technical indicators provide *different* views on time series. Take an oscillator. There are two possibilities: one trains the network to also forecast the oscillator. Then one can

check if forecasted return and forecasted oscillator match. If they don't one should be wary. Or: one does *not* forecast the oscillator but compute it from past time steps and use this to — possibly — improve forecasts of future time steps. There is very much space for creativity here! An even more esoteric application is: include weather data and use professional weather forecasts for future time steps. This could yield some predictive lift at least on weather dependent commodities.

Is it realistic to assume that every time step represents the same time interval? In the present model the weekend transition from Friday to Monday is essentially the same as any other transition during the week. This is linked to the problem whether wall clock time is an adequate representation of time. Volatility based time scales provide an alternative. However, considering assets from different time zones it remains unclear how one can construct a unified volatility time scale.

The author doesn't consider the problem of missing values because it does not occur in the dataset. The shared layer perceptron provides a simple way to treat missing values. One simply excludes them from error computation. Other studies set missing values to zero. The author considers this to be a potential source of errors. However, to be sure, the author would have to compare the two approaches. Missing values also offer the possibility to use data of different timescales, e.g., daily and monthly data. One can inject monthly data on the day it becomes available and treat the other days as missing. Does this improve forecasts? It gives one the opportunity to include general economic trends as opposed to daily market jitter in the model. I'm thinking of, e.g., consumer price indices, jobless numbers, GDP, and others. It is not sure that these numbers significantly influence short term movements.

My last criticism is more general and concerns the coherent market model. Right now the author is modeling an *entire* world market. The model proves to perform well on each asset. If one is interested in a single asset shouldn't one build an *even better* model just for this asset? Does a dedicated model work better? Personally the author has more trust in a general robust model than in a model which one cannot check on other assets. But the author concedes that this needs a thorough empirical study.

5.3.4 Comprehensive Assessment

Looking at this book at a whole the reader will see that a framework linking the steps from idea to model implementation is still missing. Such a framework would include a step by step guide to facilitate the process of model building. It would also help in taking away the fear from implementing a complex solution as the author describes it in the three main chapters. Still, the author gives some concrete recommendations in section 5.5.

The problem with presenting a unifying framework is that it links several professions. It has to *speak* the language of each profession:

- Typically, a *corporate treasurer*, a *portfolio manager* or a *trader* generates a view. Here, the process starts. The view is a description in words of some market phenomenon and how one would like to exploit it. Are you involved with risk management, with commodity purchasing, with trading?

- Then you transmit the idea to a *modeler*, probably an *applied mathematician* or an *engineer*. She puts the idea in the form of an algorithm. She probably also clarifies some points that are vague in the original formulation.

- A *programmer* implements the algorithm. He has to have a solid grasp of how to interface with price databases for back testing.

- It may well be that one needs additional hardware to implement ideas. Then one turns to a *hardware engineer*. She specifies a system. She will also mention, e.g., uninterruptible power supplies, backup space, failover hardware, redundant connections, etc.

- You will buy the required system at your trusted *vendor*. Here you must think about service level agreements. E.g., do you want your system to run 24/7? This will cost you.

- Now, your system is up and running. But it needs continuous maintenance. Your *system administrator* cares for your applications and computers running smoothly. She is especially useful when you are using a grid solution similar to the one the author mentions in this book. There *will* be problems with users interfering with the system. Computers will break and you need someone to fix them.

- Finally, a grid solution is not compatible with every organizational policy. You will have to talk to *department heads*, the *chief information officer*, and the *chief information security officer* to pull through the grid concept. When you are involving user's personal workstations the *data security officer* will also have his say.

Keeping track of the number of different professions involved one sees that there are typically *seven* different groups to talk to. Each group has its own interests. And each group has a different view of the world. Implementing a neural network model with grid computing involves the *entire organization*. Some groups may be represented by the same person. E.g., modeler and programmer are often the same. System engineer and administrator are nearby competences.

One can only implement parts of what the author suggests. E.g., one could use the shared layer perceptron without an organization wide grid. This already reduces some hassles. Or one can, as a treasurer, use a modeling software which interfaces to the shared layer perceptron and deploy the model yourself. One looses some flexibility but it simplifies the process.

In any case one still *needs* a framework which defines roles and interfaces. It is not the focus of this book to develop this framework. The author clearly put technical, mathematical and application oriented aspects in the foreground. The author acknowledges that the organizational aspect is simply missing. The author also acknowledges that dealing with the organizational aspect is key to implementing the ideas in a real context.

Another point concerns user friendliness. The software should facilitate access to the underlying neural network model. With the web interface this is easily possible, without local install. However, the web interface is very sober. It does not particularly engage the user. It is also not interactive. The author's vision is to augment the web interface with, e.g., Web 2.0 techniques. Don't get the author wrong. The sober interface should still persist for experienced users. And I'm not a fan of everything moving and gliding on a web page. But the graphical user interface *is* important. Period. It is the first encounter with the application. And if one doesn't help first-time users they will drop an — otherwise good — application.

Another idea is to realize a local grid enabled client. One would have all the power of local applications at your fingertips but still the grid engine for coarse grained parallelization. This, however, raises new issues. One has to think about lost connections and concurrency much more intensively than when every job starts on the

server. Both software engineering topics are beyond the scope of this book. They are nevertheless very important to spread neural network usage.

To summarize all of the above the author thinks that the following unanswered questions are the most important. The author plans to address these first in further research.

- How do you get from an *idea* to grid enabled predictive modeling with neural networks?
- How robust is the model when used for *real* decision support?
- Is a general model better than a specialized model?
- How should you best select input time series for a shared layer perceptron? Can you automate this process?

5.4 Further Research Areas

The critical assessment section already outlines aspects where more research is warranted. The author will now highlight the most important aspects of every chapter. Where appropriate concrete steps are described.

In the author's opinion fine grained parallelization on the graphics card has a bright future. The next logical step is to experiment with the FAUN code on a *real* graphics card. This is a little bit complicated by the fact that GPU enabled FORTRAN compilers have just started to appear. Otherwise it would be necessary to port the source code to C. While this is certainly possible the author would like to avoid this. The author is confident that the market of GPU enabled compilers will grow and that GPU usage will be further simplified. The author thinks that it is realistic to have a satisfactorily working version of FAUN on the graphics card by mid 2010. Concerning expected speedup training neural networks falls into the category of good-natured applications: once the computation is started no input or output is necessary until the network is trained.

The potential of the shared layer perceptron topology is far from being exploited. From the range of possible research the author would choose exploring properties of the distribution. The author suggests to start with data used in this book. Then, one could analyze different percentiles of forecasted distributions. This should enhance quality of the forecast by providing confidence bands.

Another interesting point concerns data intervals. The author would especially like to research how the shared layer perceptron copes with intraday data. This introduces a new challenge because even hourly data is not available at every time of the day. Consider, e.g., assets which are only traded in certain time zones. Even with highly traded assets like S&P500 futures the after-hour prices are based on a much smaller volume. The question is whether the available prices on Globex are indicative. The shared layer perceptron offers the possibility to let the network run freely where no prices are available. The network produces an interpolation of the unknown price dynamics. It is of interest to analyze, whether such a setup improves intraday forecasts.

Relatedly, one could imagine merging long term and short term models. E.g., the 20 step forecast of a daily model could be augmented by the forecast of a monthly model. This could help to spot divergences in the dynamics of the underlying systems: if monthly data suggests a completely different result than daily data care should be taken in the interpretation of the model. Similar ideas could apply when considering a short term intraday model merged with a long term daily model.

The author would also like to examine the effect of exogeneous, non forecasted, time series in model building. One domain of time series in this category includes all dummy variables related to time: e.g., day of the week, day of the month, month, holidays, a priori known economic events, etc. These are *always* known in advance. A second class includes observeables which are better forecasted using other means. This is especially true for weather forecasts which play a considerable role especially in the pricing of food commodities. Fuels are involved, too. One could take these forecasts as given exogeneously, teacher force the appropriate states and neglect the error. These observeables would therefore act as true input variables. If these observables manage to improve forecasts significantly this is attractive because the improvement comes at reduced additional cost. There is no need to calculate and feed back local training error on these states.

Considering financial applications one is obviously interested in improving performance based on some *financial criteria*, generally some kind of *return*. Training error is of secondary interest. As it is still unclear how much time series make up a good model the author suggests to study thoroughly the effect of adding or removing possible explanatory time series. What happens if one only takes equities or only currencies? What happens if one doubles the number of available time series? What happens if one adds redundant time series? All these questions are —

in principle — easy to answer by performing the appropriate experiment.

One important aspect of this book is using shared layer perceptrons as part of a financial decision support system. As already mentionned a decision support systems consists of more than just the mathematical kernel. The other components which deliver a useable tool are just as important. Considering this it is important for the author to put efforts into developing a tool which provides an easy to use interface to the shared layer perceptron. One has to keep in mind that mathematical details are better abstracted from most users.

The present book only considers financial applications. However, multi variate multi step forecasts may of value in other domains, too. As the author has experience in the field of dynamic games the topic *optimal control* comes to mind. When using a feedback controller one often only forecasts the next step of the dynamic system. Many systems are slow to react. In these cases a multi step forecast might offer added value. The forecast prevents overreaction. One may think of systems like cars of airplanes. Especially airplanes will only react comparatively slowly to, e.g., a change in direction. Can shared layer perceptrons improve dynamic game modelling, especially when an analytical solution is not available? As a first example the author could use investigations of the game of two cars, a collision avoidance scenario. This has already successfully been addressed using a three layer perceptron. A comparative study is of interest: can shared layer perceptrons improve the reaction of the evader?

5.5 Management Recommendations

This section intends to give the reader some important aspects to consider if one wants to implement grid enabled predictive modeling with neural networks. It is — still — not a comprehensive framework. But the points give the reader criteria to base a decision on.

- Using the FAUN grid computing client provides significant hardware cost savings if you already have user workstations available. Keep in mind that installing the grid computing client organization-wide might require to change some networking setups.
- If you plan to buy new hardware for computation you don't need expensive massively parallel computers. Simple quad cores or dual quad core servers

are sufficient.

- Communication requirements are low. An existing 100 MBit network can be used without problems. If more than 20 clients are involved in the computation the author recommends using standard 1 GBit networking equipment. In any case you don't need more expensive high bandwidth or low latency equipment because they are not required for coarse grained parallelization.

