MULTI-AGENT MARKET MODELING BASED ON NEURAL NETWORKS

by

RALPH GROTHMANN

University of Bremen, Germany
Siemens AG, Corporate Technology, Munich, Germany

Thesis presented for the Degree of Doctor of Economics (Dr. rer. pol.)
Faculty of Economics, University of Bremen, Germany

Contents

Acronyms

adaline	adaptive linear element
approx.	approximately
AR	autoregressive
ARIMA	autoregressive integrated moving average
ARMA	autoregressive moving average
CARA	constant absolute risk aversion
chp.	chapter
CRB	Commodity Research Bureau
CRRA	constant relative risk aversion
EBD	early brain damage
ECNN	error correction neural network
Ed.	editor
EMH	efficient market hypothesis
et al.	et aliter
etc.	et cetera
eq.	equation
e. g.	exempli gratia
fig.	figure
FTSE 100	Financial Times Stock Exchange 100 stock index
FX	foreign exchange
FX-market	foreign exchange market
FX-rate	foreign exchange rate

GARCH	generalized autoregressive conditional heteroskedasticity
DAX	German stock market index
DEM	German Mark
i. e.	id est
Ind.	index
LMS	least mean square
MIC	maximum intertemporal connectivity
MLP	multi-layer perceptron
MSE	mean square error
NARMA	nonlinear autoregressive moving average
NARX	nonlinear autoregressive with exogenous inputs
NYSE	New York stock exchange
OBD	optimal brain damage
OLS	ordinary least squares
Oz.	ounce
PCA	principal component analysis
RBF	radial basis function
REE	rational expectations equilibrium
resp.	respectively
RNN	recurrent neural network
sec.	section
SR	Sharpe ratio
SRN	simple recurrent network
SOM	self-organizing map
tab.	table
TAR	threshold autoregressive
USD	US-Dollar
VAR	vector autoregressive
vs.	versus
YEN	Japanese Yen

List of Figures

List of Tables

Chapter 1

INTRODUCTION

One of the challenges of financial research is to develop models that are capable of explaining and forecasting market price movements and returns. Basically, there are two approaches to the modeling of financial markets: The first one is from an econometric point of view, while the second one is given by (micro-)economic multi-agent theory. Let us contrast these perspectives of financial market modeling and point to major difficulties of each research direction.

Econometrics

In econometrics, the behavior of financial markets is described by quantitative variables. Mathematical and statistical methods are used to explore economic relationships among the quantitative variables and to forecast the future development of the market prices and returns [PDP00, p. 199].

The quantitative variables included in an econometric model are usually aggregates like prices, interest rates or market indices. An aggregate is a superposition of the decision-taking processes of individual economic agents. Dealing with aggregates, one abstracts from the underlying decisions of a larger number of agents. In other words, econometric models do not concern the decisions of individual agents, but focus on the net outcome of the agents' interaction which is reflected in the market aggregates. Such a simplification lacks explanatory power, since market prices are not explained by a causal analysis of the behavioral decisions of interacting agents. However, an econometric approach opens the chance to integrate a lot of quantitative dynamics into the modeling. Hence, we are able to fit an econometric model to real-world financial data and to predict market price changes.

There is a broad spectrum of econometric approaches to the modeling of market price and return time series. A common classification of econometric techniques is outlined in Tab. 1.1 [Pod99, Zim94, p. 399 and p. 13].[1]

Table 1.1 Classification of econometric techniques.

	univariate	**multivariate**
linear	autoregression, ARIMA	linear regression
nonlinear	Threshold autoregression, neural networks	logistic regression, neural networks

According to Tab. 1.1, econometric methods can be classified by their ability to model *linear* and *nonlinear* relationships among the concerned variables. Within each class, one may distinguish between *univariate* and *multivariate* methods. The main focus of this work is on neural networks. – Readers who are interested in further details and comprehensive overviews of the econometric methods listed in Tab. 1.1 should refer to e. g. Wei (1990) [Wei90], Poddig (1999) [Pod99] or Poddig et al. (2000) [PDP00].

From an econometric point of view, the application of neural networks is often motivated by the fact, that a 3-layer feedforward neural network has the ability to approximate any structure contained in a data set (so-called approximation theorem) [HSW92, p. 14-20]. Furthermore, neural networks allow the construction of models, which are able to handle high-dimensional problems along with a high degree of nonlinearity [Zim94, p. 14]. Therefore, neural networks are an ideal modeling framework for complex economic systems, which typically show high-dimensional inter-variable dependencies as well as non-linear behavior [Zim94, p. 14].