- Do you have appropriate equipment in place to guarantee that computers are *always* available? Otherwise Murphy's Law will cause hardware to go down when you most need it.

- What do you want to model? Are you sure you need a non linear model? Or is a standard linear model enough? Just test you problem with any statistics software. Do you get satisfactory results? If you do, you can probably stop here.

- Do you have sufficient data available? Neural networks are generally best employed when history abounds. You should have several hundred time steps. But a sparse shared layer perceptron can in principle also perform on much fewer data.

- Select time series wisely. Too much time series will increase training times. Too few time series will not lead to the desired affect of modeling an entire market. As a heuristic you should not only take time series from the asset class you are trying to model. Other assets should also be included. Using approximately 25 time series provides good results.

- Invest great efforts in data quality and data analysis. Neural network learning depends *exclusively* on the data. This means that outliers will be learned or will at least affect forecasts negatively. Compute the usual descriptive statistics but do not forget to also perform a *visual* analysis. This will rapidly uncover anomalies and periods which might be difficult to forecast.

- Do not judge network performance on training error alone. The best performance measure is *always* the one you are interested in. If you are, e.g., interested in timing the lowest entry point within the next 20 days then you have to benchmark the network on this.

- Test a model on different markets and different time periods. Only a model which performs well on a broad range is trustworthy. The shared layer perceptron will generally perform well or very well on most assets and most time spans. Contrast this to benchmark models.
- Do not trust any model blindly. Always check if market circumstances are such that you can realistically assume the model to work properly. Account for market shocks.

Paying attention to the above points will not cause one to beat the market in every case. But it assures that one is aware of possibilities and limitations of FAUN in general and the grid computing version in particular.

5.6 Some Final Words...

Writing this book provided me with a lot of pleasure. It is always rewarding when germs of ideas take form. The author hopes that you had as much pleasure reading this book. The author hopes that he could transfer some of his enthusiasm vis-à-vis neural networks to the reader. If you have any open question, please do not hesitate to contact the author, e.g., at `mettenheim@iwi.uni-hannover.de`. The author is always thankful and open for feedback, further ideas and co-operations.

And now: happy neural networking!

Bibliography

[1] Y. S. Abu-Mostafa. Financial model calibration using consistency hints. *Neural Networks, IEEE Transactions on*, 12(4):791–808, Jul 2001.

[2] O. Adam, J.-L. Zarader, and M. Milgram. Identification and prediction of non-linear models with recurrent neural network. In *IWANN '93: Proceedings of the International Workshop on Artificial Neural Networks*, pages 531–535, London, UK, 1993. Springer-Verlag.

[3] Agence France Trésor. *OATs: Investing directly*, november 2005.

[4] A. Ali, A. Anjum, T. Azim, J. Bunn, A. Mehmood, R. McClatchey, H. Newman, W. Rehman, C. Steenberg, M. Thomas, F. van Lingen, I. Willers, and M. Zafar. Resource management services for a grid analysis environment. In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pages 53–60, 2005.

[5] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schott, E. Seidel, and B. Ullmer. The grid application toolkit: toward generic and easy application programming interfaces for the grid. *Proceedings of the IEEE*, 93(3):534–550, 2005.

[6] A. S. Andreou, E. F. Georgopoulos, and S. D. Likothanassis. Exchange-rates forecasting: A hybrid algorithm based on genetically optimized adaptive neural networks. *Computational Economics*, 20(3):191–210, 2002.

[7] P. C. Andreou, C. Charalambous, and S. H. Martzoukos. Pricing and trading european options by combining artificial neural networks and parametric models with implied parameters. *European Journal of Operational Research*, 185(1415–1433), 2008.

[8] K. K. Ang and C. Quek. Stock trading using rspop: A novel rough set-based neuro-fuzzy approach. *Neural Networks, IEEE Transactions on*, 17(5):1301–1315, Sept. 2006.

[9] J. J. Angel. *Market Mechanics: A Guide to U.S. Stock Markets*. NASDAQ educational foundation, 1.2 edition, 2002.

[10] D. Aronson. *Evidence-Based Technical Analysis*. Wiley, Hoboken, New Jersey, 2007.

[11] R. T. Baillie and T. Bollerslev. The message in daily exchange rates: A conditional-variance tale. *Journal of Business and Economic Statistics*, 7:297–305, 1989.

[12] R. T. Baillie and T. Bollerslev. Intra-day and inter-market volatility in foreign exchange rates. *The Review of Economic Studies*, 58(3):565–586, 1991.

[13] R. T. Baillie and T. Bollerslev. Prediction in dynamic models with time-dependent conditional variances. *Journal of Econometrics*, 52:91–113, 1992.

[14] P. P. Balestrassi, E. Popova, A. P. Paiva, and J. W. M. Lima. Design of experiments on neural network's training for nonlinear time series forecasting. *Neurocomputing*, 72(4-6):1160–1178, 2009.

[15] S. D. Balkin and J. K. Ord. Automatic neural network modeling for univariate time series. *International Journal of Forecasting*, 16:509–515, 2000.

[16] P. Bartels. *Echtzeit-Bewertung von Optionen mit Marktpreisen durch Web-Mining und Neurosimulation*. PhD thesis, Wirtschaftswissenschaftliche Fakultät der Leibniz Universität Hannover, 2007. In German.

[17] P. Bartels and M. H. Breitner. *Warrant Pro 1: Market Price Synthesis with a Software Agent and a Neurosimulator*, volume 300 of *Diskussionspapiere der Wirtschaftswissenschaftlichen Fakultät*. Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2004.

[18] P. Bartels and M. H. Breitner. Real-time market valuations of options based on web mining and neurosimulation. In H. Österle, J. Schelp, and R. Winter, editors, *Proceedings of the European Conference on Information Systems (ECIS) 2007, St. Gallen*, pages 466–479, 2007.

[19] A. Barthel. Effiziente Approximation von Kenngrößen für Warteschlangen mit dem Neurosimulator FAUN 1.0. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2003. In German. Diplomarbeit ≈ Master's thesis.

[20] K. Bartlmae and F. A. Rauscher. Measuring DAX market risk: A neural network volatility mixture approach. In *Proceedings of the Forecasting Financial Markets Conference, London*, 2000.

[21] BBA. *Understanding BBA LIBOR*, 2008.

[22] S. D. Bekiros and D. A. Georgoutsos. *Forecasting stock index returns using a volatility based recurrent neural network*. Department of Accounting and Finance, Athens University of Economics and Business, October 2004. Preliminary version.

[23] S. D. Bekiros and D. A. Georgoutsos. Evaluating direction-of-change forecasting: Neurofuzzy vs. neural networks. *Mathematical and Computer Modelling*, 46:38–46, 2007.

[24] S. Benkner and G. Engelbrecht. A generic qos infrastructure for grid web services. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 141–141, 2006.

[25] K. Bergerson and D. C. Wunsch. A commodity trading model based on a neural network-expert system hybrid. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN), Seattle*, volume 1, pages 289–293, 1991.

[26] T. Berghage and G. Olander. *Predicting the Unpredictable: How Neural Networks Enhance Predictability and Performance in Portfolio Management*. Acacia Publishing, Phoenix Arizona, 2007.

[27] J. Berk and P. DeMarzo. *Corporate Finance*. Pearson Education, 2007.

[28] L. Bertolini. *Quantitative Methods for FX Carry Trading*. Cass Business School, London, December 2008. Talk given during GOR session.

[29] C. Bischof, A. Carle, P. Hovland, P. Khademi, and A. Mauer. Adifor 2.0 users' guide (revision d). Technical Report CRPC-95516-S, Center for Research on Parallel Computation, June 1998.

[30] C. Bischof, A. Carle, P. Khademi, and A. Mauer. The adifor 2.0 system for the automatic differentiation of fortran 77 programs. Technical Report CRPC-TR94491, Center for Research on Parallel Computation, 1998.

[31] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[32] F. Black and M. Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[33] J. Blotner. It's more than just toys and food: leading agile development in an enterprise-class start-up. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 81–91, 2003.

[34] M. Bock and R. Mestel. *A regime-switching relative value arbitrage rule*. Institute for Banking and Finance, University of Graz, 2008. Working paper.

[35] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, pages 1–51, 1995.

[36] N. P. B. Bollen, S. F. Gray, and R. E. Whaley. Regime switching in foreign exchange rates: Evidence from currency option prices. *Journal of Econometrics*, 94:239–276, 2000.

[37] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[38] R. Bookstaber. *A Demon of our own Design: Markets, Hedge Funds, and the Perils of Financial Innovation*. Wiley, Hoboken, New Jersey, 2007.

[39] J. Borkowski, D. Kopanski, and M. Tudruj. Application control on grid using predicates defined on global states of co-operating parallel programs. In *Parallel, Distributed, and Network-Based Processing, 2006. PDP 2006. 14th Euromicro International Conference on*, pages 248–255, 2006.

[40] A. Brabazon and M. O'Neill. Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. *Computational Management Science*, 1(3-4):311–327, 2004.

[41] M. H. Breitner. Heuristic option pricing with neural networks and the neurocomputer SYNAPSE 3. *Optimization*, 47(3):319–333, 2000.

[42] M. H. Breitner. Robust optimal on-board reentry guidance of a space shuttle: Dynamic game approach and guidance synthesis with neural networks. *Journal of Optimization Theory and Applications*, 107:483–505, 2000.

[43] M. H. Breitner. *Nichtlineare, multivariate Approximation mit Perzeptrons und anderen Funktionen auf verschiedenen Hochleistungsrechnern*, volume 263 of *Dissertationen zur Künstlichen Intelligenz*. Akademische Verlagsgesellschaft, Berlin, 2003. In German.

[44] M. H. Breitner. Usage of artificial neural networks for the numerical solution of dynamic games (a tutorial). In *Proceedings of the Eleventh International Symposium on Dynamic Games and Applications (Tucson)*, 2004.

[45] M. H. Breitner. Customer tailored derivatives: Simulation, design and optimization with the WARRANT-PRO-2 software. In M. H. Breitner, G. Denk, and P. Rentrop, editors, *From Nano to Space: Applied Mathematics inspired by Roland Bulirsch*, pages 211–228. Springer, Berlin Heidelberg, 2008.