However, this statistical view on neural networks has some major drawbacks: Referring to the approximation theorem, the task of modeling a time series is transferred into a pattern recognition approach. Consequently, the quality of the available data determines the performance of the resulting model [NZ98, p. 373]. This is especially crucial in financial applications, because financial data is often unreliable due to noise, outliers, missing values or unknown external influences [NZ98, p. 373]. Another well-known problem of neural networks is overfitting

[1]For details on ARIMA (Autoregressive integrated moving average) see Box and Jenkins (1976) [BJ76, p. 89-90]. Threshold autoregression is outlined in Tong and Lim (1980) [TL80].

[Pod99, Zim94, p. 430 and p. 58-60]. A complex neural network may not only fit the underlying structure of the time series, but also the noise included in the data. Overfitting is dangerous, because the network loses its generalization abilities.

Economic Theory of Multi-Agent Market Modeling

In contrast to an econometric approach, (micro-)economic models focus directly on the underlying structure of the market. The basic idea is, that the market price dynamics arises from the interaction of many individual agents [Bar97, LeB00, p. 813-4 and p. 679-81].

Approaching financial markets in this manner, one starts off with the modeling of the agents' decision making schemes on the microeconomic level of the market. Thereafter, market price changes can be determined on the macroeconomic level by a superposition of the agents' buying and selling decisions. An example of such a price formation mechanism is to define a so-called market impact function [Far98, p. 7-14]. In this case, one assumes that price changes are performed by a market maker in response to the observed market excess (i. e. demand - supply). As it can be seen from this outline, the aim of a (micro-)economic model is to explain market prices by a detailed causal analysis of the agents' decision making behavior. The market price results from an aggregation of the agents' decisions. Following this concept, microeconomic market models not only give a phenomenological description of price changes (as in econometrics) but offer an explanation of price shifts by excess demand or supply.

Remarkably, agent-based financial markets provide a new explanatory framework supplementing the traditional economic concepts of equilibrium theory and efficient markets [Art95, Hom01]. Such a supplementing framework is needed, because in real-world financial markets the underlying assumptions of equilibrium or efficient market theory are often violated [Far99a, p. 9991-2]. For example, assumptions like perfect rationality, common expectations or complete information are typically not valid, because "rationality is difficult to define, human behavior is often unpredictable, information can be difficult to interpret, technology and institutions change constantly, and there are significant 'frictional' costs to gathering and processing information, and transacting" [Far99a, p. 9992]. In this connection, multi-agent models give us a rise to the dynamics of real-world financial markets from the perspective of individuals (agents) interacting with other market participants. Being so, multi-agent models are able "to capture complex learning behavior and dynamics in financial markets using more realistic markets, strategies, and information structures" [Far99a, p. 9992]. Thus, multi-agent models give an explanation of many facets of financial markets which lie not within the scope of the efficient market theory [Far99a, p. 9992].

Examples of agent-based financial markets analyzing market anomalies (i. e. deviations from standard equilibrium theory) are Levy et al. (1994) [LLS94], Kaizoji (2000) [Kai00], Joshi, Parker and Bedau (1999) [JPB00] or Lux (1998) [Lux98]. Levy et al. (1994) [LLS94, p. 104-6] use heterogeneous agents, who differ in their memory spans and are influenced by individual psychological factors. The authors found, that if the investors are too homogeneous, market crashes are likely to appear. Kaizoji (2000) [Kai00, p. 501-5] points out, that stock market bubbles and crashes are due to the speculative activity of interacting agents. Joshi, Parker and Bedau (1999) [JPB00, p. 472-7] use their agent-based model to provide an explanation for the profitability of technical trading rules. Finally, the interaction of the agents in the model of Lux (1998) [Lux98, p. 155-62] generates stylized facts of financial markets like e. g. volatility clustering, fat tailed return distributions or persistence in volatility and volume. As it can be seen from these examples, the analysis of the individual behavior and interaction of the agents supplements standard economic theory and provides deeper insights into the dynamics of financial markets [Far99a, LAHP97, p. 9991-2, p. 1-2].

However, agent-based models of financial markets are typically intractable, because they come along with high degrees of freedom and complexity [LeB01a, p. 255]. As a consequence, it is neither possible to fit an agent-based model to real-world financial data, nor to generate reasonable forecasts. Up to now, multi-agent models only provide *qualitative* results, i. e. the interaction of the agents gives 'only' an explanation for complex economic phenomena [Far99a, LeB01a, p. 9992 and p. 259-60].