[46] M. H. Breitner, F. Köller, S. König, and R. Kossow. Efficient synthesis of strategies with the advanced neurosimulator FAUN 1.0. In M. H. Breitner, editor, *Proceedings of the Fourth International ISDG Workshop, Goslar, 2003*.

[47] M. H. Breitner, F. Köller, S. König, and H.-J. von Mettenheim. Intelligent decision support systems and neurosimulators: A promising alliance for financial services providers. In H. Österle, J. Schelp, and R. Winter, editors, *Proceedings of the European Conference on Information Systems (ECIS) 2007, St. Gallen*, pages 478–489, 2007.

[48] R. G. Brown. Maximizing Beowulf performance. In USENIX, editor, *Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, October 10–14, 2000, Atlanta, Georgia, USA*. USENIX, 2000.

[49] D. Butterworth and P. Holmes. Mispricing in stock index futures contracts: evidence for the FTSE 100 and FTSE mid 250 contracts. *Applied Economics Letters*, 7:795–801, 2000.

[50] D. Butterworth and P. Holmes. Inter-market spread trading: evidence from UK index futures markets. *Applied Financial Economics*, 12:783–790, 2002.

[51] R. Buyya, D. Abramson, and J. Giddy. An economy driven resource management architecture for global computational power grids, may 2000.

[52] L. Canina and S. Figlewski. The informational content of implied volatility. *The Review of Financial Studies*, 6(3):659–681, 1993.

[53] J. Cao, S. Jarvis, S. Saini, and G. Nudd. Gridflow: workflow management for grid computing. In *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, pages 198–205, 2003.

[54] P. Cashin, C. J. McDermott, and A. Scott. *Booms and slumps in world commodity prices*. Reserve Bank of New Zealand, December 1999. Discussion paper series, G99/8.

[55] C. E. Catlett. Teragrid: A foundation for US cyberinfrastructure. In *NPC*, page 1, 2005.

[56] E. P. Chan. *Quantitative Trading*. Wiley, Hoboken, New Jersey, 2009.

[57] N. H. Chan and C. Genovese. A comparison of linear and nonlinear statistical techniques in performance attribution. *Neural Networks, IEEE Transactions on*, 12(4):922–928, Jul 2001.

[58] P. C. Chang, C. H. Liu, J. L. Lin, C. Y. Fan, and C. S. P. Ng. A neural network with a case based dynamic window for stock trading prediction. *Expert Systems with Applications*, 36:6889–6898, 2009.

[59] N. Chapados and Y. Bengio. Cost functions and model combination for VaR-based asset allocation using neural networks. *Neural Networks, IEEE Transactions on*, 12(4):890–906, Jul 2001.

[60] S. J. Chapman. *Fortran 95/2003 for Scientists and Engineers*. McGraw-Hill, New York, 3rd edition, 2008.

[61] A. S. Chen and M. T. Leung. Regression neural network for error correction in foreign exchange forecasting and trading. *Computers and Operations Research*, 31(7):1049–1068, 2004.

[62] A. S. Chen, M. T. Leung, and H. Daouk. Application of neural networks to an emerging financial market: forecasting and trading the Taiwan stock index. *Computers and Operations Research*, 30(6):901–923, 2003.

[63] G. Chen, P. Thulasiraman, and R. Thulasiram. Distributed quasi-monte carlo algorithm for option pricing on hnows using mpc. *Simulation Symposium, 2006. 39th Annual*, April 2006.

[64] B. Cheng and D. M. Titterington. Neural networks: A review from a statistical perspective. *Statistical Science*, 9(1):2–30, 1994.

[65] D. P. Chiras and S. Manaster. The information content of option prices and a test of market efficiency. *Journal of Financial Economics*, 6:213–234, 1978.

[66] T. Choudhry, F. McGroarty, K. Peng, and S. Wang. Artificial neural network and high frequency exchange rate prediction. In C. Dunis, M. Dempster, and V. Terraza, editors, *Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, Luxembourg, 27–29 May 2009*, 2009.

[67] L. Chunlin and L. Layuan. An agent-based approach for grid computing. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 608–611, 2003.

[68] A. Cifter and A. Ozun. A signal processing model for time series analysis: The effects of international f/x markets on domestic currencies using wavelet networks. *International Review of Electrical Engineering (IREE)*, 3(3), 2008.

[69] R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5:559–583, 1989.

[70] CME Group. *Nikkei 225 Futures and Options*, 2007.

[71] CME Group. *Eurodollar Futures*, 2008.

[72] J. Cohen. Statistical power analysis. *Current Direction in Psychological Science*, 1(3):98–101, June 1992.

[73] E. Constantinou, R. Georgiades, A. Kazandjian, and G. P. Kouretas. *Regime Switching and Artificial Neural Network Forecasting of the Cyprus Stock Exchange Daily Returns*. Department of Accounting and Finance, The Philips College, Nicosia, Cyprus, 2005.

[74] C. Cooley. Daily iterations: approaching code freeze and half the team is not agile. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 162–164, 2003.

[75] C. M. Corcoran. *Long/Short Market Dynamics*. Wiley, Southern Gate, Chichester, 2007.

[76] J. A. Coutts and K.-C. Cheung. Trading rules and stock returns: some preliminary short run evidence from the hang seng 1985-1997. *Applied Financial Economics*, 10:579–586, 2000.

[77] S. F. Crone. *Forecasting with Neural Networks: a hands-on tutorial*. Lancaster Center for Forecasting, December 2008. Talk given during GOR session.

[78] S. F. Crone, S. Lessmann, and S. Pietsch. Forecasting with computational intelligence - an evaluation of support vector regression and artificial neural networks for time series prediction. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 3159–3166, 0-0 2006.

[79] S. F. Crone, S. Lessmann, and S. Pietsch. Parameter Sensitivity of Support Vector Regression and Neural Networks for Forecasting. In *International Conference on Data Mining, Las Vegas (USA)*, 2006.

[80] M. M. Dacorogna, R. Gencay, U. Müller, R. B. Olsen, and O. V. Pictet. *An Introduction to High-Frequency Finance*. Academic Press, San Diego, California, 2001.

[81] F. Darema. Grid computing and beyond: the context of dynamic data driven applications systems. *Proceedings of the IEEE*, 93(3):692–697, 2005.

[82] A. F. Darrat and M. Zhong. On testing the random-walk hypothesis: A model-comparison approach. *The Financial Review*, 35:105–124, 2000.

[83] M. De Leenheer, P. Thysebaert, B. Volckaert, F. De Turck, B. Dhoedt, P. Demeester, D. Simeonidou, R. Nejabati, G. Zervas, D. Klonidis, and M. O'Mahony. A view on enabling-consumer oriented grids through optical burst switching. *IEEE Communications Magazine*, 44(3):124–131, 2006.

[84] T. DeMarco and B. Boehm. The agile methods fray. *Computer*, 35(6):90–92, 2002.

[85] M. A. H. Dempster, T. W. Payne, Y. Romahi, and G. W. P. Thompson. Computational learning techniques for intraday fx trading using popular technical indicators. *Neural Networks, IEEE Transactions on*, 12(4):744–754, Jul 2001.

[86] J. E. Dennis and J. Moré. Quasi-newton methods, motivation and theory. Technical Report 74-217, Cornell University, November 1974.

[87] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, chapter 6, Globally Convergent Modifications of Newton's Method. Society for Industrial Mathematics, Philadelphia, 1996 (reprint).

[88] Deutsche Börse. *Guide to the Equity Indices of Deutsche Börse*, 6.11 edition, April 2009.

[89] Deutsche Börse. *Xetra Auction Plan*, January 2009.

[90] V. Dhar and D. Chou. A comparison of nonlinear methods for predicting earnings surprises and returns. *Neural Networks, IEEE Transactions on*, 12(4):907–921, Jul 2001.

[91] A. Diekmann and S. Gutjahr. Prediction of the Euro-Dollar future using neural networks: a case study for financial time series prediction. In *Intelligent data engineering and learning: perspectives on financial engineering and data mining: 1st International Symposium, IDEAL'98: Hong Kong, October 14-16, 1998*, pages 13–18. Eureka Publications, 1998.

[92] P. Disorntetiwat and C. Dagli. Simple ensemble-averaging model based on generalized regression neural network in financial forecasting problems. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 477–480, 2000.

[93] R. G. Donaldson and M. Kamstra. Forecast combining with neural networks. *Journal of Forecasting*, 15:49–61, 1996.

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

[95] R. G. Donaldson and M. Kamstra. Neural network forecast combining with interaction effects. *Journal of the Franklin Institute*, 336:2237–236, 1999.

[96] R. Dubil. *An Arbitrage guide to financial markets*. John Wiley, Southern Gate, Chichester, 2004. Reprinted October 2006.

[97] C. Dunis, A. Harris, S. Leong, and P. Nacaskul. Optimising intraday trading models with genetic algorithms. *Neural Network World*, 9:193–224, 1999.

[98] C. Dunis, J. Laws, and P. Naim, editors. *Applied quantitative methods for trading and investment*. Wiley, Southern Gate, Chichester, 2003.

[99] C. Dunis, J. Laws, and G. Sermpinis. *Higher Order and Recurrent Neural Architectures for Trading the EUR/USD Exchange Rate*. Liverpool Business School and CIBEF, February 2008. CIBEF working papers.

[100] C. L. Dunis and Y. X. Chen. Alternative volatility models for risk management and trading: An application to the EUR/USD and USD/JPY rates. *Derivatives Use, Trading and Regulation*, 11(2):126–156, 2005.

[101] C. L. Dunis and X. Huang. Forecasting and trading currency volatility: An application of recurrent neural regression and model combination. *Journal of Forecasting*, 21(5):317–354, 2002.

[102] C. L. Dunis and J. Jalilov. Neural network regression and alternative forecasting techniques for predicting financial variables. *Neural Network World*, 12(2):113–139, 2002.

[103] C. L. Dunis and V. Karalis. Weather derivatives pricing and filling analysis for missing temperature data. *Derivatives Use, Trading and Regulation*, 9(1):61–83, 2003.

[104] C. L. Dunis, J. Laws, and B. Evans. Modelling and trading the gasoline crack spread: A non-linear story. *Derivatives Use, Trading and Regulation*, 12(1/2):126–147, 2006.

[105] C. L. Dunis, J. Laws, and B. Evans. Modelling and trading the soybean-oil crush spread with recurrent and higher order networks: A comparative analysis. *Neural network World*, 13(3/6):193–213, 2006.

[106] C. L. Dunis, J. Laws, and B. Evans. Trading futures spreads: an application of correlation and threshold filters. *Applied Financial Economics*, 16(12):903–914, 2006.

[107] C. L. Dunis, J. Laws, and B. Evans. *Trading and Filtering Futures Spread Portfolios: Further Applications of Threshold and Correlation Filters*. Liverpool Business School and CIBEF, December 2008. CIBEF working papers.

[108] C. L. Dunis, J. Laws, and B. Evans. Trading futures spread portfolios: applications of higher order and recurrent networks. *The European Journal of Finance*, 14(6):503–521, 2008.

[109] C. L. Dunis, J. Laws, and A. Karathanasopoulos. Modeling the greek stock market with hybrid-arma network models. In C. Dunis, M. Dempster, and V. Terraza, editors, *Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, Luxembourg, 27–29 May 2009*, 2009.

[110] C. L. Dunis, J. Laws, and G. Sermpinis. *Modelling and Trading the EUR/USD Exchange Rate at the ECB Fixing*. Liverpool Business School and CIBEF, June 2008. CIBEF working papers.

[111] C. L. Dunis, J. Laws, and G. Sermpinis. *Modelling Commodity Value at Risk with Higher Order Neural Networks*. Liverpool Business School and CIBEF, September 2008. CIBEF working papers.

[112] C. L. Dunis, J. Laws, and G. Sermpinis. *The Robustness of Neural Networks for Modelling and Trading the EUR/USD Exchange Rate at the ECB Fixing*. Liverpool Business School and CIBEF, November 2008. CIBEF working papers.

[113] C. L. Dunis, J. Laws, and G. Sermpinis. *Modelling and Trading the Realised Volatility of the FTSE100 Futures with Higher Order Neural Networks*. Liverpool Business School and CIBEF, March 2009. CIBEF working papers.

[114] C. L. Dunis and J. Miao. Advanced frequency and time domain filters for currency portfolio management. *Journal of Asset Management*, 7(1):22–33, 2006.

[115] C. L. Dunis and V. Morrison. The economic value of advanced time series methods for modelling and trading 10-year government bonds. *European Journal of Finance*, 13(4):333–352, 2007.

[116] C. L. Dunis and A. Nathani. Quantitative trading of gold and silver using nonlinear models. *Neural Network World*, 17(2):93–110, 2007.

[117] C. L. Dunis and J. A. Triantafyllidis. Alternative forecasting techniques for predicting company insolvencies: The UK example (1980-2001). *Neural Network World*, 13(4):326–360, 2003.

[118] C. L. Dunis and M. Williams. Modelling and trading the EUR/USD exchange rate: Do neural network models perform better? *Derivatives Use, Trading and Regulation*, 8(3):211–239, 2002.

[119] R. D. Edwards and J. Magee. *Technical Analysis of Stock Trends*. BN Publishing, 2008.

[120] G. Eisenhauer, K. Schwan, and F. Bustamante. Publish-subscribe for high-performance computing. *IEEE Internet Computing*, 10(1):40–47, 2006.

[121] G. Elliott, T. J. Rothenberg, and J. H. Stock. Efficient tests for an autoregressive unit root. *Econometrica*, 64(4):813–836, July 1996.

[122] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[123] H. Englisch and S. Mayhew. Artificial market making with neural nets: an application to options. *Computational Intelligence for Financial Engineering, 1995.,Proceedings of the IEEE/IAFE 1995*, pages 156–159, Apr 1995.

[124] Eurex Frankfurt AG. *Interest Rate Derivatives: Fixed Income Futures*, October 2007.

[125] T. Fahringer, J. Qin, and S. Hainzer. Specification of grid workflow applications with AGWL: an abstract grid workflow language. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 676–685, 2005.

[126] E. F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105, 1965.

[127] R. Ferland and S. Lalaneette. *Dynamics of Realized Volatility and Correlations: An Empirical Study Using Interest Rate Spread Options*. Department of Mathematics, University of Quebec in Montreal, August 2001. Working paper.

[128] F. Fernandez-Rodrıguez, C. Gonzalez-Martel, and S. Sosvilla-Rivero. On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market. *Economics Letters*, 69(1):89–94, 2000.

[129] J. Fleming. The quality of market volatility forecasts implied by s&p100 index option prices. *Journal of Empirical Finance*, 5:317–345, 1998.

[130] I. Foster. What is the grid? a three point checklist, July 2002.

[131] I. Foster. Service-oriented science. *Science*, 308(5723):814–817, May 2005.

[132] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, June 28 2002.

[133] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid. *International Journal of High Performance Computer Applications*, 15(3):200–222, October 2001.

[134] I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. In *NPC*, pages 2–13, 2005.

[135] G. Fox. Grids of grids of simple services. *Computing in Science & Engineering [see also IEEE Computational Science and Engineering]*, 06(4):84–87, 2004.

[136] P. H. Franses and K. van Griensven. Forecasting exchange rates using neural networks for technical trading rules. *Studies in Nonlinear Dynamics & Econometrics*, 2(4):109–114, 1997.

[137] P. H. Franses and P. van Homelen. On forecasting exchange rates using neural networks. *Applied Financial Economics*, 8:589–596, 1998.

[138] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996.

[139] M. Friedman. The euro-dollar market: Some first principles. *Journal of the Federal Reserve Bank of St. Louis*, July 1971.

[140] FTSE. *FTSE MIB Index*, March 2009.

[141] J. Fulcher, M. Zhang, and S. Xu. Application of higher-order neural networks to financial time-series prediction. In J. Kamruzzaman, R. K. Begg, and R. A. Sarker, editors, *Artificial Neural Networks in Finance and Manufacturing*, volume 5, pages 80–108. Idea Group Publishing, 2006.

[142] G. Fusai and A. Roncoroni. *Implementing Models in Quantitative Finance: Methods and Cases*. Springer, Berlin, Heidelberg, New York, 2008.

[143] J. E. Gagnon and A. P. Chaboud. What can the data tell us about carry trades in japanese yen. *International Finance Discussion Papers*, 889, July 2007. Published by the Board of Governors of the Federal Reserve System.

[144] R. Garcia and R. Gencay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, 94:93–115, 2000.

[145] M. B. Garman and S. W. Kohlhagen. Foreign currency option values. *Journal of International Money and Finance*, 2:231–237, 1983.

[146] V. V. Gavrishchaka and S. Banerjee. Support vector machine as an efficient framework for stock market volatility forecasting. *Computational Management Sciences*, 3(2):147–160, 2006.

[147] R. Gencay. Non-linear prediction of security returns with moving average rules. *Journal of Forecasting*, 15:165–174, 1996.

[148] R. Gencay. Linear, non-linear and essential foreign exchange rate prediction with simple technical trading rules. *Journal of International Economics*, 47:91–107, 1999.

[149] R. Gencay and R. Gibson. Model risk for european-style stock index options. *Neural Networks, IEEE Transactions on*, 18(1):193–202, Jan. 2007.

[150] R. Gencay and M. Qi. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *Neural Networks, IEEE Transactions on*, 12(4):726–734, Jul 2001.

[151] R. Gencay, F. Selcuk, and B. Whitcher. *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*. Academic Press, San Diego, California, 2002.

[152] J. W. Gentry, S. Jun, S. Chun, H. Kang, and G. Ko. Korea – two countries, sharp contrasts, but a common heritage. In A. Pectoich and C. Shultz, editors, *Handbook of markets and economies: East Asia, Southeast Asia, Australia, New Zealand*. M. E. Sharpe, 2006.

[153] M. R. Gibbons and P. Hess. Day of the week effects and asset returns. *The Journal of Business*, 54(4):579–596, October 1981.

[154] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for LSSOL. Technical Report SOL 86-1, Stanford University, 1986.

[155] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Some theoretical properties of an augmented lagrangian merit function. Technical Report SOL 86-6R, Stanford University, 1986.

[156] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for NPSOL 5.0: A Fortran package for nonlinear programming. Technical Report SOL 86-1, University of California, San Diego, 1998.

[157] M. Giordano and M. M. Furnari. An incremental compilation approach for openmp applications. In *NPC*, pages 249–252, 2005.

[158] A. Gokhale and B. Natarajan. GriT: a CORBA-based grid middleware architecture. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, 2003.

[159] S. C. Gold and P. Lebowitz. Computerized stock screening rules for portfolio selection. *Financial Services Review*, 8:61–70, 1999.

[160] N. I. M. Gould, D. Orban, and P. L. Toint. Numerical methods for large-scale nonlinear optimization. Technical Report RAL-TR-2004-032, Council for the Central Laboratory of the Research Councils, Oxfordshire, United Kingdom, November 2004.

[161] B. Graham. *The Intelligent Investor*. Collins Business Essentials, 4th edition, 2006. Revised edition of the original book from 1973.

[162] C. W. J. Granger. Forecasting stock market prices: Lessons for forecasters. *International Journal of Forecasting*, 8:3–13, 1992.

[163] J. E. Gray. *Best of Ruby Quiz*. 2006.

[164] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics Philadelphia, 2000.

[165] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. Scientific and Engineering Computation Series. MIT Press, Cambridge, Massachusetts, 2nd edition, 1999.

[166] R. Grothmann. *Multi-Agent Market Modeling Based on Neural Networks*. PhD thesis, Faculty of Economics, University of Bremen, Germany, 2002.

[167] S.-F. Guo, W. Zhang, D. Ma, and W.-L. Zhang. Grid mobile service: using mobile software agents in grid mobile service. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 1, pages 178–182, 2004.

[168] T. Gyires. A resource allocation protocol for providing quality of service in grid computing. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 44–44, 2006.

[169] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, November 1994.

[170] S. A. Hamid. *Primer on Using Neural Networks for Forecasting Market Variables*. School of Business, Southern New Hampshire University, Manchester, 2004. Working Paper No. 2004-03.

[171] J. D. Hamilton and G. Perez-Quiros. What do the leading indicators lead? *The Journal of Business*, 69(1):27–49, January 1996.

[172] J. V. Hansen and R. D. Nelson. Neural networks and traditional time series methods: a synergistic combination in state economic forecasts. *Neural Networks, IEEE Transactions on*, 8(4):863–873, Jul 1997.