Another drawback of nearly all multi-agent models is that the decision making schemes of the agents do not contain semantic specifications. Instead of modeling the underlying cognitive processes of the agents, one often assumes arbitrary ad-hoc functional relationships for the agents' cognition. This is clearly an unrealistic approach to the modeling of the behavior of real-world market participants. Typically, the decision schemes of the agents are adapted from the field of econometric time series processing. Among these econometric methods, one may e. g. refer to feedforward neural networks (Beltratti et al. (1996) [BMT96, p. 223-6]), recurrent neural networks (Yang (1999) [Yan99]) or autoregression models (Brock et al. (1997) [BH97]). However, more realistic approaches to the modeling of the agents' behavior are desirable [EM01, p. 759-61].

Our Vision: Combining Econometrics and Economic Theory of Multi-Agent Modeling

The purpose of this work is to combine the two outlined perspectives in a unified approach of market modeling. Merging the econometric and the (micro-)economic framework of market modeling, we want to benefit from the advantages of each approach without bearing its drawbacks. In other words, we want to combine the predictive power of econometric models with the explanatory skills of (micro-)economic market models.

In order to establish the cross-link between econometric and economic market modeling, we refer to feedforward and time-delay recurrent neural networks as an appropriate framework for the modeling of high dimensional nonlinear dynamical systems [NZ98, ZNG01a, p. 375-86 and p. 246-60]. As we will show, neural networks allow the integration of the decision behavior of individual economic agents into a market model. Based on the perspective of interacting agents, the resulting market model allows us to capture the underlying dynamics of financial markets, to fit real-world financial data, and to forecast future market price movements.

Furthermore, our intention is to provide a more realistic approach to the modeling of the agents' decision making schemes. This approach is based on the idea of a basic cognitive system. Agents perceive the development of the market and initiate actions on the basis of their internal expectations. The structural representation of the cognitive system is given by time-delay recurrent neural networks.

As a side aspect of this work, we point out that neural networks allow to set up a joint framework of econometric model building [NZ98, ZN01, p. 373-4 and p. 350]. Following this modeling philosophy, neural networks are ideal for the modeling of complex (economic) systems: Besides the learning from data, one may integrate prior knowledge about the underlying dynamical system into the modeling. Furthermore, the modeling can be enhanced by the integration of first principles, e. g. the separation of variants and invariants in high-dimensional dynamical systems. These additional elements of the model building are incorporated into the neural networks in form of architectural enhancements [NZ98, ZN01, ZNG01a]. This way of model building helps to overcome the drawbacks of purely data driven approaches and may reduce overfitting [NZ98, ZN01, p. 373-4 and p. 311].

Organization of this Work

To establish our idea of an agent-based financial market on the basis of neural networks, this work consists of eight main chapters. Chapters

2 to 5 are written from the perspective of econometrics. Besides a basic introduction to neural networks and historical notes on the research field, we introduce feedforward and time-delay recurrent network architectures that are in turn used to develop our multi-agent models. Chapters 6 to 8 deal with the economic theory of agent-based financial markets. While chapter 6 outlines recent developments in the literature of multi-agent market modeling, chapters 7 and 8 introduce our contributions to this research field. Here, we utilize the feedforward and time-delay recurrent neural networks that are developed in the econometric part of this work. We show, how these neural network architectures can serve as a structural representation of the agents' decision making schemes. Finally, chapter 9 summarizes major results and contributions of this work and gives an outlook to future directions of research. Let us outline the contents of chapters 2 to 9 in greater detail.

Chapter 2: Neural Networks – A Basic Introduction and Historical Notes

Chapter 2 deals with a brief introduction to neural networks and some historical notes on the research field.

First, we consider the biological foundation of neural networks [Hay94, p. 1-34] and point out, that a neural network can also be seen as the interaction of many (economic) decisions [Zim89, Zim94, p. 496-8 and p. 6-10]. Another major topic of the chapter is a discussion of the well-known 3-layer feedforward neural network with a particular focus on its disadvantages. As a remedy, we suggest to incorporate prior knowledge about the specific task into the model building process [NZ98, ZN01, ZNG01a].