[173] A. Harvell. *FTSE All-Share Index Series Constitutents, Weightings and Performance*. FTSE, September 2008.

[174] L. Hascoët and V. Pascual. Tapenade 2.1 user's guide. Technical Report 0300, Institut National de Recherche en Informatique et en Automatique, France, September 2004.

[175] S. Hashem. Algorithms for optimal linear combinations of neural networks. In *Neural Networks,1997., International Conference on*, volume 1, pages 242–247 vol.1, Jun 1997.

[176] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, New Jersey, 3rd edition, 2009.

[177] S. Hochreiter. Recurrent neural net learning and vanishing gradient. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[178] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[179] P. Hodgetts. Refactoring the development process: experiences with the incremental adoption of agile practices. In *Agile Development Conference, 2004*, pages 106–113, 2004.

[180] M. Hohman. Estimating in actual time [extreme programming]. In *Agile Conference, 2005. Proceedings*, pages 132–138, 2005.

[181] L. Holmström and P. Koistinen. Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38, 1992.

[182] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[183] B. Hu and K.-W. Tsui. Distributed evolutionary monte carlo with applications to bayesian analysis. Technical Report 1112, Department of Statistics, University of Wisconsin-Madison, Madison, November 2005.

[184] M. Y. Hu, G. Zhang, C. X. Jiang, and B. E. Patuwo. A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting. *Decision Sciences*, 30(1):197–216, 1999.

[185] W. Huang, K. K. Lai, Y. Nakamori, and S. Wang. Forecasting foreign exchange rates with artificial neural networks: A review. *International Journal of Information Technology and Decision Making*, 3(1):145–165, 2004.

[186] W. Huang, K. K. Lai, and S. Wang. Application of neural networks for foreign exchange rates forecasting with noise reduction. *Lecture Notes in Computer Science*, 4488:455–461, 2007.

[187] W. Huang, S. Wang, H. Zhang, and R. Xiao. Selection of the appropriate lag structure of foreign exchange rates forecasting based on autocorrelation coefficient. *Lecture Notes in Computer Science*, 3979:512–517, 2006.

[188] J. C. Hull. *Options, Futures and Other Derivatives*. Prentice Hall, 6th edition, 2006.

[189] L. Ingber and R. P. Mondescu. Optimization of trading physics models of markets. *Neural Networks, IEEE Transactions on*, 12(4):776–790, Jul 2001.

[190] D. Ivanova, K. Lahiri, and F. Seitz. Interest rate spreads as predictors of german inflation and business cycles. *International Journal of Forecasting*, 16:39–58, 2000.

[191] P. Iyer, A. Nagargadde, S. Gopalan, and V. Sridhar. SOA for hybrid p2p based self organising grids. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 140–140, 2006.

[192] A. Jackson, S. L. Tsang, A. Gray, C. Driver, and S. Clarke. Behind the rules: XP experiences. In *Agile Development Conference, 2004*, pages 87–94, 2004.

[193] C. M. Jarque and A. K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6:255–259, 1980.

[194] T. Jasic and D. Wood. The profitability of daily stock market indices trades based on neural network predictions: case study for the S&P 500, the DAX, the TOPIX and the FTSE in the period 1965-1999. *Applied Financial Economics*, 14:285–297, 2004.

[195] P. Jorion. Predicting volatility in the foreign exchange market. *The Journal of Finance*, 50(2), 507–528 1995.

[196] A. Kanas. Non-linear forecasts of stock returns. *Journal of Forecasting*, 22:299–315, 2003.

[197] A. Kanas and A. Yannopoulos. Comparing linear and nonlinear forecasts for stock returns. *International Review of Economics and Finance*, 10:383–398, 2001.

[198] O. Karaali, W. Edelberg, and J. Higgins. Modelling volatility derivatives using neural networks. *Computational Intelligence for Financial Engineering (CIFEr), 1997, Proceedings of the IEEE/IAFE 1997*, pages 280–286, Mar 1997.

[199] J. O. Katz and D. L. McCormick. *The Encyclopedia of Trading Strategies*. McGraw-Hill, New York, 2000.

[200] P. J. Kaufman. *New Trading Systems and Methods*. John Wiley, Hoboken, New Jersey, 4th edition, 2005.

[201] M. Kehlenbeck. Weiterentwicklung des Neurosimulators FAUN durch Softwareengineering und Einsatz optimierter BLAS-Bibliotheken. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2004. In German. Diplomarbeit ≈ Master's thesis.

[202] K. keung Hung, Y. ming Cheung, and L. Xu. An extended asld trading system to enhance portfolio management. *Neural Networks, IEEE Transactions on*, 14(2):413–425, Mar 2003.

[203] A. Knowles, A. Hussain, W. E. Deredy, P. G. J. Lisboa, and C. Dunis. *Higher-Order Neural Networks with Bayesian Confidence Measure for Prediction of EUR/USD Exchange Rate*. Liverpool Business School and CIBEF, 2005. CIBEF working papers.

[204] F. Köller. *Optimale Echtzeit-Personaleinsatzplanung für Inbound Call Center durch Approximation von Warteschlangenkennzahlen mit Künstlichen Neuronalen Netzen*. PhD thesis, Wirtschaftswissenschaftliche Fakultät der Leibniz Universität Hannover, 2007. In German.

[205] F. Köller and M. H. Breitner. Optimierung von Warteschlangensystemen durch Approximation mit Neuronalen Netzen. In H.-D. Haasis, H. Kopfer, and J. Schönberger, editors, *Operations Research Proceedings 2005: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Bremen, September 7–9, 2005*. Springer Verlag, Heidelberg, 2006.

[206] V. V. Kondratenko and Y. A. Kuperin. *Using Recurrent Neural Networks To Forecasting of Forex*. Department of Physis, St. Petersburg State University, 2003. Working Paper.

[207] S. König. Entwicklung eines PHP-Web-Frontends zur Neurosimulation auf einem Compute Server. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2004. In German. Diplomarbeit ≈ Master's thesis.

[208] S. König, F. Köller, and M. H. Breitner. *FAUN 1.1 User Manual*, volume 16 of *IWI Discussion Paper Series*. Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, August 2005.

[209] Korea Exchange. *KRX Fact Book 2008*, May 2009.

[210] C.-M. Kuan and T. Liu. Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics*, 10(4):347–364, 1995.

[211] Y. Kuranuki and K. Hiranabe. Antipractices: Antipatterns for XP practices. In *Agile Development Conference, 2004*, pages 83–86, 2004.

[212] A. Kuznetsov. *The Complete Guide to Capital Markets for Quantitative Professionals*. McGraw-Hill, 2006.

[213] Y.-K. Kwon and B.-R. Moon. A hybrid neurogenetic approach for stock forecasting. *Neural Networks, IEEE Transactions on*, 18(3):851–864, May 2007.

[214] P. Lajbcygier. Improving option pricing with the product constrained hybrid neural network. *Neural Networks, IEEE Transactions on*, 15(2):465–476, March 2004.

[215] H. A. Latane and R. J. Rendleman. Standard deviations of stock price ratios implied in option prices. *The Journal of Finance*, 31(2):369–381, 1976.

[216] B. LeBaron and A. Weigend. A bootstrap evaluation of the effect of data splitting on financial time series. *Neural Networks, IEEE Transactions on*, 9(1):213–220, Jan 1998.

[217] V. C. S. Lee and H. T. Wong. A multivariate neuro-fuzzy system for foreign currency risk management decision making. *Neurocomputing*, 70:942–951, 2007.

[218] P. Lequeux, editor. *Financial Markets Tick by Tick: Insights in Financial Markets Microstructure*. John Wiley, Chichester, England, 1999.

[219] N. Lesna, V. Repka, T. Shatovska, and A. Koryak. Decision support banking system based on neural network technologies. In *Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004. Proceedings of the International Conference*, pages 366–367, Feb. 2004.

[220] H. Li and R. Kozma. A dynamic neural network method for time series prediction using the kiii model. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 1, 2003.

[221] Y. Li, C. Lin, Q. Li, and Z. Shan. Performance modeling and analysis for resource scheduling in data grids. In *NPC*, pages 32–39, 2005.

[222] Y. Li, F. Rao, Y. Chen, D. Liu, and T. Li. Services ecosystem: towards a resilient infrastructure for on demand services provisioning in grid. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 394–401, 2004.

[223] W. T. Lin and Y. H. Chen. Forecasting foreign exchange rates with an intrinsically nonlinear dynamic speed of adjustment model. *Applied Economics*, 30:295–312, 1998.

[224] A. Lindemann, C. L. Dunis, and P. Lisboa. Probability distributions, trading strategies and leverage: an application of Gaussian mixture models. *Journal of Forecasting*, 23(8):559–585, 2004.

[225] A. Lindemann, C. L. Dunis, and P. Lisboa. Level estimation, classification and probability distribution architectures for trading the EUR/USD exchange rate. *Neural Computing & Applications*, 14(3):256–271, 2005.

[226] A. Lindemann, C. L. Dunis, and P. Lisboa. Probability distribution architectures for trading silver. *Neural Network World*, 15(5):437–470, 2005.

[227] A. Lindemann, C. L. Dunis, and P. Lisboa. Probability distributions and leveraged trading strategies: an application of Gaussian mixture models to the Morgan Stanley Technology Index Tracking Fund. *Quantitative Finance*, 5(5):459–474, 2005.

[228] T. Little, F. Greene, T. Phillips, R. Pilger, and R. Poldervaart. Adaptive agility. In *Agile Development Conference, 2004*, pages 63–70, 2004.

[229] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–313, 1988.

[230] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, August 1978.

[231] London Bullion Market Association. *A Guide to the London Precious Metals Markets*, August 2008.

[232] P. Louridas. JUnit: unit testing and coiling in tandem. *IEEE Software*, 22(4):12–15, 2005.

[233] R. Lowenstein. *When Genius Failed*. Fourth Estate, London, 2002.

[234] G. Luck. Subclassing XP: breaking its rules the right way. In *Agile Development Conference, 2004*, pages 114–119, 2004.

[235] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, Sao Paulo, Delhi, 2003. Seventh printing 2008.

[236] S. Mahadevan. *Making Use of RUBY*. 2002.

[237] B. Mandelbrot and R. L. Hudson. *The (Mis)Behavior of Markets: A Fractal View of Risk, Ruin and Reward*. Basic Books, Cambridge, 2004.

[238] M. Marena, D. Marazzina, and G. Fusai. Option pricing, maturity randomization and grid computing. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[239] H. Marzi and M. Turnbull. Use of neural networks in forecasting financial market. *Granular Computing, 2007. GRC 2007. IEEE International Conference on*, pages 516–516, Nov. 2007.

[240] S. Matsuoka, S. Shinjo, M. Aoyagi, S. Sekiguchi, H. Usami, and K. Miura. Japanese computational grid research project: NAREGI. *Proceedings of the IEEE*, 93(3):522–533, 2005.

[241] J. M. Matías and J. C. Reboredo. Nonlinear predictability of intraday stock returns. In C. Dunis, M. Dempster, and V. Terraza, editors, *Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, Luxembourg, 27–29 May 2009*, 2009.

[242] P. J. McCann and B. L. Kalman. *A Neural Network Model for the Gold Market*. Department of Computer Science, Washington University, St. Louis, Missouri, 1994.

[243] S. A. McCrary. *Hedge fund course*. Wiley, Hoboken, New Jersey, 2005.

[244] P. D. McNelis. *Neural Networks in Finance: Gaining Predictive Edge in the Market*. Academic Press, Burlington, 2005.

[245] M. C. Medeiros, A. Veiga, and C. E. Pedreira. Modeling exchange rates: smooth transitions, neural networks, and linear models. *Neural Networks, IEEE Transactions on*, 12(4):755–764, Jul 2001.

[246] P. Mehmert. Mittelfristige Zinsprognose basierend auf technischen Ansätzen mit parallel trainierten Perzeptrons durch FAUN 0.2-PVM. Master's thesis, Institut für Mathematik, Technische Universität Clausthal, July 2000. In German. Diplomarbeit ≈ Master's thesis.

[247] G. Meissner and N. Kawano. Capturing the volatility smile of options on high-tech stocks — a combined garch-neural network approach. *Journal of Economics and Finance*, 25(3):276–291, 2001. Also in Derivatives Use, Trading and Regulation.

[248] M. Metcalf, J. Reid, and M. Cohen. *Fortran 95/2003 explained*. Oxford University Press, 2008. Reprint with corrections.

[249] A. Meyer. Numerische Lösung einer Klasse dynamischer Zwei-Personen-Nullsummen-Spiele durch Diskretisierung und Synthese mit einem Neurosimulator. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2005. In German. Diplomarbeit ≈ Master's thesis.

[250] T. Miazhynskaia, S. Frühwirth-Schnatter, and G. Dorffner. Neural network models for conditional distribution under bayesian analysis. *Neural Computation*, 20(2):504–522, 2008.

[251] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–545, 1993.

[252] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the eleventh international joint conference on artificial Intelligence*, volume 123, 1989.

[253] J. Moody and M. Saffell. Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on*, 12(4):875–889, Jul 2001.

[254] J. J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance, 1999.

[255] V. Nagarajan, Y. Wu, M. Liu, and Q.-G. Wang. Forecast studies for financial markets using technical analysis. In *Control and Automation, 2005. ICCA '05. International Conference on*, volume 1, pages 259–264 Vol. 1, June 2005.

[256] R. Neuneier and H. G. Zimmermann. How to train neural networks. *Lecture notes in computer science*, pages 373–423, 1998.

[257] J. Nocedal and S. J. Wright. *Numerical Optimization*, chapter 7. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd edition, 2006.

[258] K. Nygren. Stock prediction – a neural network approach. Master's thesis, Royal Institute of Technology, KTH, March 2004.

[259] NYSE Euronext. *Rules for the CAC 40 index*, January 2009.

[260] D. Olson and C. Mossman. Neural network forecasts of canadian stock returns using accounting ratios. *International Journal of Forecasting*, 19(3), 2003.

[261] P. S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, San francisco, California, 1997.

[262] N. G. Pavlidis, D. K. Tasoulis, and M. N. Vrahatis. Financial forecasting through unsupervised clustering and evolutionary trained neural networks. In *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*, pages 2314–2321, 2003.

[263] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, September 1995.

[264] L. Peng, L. K. Ng, and S. See. Yellowriver: A flexible high performance cluster computing service for grid. In *High-Performance Computing in Asia-Pacific Region, 2005. Proceedings. Eighth International Conference on*, pages 553–558, 2005.

[265] L. Peng, S. See, Y. Jiang, J. Song, A. Stoelwinder, and H. K. Neo. Performance evaluation in computational grid environments. In *High Performance Computing and Grid in Asia Pacific Region, 2004. Proceedings. Seventh International Conference on*, pages 54–62, 2004.

[266] J. Penn. *The role of the Baltic Exchange*. The Baltic Exchange, September 2005.

[267] P. C. B. Phillips and P. Perron. Testing for a unit root in time series regression. *Biometrika*, 75(2):335–346, June 1988.

[268] M. Pires and T. Marwala. American option pricing using bayesian multi-layer perceptrons and bayesian support vector machines. *Computational Cybernetics, 2005. ICCC 2005. IEEE 3rd International Conference on*, pages 219–224, April 2005.

[269] M. M. Pires and T. Marwala. Option pricing using bayesian neural networks. In *Proceedings of the Annual Symposium of the Pattern Recognition Association of South Africa, Cape Town*, pages 161–166, 2004.

[270] A. Pole. *Statistical Arbitrage: Algorithmic Trading Insights and Techniques*. John Wiley, Hoboken, New Jersey, 2007.

[271] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. With contributions by J. R. Koza.

[272] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, Hoboken, New Jersey, 2007.

[273] A.-P. Refenes and W. Holt. Forecasting volatility with neural regression: A contribution to model adequacy. *Neural Networks, IEEE Transactions on*, 12(4):850–864, Jul 2001.

[274] Reuters and Jefferies. *CRB*. Reuters, 2005.

[275] S. G. Rhee and R. P. Chang. Intra-day arbitrage opportunities in foreign exchange and eurocurrency markets. *The Journal of Finance*, 47(1):363–379, March 1992.

[276] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 2007.

[277] H. Robinson and H. Sharp. XP culture: why the twelve practices both are and are not the most significant thing. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 12–21, 2003.

[278] D. Rösch and H. Scheule, editors. *Stress Testing for Financial Institutions: Applications, Regulations and Techniques*. Risk Books, London, 2008.

[279] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *Neural Networks, IEEE Transactions on*, 9(6):1456–1470, Nov 1998.

[280] S. Samarasinghe. *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*. Auerbach Publications, Boca Raton, New York, 2007.

[281] A. M. Schaefer, S. Udluft, and H.-G. Zimmermann. A recurrent control neural network for data efficient reinforcement learning. In *Proceedings of the 2007*

IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007), pages 151–157, 2007.

[282] A. M. Schäfer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks — ICANN 2006*, volume 4131 of *Lecture Notes in Computer Science*, pages 632–640. Springer Berlin, Heidelberg, 2006.

[283] M. J. Schervish. P values: What they are and what they are not. *The American Statistician*, 1996.

[284] C. Schittenkopf and G. Dorffner. Risk-neutral density extraction from option prices: improved pricing with mixture density networks. *Neural Networks, IEEE Transactions on*, 12(4):716–725, July 2001.

[285] T. Schmidtke. Optimierung der Marktpreisberechnung von Optionen auf Basis von Web Mining und Künstlichen Neuronalen Netzen. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2006. In German. Diplomarbeit ≈ Master's thesis.

[286] M. Schulz. *Statistical Physics and Economics: Concepts, Tools, and Applications*, volume 184/2003 of *Springer Tracts in Modern Physics*. Springer, New York, Berlin, Heidelberg, 2003.

[287] A. F. Shapiro. A hitchhiker's guide to the techniques of adaptive nonlinear models. *Insurance: Mathematics and Economics*, 26:119–132, 2000.

[288] M. R. E. Shazly and H. E. E. Shazly. Comparing the forecasting performance of neural networks and forward exchange rates. *Journal of Multinational Financial Management*, 7:345–356, 1997.

[289] E. Sinclair. *Volatility Trading*. Wiley, Hoboken, New Jersey, 2008.

[290] C. Slim. Forecasting the volatility of stock index returns: A stochastic neural network approach. *Lecture Notes in Computer Science*, 3045:935–944, 2004.

[291] A. S. Soofi and L. Cao, editors. *Modelling and Forecasting Financial Data: Techniques of Nonlinear Dynamics*. Studies in computational Finance. Kluwer academic publishers, Boston, Dordrecht, London, 2002.

[292] M. Spayd. Evolving agile in the enterprise: implementing XP on a grand scale. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 60–70, 2003.

[293] J. Spence. There has to be a better way! [software development]. In *Agile Conference, 2005. Proceedings*, pages 272–278, 2005.

[294] S. Squartini, A. Hussain, and F. Piazza. Attempting to reduce the vanishing gradient effect through a novel recurrent multiscale architecture. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2819–2824, 2003.

[295] Standard & Poor's. *S&P 500*, June 2009.

[296] F. Stefanini. *Investment strategies of hedge funds*. Wiley, Southern Gate, Chichester, 2006.

[297] J. J. Steil. Online stability of backpropagation-decorrelation recurrent learning. *Neurocomputing*, 69:642–650, 2006.

[298] J. D. Sterman. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw-Hill, 2000.

[299] Stoxx. *Dow Jones Euro Stoxx Size Indices*, July 2009.

[300] T. Stridsman. *Trading Systems That Work: Building and Evaluating Effective Trading Systems*. McGraw-Hill, 2000.

[301] S. J. Taylor. *Modelling Financiel Time Series*. World Scientific, Singapore, 2nd edition, 2008.

[302] S. J. Taylor and X. Xu. The incremental volatility information in one million foreign exchange quotations. *Journal of Empirical Finance*, 4:317–340, 1997.

[303] P. Tenti. Forecasting foreign exchange rates using recurrent neural networks. *Applied Artificial Intelligence*, 10:567–581, 1996.

[304] D. Thomas. Agile programming: design to accommodate change. *IEEE Software*, 22(3):14–16, 2005.

[305] D. Thomas, C. Fowler, and A. Hunt. *Programming Ruby: The Pragmatic Programmers' Guide*. 2005.

[306] P. Tino, C. Schittenkopf, and G. Dorffner. Financial volatility trading using recurrent neural networks. *IEEE Transactions on Neural Networks*, 12(4):865–874, 2001.

[307] N. B. Toomarian and J. Barhen. Learning a trajectory using adjoint functions and teacher forcing. *Neural Networks*, 5:473–484, 1992.

[308] H. Tsangari. An alternative methodology for combining different forecasting models. *Journal of Applied Statistics*, 34(4):403–421, 2007.

[309] E. W. Tyree and J. A. Long. Forecasting currency exchange rates: Neural networks and the random walk model. In *Proceedings of the Third International Conference on Artificial Intelligence Applications on Wall Street, New York*, 1995.

[310] R. Ueda, M. Hiltunen, and R. Schlichting. Applying grid technology to web application systems. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 550–557, 2005.

[311] United Kingdom Debt Management Office. *UK Government Securities: a Guide to Gilts*, 7th edition, June 2009.

[312] J. Utke and U. Nauman. Openad: User manual. Argonne National Laboratory, November 2006.

[313] T. Van Gestel, J. A. K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on*, 12(4):809–821, Jul 2001.

[314] N. G. van Kampen. Remarks on non-markov processes. *Brazilian Journal of Physis*, 28(2):90–96, June 1998.

[315] B. Vanstone and G. Finnie. *Combining Technical Analysis and Neural Networks in the Australian Stockmarket*. School of Information Technology, Bond University, 2006. Information Technology Papers.

[316] B. Vanstone and G. Finnie. Enhancing Existing Stockmarket Trading Strategies Using Artificial Neural Networks: A Case Study. *Lecture Notes In Computer Science*, 4985:478–487, 2008. Preprint from School of Information Technology, Bond University.

[317] B. Vanstone and G. Finnie. An empirical methodology for developing stockmarket trading systems using artificial neural networks. *Expert Systems with Applications*, 36:6668–6680, 2009. Preprint from School of Information Technology, Bond University.

[318] B. J. Vanstone. *Trading in the Australian Stockmarket using Artificial Neural Networks*. PhD thesis, School of Information Technology, Bond University, November 2005.

[319] B. J. Vanstone and T. Hahn. *Creating short-term stockmarket trading strategies using Artificial Neural Networks: A Case Study*. School of Information Technology, Bond University, 2008. Information Technology Papers.

[320] G. Vidyamurthy. *Pairs Trading: Quantitative Methods and Analysis*. Wiley, Hoboken, New Jersey, 2004.

[321] B. Volckaert, P. Thysebaert, M. De Leenheer, F. De Turck, B. Dhoedt, and P. Demeester. A distributed resource and network partitioning architecture for service grids. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 426–433, 2005.

[322] H. von Jouanne-Diedrich, H.-G. Zimmermann, R. Grothmann, and L. Bertolini. *Dow Jones Index Option Trading with Advanced Neural Networks*. Siemens Corporate Technology, December 2008. Talk given during GOR session.

[323] H.-J. von Mettenheim. Derivative design: Computation and optimization of the Black-Scholes equation solution with a Crank-Nicolson scheme and SQP-methods. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover. Diplomarbeit ≈ Master's thesis.

[324] H.-J. von Mettenheim. Entwicklung der grob granularen Parallelisierung für den Neurosimulator FAUN 1.0 und Anwendungen in der Wechselkursprognose. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2003. In German. Diplomarbeit ≈ Master's thesis.

[325] H.-J. von Mettenheim and M. H. Breitner. Coarse-grained parallelization of the advanced neurosimulator FAUN 1.0 with pvm and the enhanced cornered rat game revisited. *International Game Theory Review*, 7, September 2005.

[326] H.-J. von Mettenheim and M. H. Breitner. Neural network forecasting with high performance computers. In *Proceedings of the 13th International Workshop on Dynamics and Control: Modeling and Control of Autonomous Decision Support Based Systems, Wiesensteig 2005*, May 2005.

[327] H.-J. von Mettenheim and M. H. Breitner. Distributed neurosimulation. In H.-D. Haasis, H. Kopfer, and J. Schönberger, editors, *Operations Research Proceedings 2005: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Bremen, September 7–9, 2005*. Springer Verlag, Heidelberg, 2006.

[328] H.-J. von Mettenheim and M. H. Breitner. Derivative design. In C. Dunis, M. Dempster, and V. Terraza, editors, *Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, Luxembourg, 27–29 May 2009*, 2009.

[329] H.-J. von Mettenheim and M. H. Breitner. Numerical solution of the game of two cars with a neurosimulator and grid computing. In P. Bernhard, V. Gaitsgory, and O. Pourtallier, editors, *Advances in Dynamic Games and Their Applications: Analytical and Numerical Developments*. Birkhäuser, Boston, 2009.

[330] C. Wang and J. C. Principe. Training neural networks with additive noise in the desired signal. *IEEE Transactions on Neural Networks*, 10(6):1511–1517, 1999.

[331] T. Weise. *Global Optimization Algorithms: Theory and Applications*. Published online at `http://www.it-weise.de`, 2nd edition, 2009.

[332] R. L. Weissman. *Mechanical Trading Systems*. Wiley, Hoboken, New Jersey, 2005.

[333] T. Weithers. *Foreign Exchange: A Practical Guide to the FX Markets*. John Wiley, Hoboken, New Jersey, 2006.

[334] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[335] D. West and M. Solano. Metaphors be with you! (metaphor system). In *Agile Conference, 2005. Proceedings*, pages 3–11, 2005.

[336] H. White and J. Racine. Statistical inference, the bootstrap, and neural-network modeling with application to foreign exchange rates. *Neural Networks, IEEE Transactions on*, 12(4):657–673, Jul 2001.

[337] K.-P. Wiedmann. Neuronale Netze als Basis eines effizienten Zielkundenmanagements in der Finanzdienstleistungsbranche. In K.-P. Wiedmann and F. Buckler, editors, *Neuronale Netze im Marketing-Management*, pages 241–274. Gabler, 2003. In German.

[338] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[339] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, architectures, and applications*, pages 433–486, 1995.

[340] P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives*. Cambridge University Press, 2002.

[341] W. Wood and W. Kleb. Exploring XP for scientific research. *IEEE Software*, 20(3):30–36, 2003.

[342] H. Xia, H. Dail, H. Casanova, and A. Chien. The microgrid: using online simulation to predict application performance in diverse grid network environments. In *Challenges of Large Applications in Distributed Environments, 2004. CLADE 2004. Proceedings of the Second International Workshop on*, pages 52–61, 2004.

[343] L. Xin, P. L. H. Yu, and K. Lam. Effectiveness of filter trading as an intraday trading rule. In C. Dunis, M. Dempster, and V. Terraza, editors, *Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, Luxembourg, 27–29 May 2009*, 2009.

[344] Z. Xu, C. Shu, H. Yu, and H. Liu. An agile programming model for grid end users. In *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, pages 3–7, 2005.

[345] J. Yao, H.-I. Poh, and T. Jasic. Foreign exchange rates forecasting with neural networks. In *Proceedings of the International Conference Neural Information Processing, Hong Kong*, pages 754–759, 1996.

[346] J. Yao and C. L. Tan. A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing*, 34:79–98, 2000.

[347] J. Yao and C. L. Tan. Guidelines for financial forecasting with neural networks. In *Proceedings of the International Conference on Neural Information Processing, Shanghai*, 2001.

[348] J. Yim. A comparison of neural networks with time series models for forecasting returns on a stock market index. *Lecture Notes in Computer Science*, 2358:53–63, 2002.

[349] L. Yu, S. Wang, W. Huang, and K. K. Lai. Are foreign exchange rates predictable? a survey from artificial neural networks perspective. *Scientific Inquiry*, 8(2):207–228, 2007.

[350] L. Yu, S. Wang, and K. K. Lai. A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates. *Computers and Operations Research*, 32:2523–2541, 2005.

[351] L. Yu, S. Wang, and K. K. Lai. *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks*. International Series in Operations Research and Management Science. Springer, New York, 2007.

[352] L. Yu, S. Wang, K. K. Lai, and W. W. Huang. Developing and assessing an intelligent forex rolling forecasting and trading decision support system for online e-service. *International Journal of Intelligent Systems*, 22:475–499, 2007.

[353] H. Zakhariya. Personaleinsatzplanung im Echtzeitbetrieb in Call Centern mit Künstlichen neuronalen Netzen. Master's thesis, Institut für Wirtschaftsinformatik, Wirtschaftswissenschaftliche Fakultät, Leibniz Universität Hannover, 2007. In German. Diplomarbeit ≈ Master's thesis.

[354] B.-L. Zhang, R. Coggins, M. A. Jabri, D. Dersch, and B. Flower. Multiresolution forecasting for futures trading using wavelet decompositions. *Neural Networks, IEEE Transactions on*, 12(4):765–775, Jul 2001.

[355] G. Zhang and M. Y. Hu. Neural network forecasting of the British Pound/US Dollar exchange rate. *Omega, International Management Science*, 26(4):495–506, 1998.

[356] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14:35–62, 1998.

[357] M. Zhang, S. Xu, and J. Fulcher. Neuron-adaptive higher order neural-network models for automated financial data modelling. *IEEE Transactions on Neural Networks*, 13(1):188–204, 2002.

[358] M. Zhao and T. Zhang. A heuristic for scheduling parallel programs with synchronous communication model in the network computing environments. In *NPC*, pages 270–273, 2005.

[359] H. G. Zimmermann. *Advanced Neural Networks in System Identification, Forecasting and Control*. Siemens, Corporate Technology, June 2009. Workshop during the 29th Annual International Symposium on Forecasting, Hong Kong, June 21–24, 2009.

[360] H. G. Zimmermann and R. Grothmann. *Forecasting Commodity Prices with Dynamical Consistent Neural Networks*. Siemens, Corporate Technology, December 2008. Talk during GOR Session.

[361] H. G. Zimmermann, R. Grothmann, A. M. Schaefer, and C. Tietz. Modeling large dynamical systems with dynamical consistent neural networks. In S. Haykin, J. C. Principe, T. J. Sejnowski, and J. McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brain*, pages 214–255. MIT Press, 2006.

[362] H. G. Zimmermann, R. Neuneier, and R. Grothmann. Modeling of dynamic systems by error correction neural networks. In *Modeling and Forecasting Financial Data: Techniques of Nonlinear Dynamics*. Kluwer Academic, 2000.

[363] H. G. Zimmermann, R. Neuneier, and R. Grothmann. Multi-agent modeling of multiple fx-markets by neural networks. *Neural Networks, IEEE Transactions on*, 12(4):735–743, Jul 2001.

[364] H. G. Zimmermann, C. Tietz, and R. Grothmann. Yield curve forecasting by error correction neural networks and partial learning. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2002, Bruges, Belgium*, pages 407–412, 2002.

[365] Y. Zu, R. Xiao, and X. Zhang. Automated conceptual design of mechanisms using enumeration and functional reasoning. *International Journal of Materials and Product Technology*, 34(3):273–294, 2009.

[366] Y. Zu-Qiao, X. Xiao-Hong, and G. Han-ping. An improved dm algorithm based on rough set theory. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 3097–3100, Sept. 2007.

Index

Curriculum Vitae and Publications

Contact

Dr. rer. pol. Hans-Jörg von Mettenheim (Dipl.-Ök., Dipl.-Math.)
Institut für Wirtschaftsinformatik
Königsworther Platz 1
30167 Hannover, Germany

tel. +49.511.762-4982
email mettenheim@iwi.uni-hannover.de

Education

until July 1999 Leibnizschule Hannover
until July 2003 Studying Economics at Leibniz Universität Hannover
until August 2008 Studying Mathematics at Leibniz Universität Hannover
August 2003 till present PhD student and stipendiary, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover

Degrees

July 1999 General qualification for university entrance ("Abitur", grade average 1.0)
July 2003 Diploma in Economics (thesis *Entwicklung der grob granularen Parallelisierung für den Neurosimulator FAUN 1.0 und Anwendungen in der Wechselkursprognose*, grade 1.0)
August 2008 Diploma in Mathematics (thesis *Derivative Design: Computation and Optimization of the Black-Scholes Equation Solution with a Crank-Nicolson Scheme and SQP-Methods*, grade 1.0)
December 2009 Dr. rer. pol. / PhD in Economics (thesis *Advanced Neural Networks: Finance, Forecast And Other Applications*, grade *summa cum laude*)

Languages

French mother tongue
German father tongue
English fluent
Czech entry level
Latin "großes Latinum"
Ancient Greek "Graecum"
Ancient Hebrew "Hebraicum"

Awards and Scholarships

1999–2003 Diploma scholarship of *Studienstiftung des Deutschen Volkes*
2005–2009 PhD scholarship at *Institut für Wirtschaftsinformatik*
2006 Best young researcher award for *Dynamic Games with Neurosimulators and Grid Computing: The Game of Two Cars Revisited*, coauthored with Michael H. Breitner, at the 12th International Symposium on Dynamic Games and Applications, July, 3rd–6th, 2006, Sophia Antipolis, France
2009 Best thesis award of the Hanover Center of Finance e. V., sponsored by Norddeutsche Landesbank Hannover, for the diploma thesis *Derivative Design: Computation and Optimization of the Black-Scholes Equation Solution with a Crank-Nicolson Scheme and SQP-Methods*

Other Qualifications and Activities

C Examination Organ playing and choir supervision
C additional qualification Jazz, Rock and Pop in the church
Organist Apostelkirche Hannover
Musical Responsibility Taizé Prayer, Apostelkirche Hannover

Publications

Where available the ranking of the publication according to the most relevant German Economics rankings is given in parantheses using following abbreviations:
HB (= HB BWL) The German business newspaper *Handelsblatt* evaluates 761 journals and conferences for Business Administration and Management. The current ranking is from 2009. The best available ranking is 1.0 Handelsblatt

points.

JQ2 (= VHB JQ2) The Jourqual 2 ranking was published in 2008 and evaluates 756 journals and conferences for Business Administration and Management under supervision of the German Verband der Hochschullehrer für Betriebswirtschaft e. V. (VHB). Methodological details of the first Jourqual 1 ranking can be found in: Hennig-Thurau, Thorsten; Walsh, Gianfranco; Schrader, Ulf (2004): *VHB-JOURQUAL: Ein Ranking von betriebswirtschaftlich-relevanten Zeitschriften auf der Grundlagen von Expertenurteilen*, Zeitschrift für betriebswirtschaftliche Forschung (zfbf), Vol. 56, pages 520–543. The best available ranking is A+.

WI (= VHB WKWI & GI FB WI) This is a ranking targeted at Information Systems Research und supervision of both the German VHB Wissenschaftliche Kommission Wirtschaftsinformatik (VHB WKWI) and the Gesellschaft für Informatik e. V. Fachbereich Wirtschaftsinformatik (GI FB WI). It evaluates 176 journals and conferences in the field. It was first published in the February 2008 volume of *Wirtschaftsinformatik* by the German scientific commission for Information Systems Research (WKWI), pages 155–163. The best available ranking is A.

1. *Plattformunabhängiges Softwareengineering eines Transportmodells zur ganzheitlichen Disposition von Strecken- und Flächenverkehren*, IWI Discussion Paper #38, about 40 pages, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, to appear January/February 2010 (in German, with Tim Rickenberg and Michael H. Breitner)

2. *Prognose und Handel von Öl-Future-Spreads durch Multi-Layer-Perceptrons und High-Order-Neuronalnetze mit FAUN 1.1*, IWI Discussion Paper #35, 39 pages, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, September 18th, 2009 (in German, with Christoph Polus and Michael H. Breitner)

3. *Prognose und Handel von Derivaten auf Strom mit Künstlichen Neuronalen Netzen*, IWI Discussion Paper #34, 29 pages, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, September 11th, 2009 (in German, with Horst-Oliver Hofmann and Michael H. Breitner)

4. *Derivative Optimization and Design* in Valerie Belton, Erwin Pesch, Gerhard J. Woeginger (eds.), Proceedings of the 23rd European Conference on Operational Research, July 5–8, 2009, Universität Siegen, Bonn (with Michael H. Breitner)

5. *Multi-Objective Optimization for Planning of Central IT Resources with Focus on Green IT* in Valerie Belton, Erwin Pesch, Gerhard J. Woeginger (eds.)i, Proceedings of the 23rd European Conference on Operational Research, July 5–8, 2009, Universität Siegen, Bonn (with Marc Klages and Michael H. Breitner)

6. *Numerical Solution of the Game of Two Cars with a Neurosimulator and Grid Computing* in Annals of the International Society for Dynamic Games (ISDG) Vol. 10, pp. 207–230, Birkhäuser, Boston, May 2009 (with Michael H. Breitner, internationally renowned annals for Dynamic Games, no German ranking due to only few German researchers)

7. *Derivative Design* in Christian Dunis, Michael Dempster, Virginie Terraza (eds.), Proceedings of the 16th International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management, May 27–29 2009, Luxembourg (with Michael H. Breitner)

8. *Industrialization of Derivative Design: Integrated Risk Management with the Financial Information System WARRANT-PRO-2* in Hans Robert Hansen, Dimitris Karagiannis, Hans-Georg Fill (eds.), Business Services: Konzepte, Technologien, Anwendungen, 9. Internationale Tagung Wirtschaftsinformatik, February 25–27 2009, Vienna, Volume 2, pp. 255–264 (with Michael H. Breitner, HB 0.1, JQ2 C, WI A)

9. *Entwicklung des Hannoveraner Referenzmodells für Sicherheit und Evaluation an Fallbeispielen*, IWI Discussion Paper #32, 31 pages, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, February 18th, 2009 (in German, with Sebastian Schmidt and Michael H. Breitner)

10. *Ganzheitliche Disposition von Strecken- und Flächenverkehren durch kombinierten Einsatz modifizierter Operations Research Verfahren* in Proceedings der Fachtagung der GOR AG Logistik und Verkehr, 13./14. 11. 2008, Frankfurt am Main (in German, with Marcus Gerasch, Michael H. Breitner and Lothar Schulze)

11. *Akzeptanz von Sicherheitsmaßnahmen: Modellierung, Numerische Simulation und Optimierung*, IWI Discussion Paper #28, 30 pages, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, October 16th, 2008 (in German, with Matthias Paul and Michael H. Breitner)

12. *Intelligent Decision Support Systems and Neurosimulators: A Promising Alliance for Financial Services Providers* in H. Österle, J. Schelp, R. Winter (eds.), Proceedings of ECIS 2007, St. Gallen, 7th–9th June 2007 (with Michael H. Breitner, Frank Köller and Simon König, HB 0.2, JQ2 B, WI A)

13. *Distributed Neurosimulation* in H.-D. Haasis, H. Kopfer, J. Schönberger (eds.), Operations Research Proceedings 2005, Selected Papers of the Annual International Conference of the German Operations Research Society, Bremen, Springer Verlag, Heidelberg, published September 2006, 6 pages (with Michael H. Breitner)

14. *Dynamic Games with Neurosimulators and Grid Computing: The Game of Two Cars Revisited* in Proceedings of the 12th International ISDG Symposium on Dynamic Games and Applications, July 3rd–6th, 2006, Sophia Antipolis/Riviera, 24 pages (with Michael H. Breitner)

15. *Coarse-grained Parallelization of the Advanced Neurosimulator FAUN 1.0 with PVM and the Enhanced Cornered Rat Game Revisited*, International Game Theory Review (IGTR), Vol. 7, No. 3, pp. 1–19, September 2005 (with Michael H. Breitner, internationally renowned journal for (Dynamic) Game Theory, no German ranking due to only few German researchers)

16. *Neural Network Forecasting with High Performance Computers* in E. P. Hofer, E. Reithmeier (eds.), Proceedings of the 13th International Workshop on Dynamics and Control — Modeling and Control of Autonomous Decision Support Based Systems, Wiesensteig, Shaker, Aachen, May 2005, 9 pages (with Michael H. Breitner)

17. *Coarse-grained Parallelization of the Advanced Neurosimulator FAUN 1.0 with PVM* in Michael H. Breitner (ed.), Proceedings of the Fourth International ISDG Workshop, Goslar, May 19th–21st, 2003, Institut für Wirtschaftsinformatik, Leibniz Universität Hannover, 2003, 15 pages (with Michael H. Breitner)