Thereafter, we introduce important milestones in the research area of neural networks. Beginning with McCulloch and Pit's model of a neuron (1943) [MP43], we deal with Rosenblatt's perceptron (1958) [Ros58] and Minsky and Papert's related critics (1969) [MiPa69]. Furthermore, we briefly consider Kohonen's theory of self-organizing feature maps (1982) [Koh88], standard error backpropagation (1974 & 1986) [Wer74, RHW86], Hopfield's fix-point recurrent neural networks (1982) [Hop92], the Boltzmann machine (1985) [AHS85], Broomhead and Lowe's radial-basis function networks (1988) [BrLo88] and Elman's recurrent neural networks (1990) [Elm90]. The historical notes are completed by focusing on the dilemma of overfitting and purely data driven modeling (1990s) [Zim94, NZ98, p. 58-61 and p. 410-18].

Chapter 3: Modeling Dynamical Systems by Feedforward Neural Networks

Chapter 3 is devoted to the modeling of dynamical systems by feedforward neural networks. In particular, we introduce the *11-layer architecture* of Neuneier and Zimmermann (1998) [NZ98, p. 375-86], which consists of several architectural building blocks.

The different building blocks fulfill specific tasks in the modeling of dynamical systems. For example, a network internal data preprocessing is responsible for the cancellation of outliers [NZ98, p. 375-6]. A so-called *square* layer is used for a differentiation and similarity analysis by merging the properties of multi-layer perceptrons and radial basis functions (RBF) [NZ98, p. 378-79]. Additional information about the underlying dynamic system is incorporated into the modeling by using the concept of forces and embeddings [NZ98, p. 383-6]. This concept is related to the *Takens-Theorem* [Tak81] and allows a complete characterization of the underlying time series dynamics. Finally, a series of forecasts is computed on the basis of the forces and embeddings. An average of these forecasts may improve the model's prediction accuracy under the assumption that the individual errors of the forecasts are uncorrelated [NZ98, p. 381-2].

In combination, these building blocks allow an advanced description of the underlying system dynamics and improve forecasting [NZ98, p. 420]. We illustrate the performance of the 11-layer network in an empirical study modeling the German bond market.

Chapter 4: Modeling Dynamical Systems by Time-Delay Recurrent Neural Networks

In chapter 4, we focus on the modeling of dynamic systems by time-delay recurrent neural networks [ZN01, ZNG01a]. We start off with a discussion of partly autonomous and partly external driven systems (so-called open systems). As a natural representation of open systems we propose time-delay recurrent neural networks. The associated parameter optimization task is solved by unfolding in time [RHW86, p. 354-7]. This means that we transform the temporal system identification task into a spatial architecture, which can be handled by a shared weights extension of standard backpropagation [Wer94, p. 269-294]. The learning of the autonomous dynamics in a partially external driven system can be enforced by overshooting [ZN01, p. 326].

Having explained these basics of time-delay recurrent networks, we introduce error correction neural networks (ECNN) [ZNG01a, p. 247-9]. As a remarkable property, an ECNN includes the last measured model error as an additional input. Hence, the learning can interpret the model's misspecification as an external shock, which can be used to guide the model dynamics afterwards.

As extensions to the basic recurrent network architecture and the ECNN, we apply techniques like undershooting, variants-invariants separation and unfolding in space and time [ZN01, ZNG01a, p. 330-2 and p. 250-60]. These first principles are integrated into the neural networks in form of architectural enhancements. By this, additional prior knowledge about the dynamics, which is *not* included in the training data, is incorporated into the modeling. Let us briefly describe the different building blocks.

Undershooting focuses on the relationship between the time grid of the model and that of the data [ZN01, ZNG02, p. 330-2 and p. 395-9]. This feature is an important prestructuring element for the modeling of dynamic systems. The time grid of the data is typically the same as the time grid of the model. We show that a refinement of the model time grid relative to a wider-meshed data time grid provides deeper insights into the dynamics. This undershooting can be derived from the principle of uniform causality.

A variants-invariants separation in form of a bottleneck coordinate transformation, enables us to handle high-dimensional forecasting problems [RHW86, CS97, ZN01, p. 335-9, p. 101-7 and p. 341-2]. The dimension reduction of the bottleneck network singles out the time variant structures of the dynamics. Clearly, we only have to build a forecast model for the time variants of the system, whereas the invariants remain constant over time. Having predicted the variants, a recombination of the variants and invariants allows us to describe the future development of the complete dynamical system.

In optimal state space reconstruction we try to specify a coordinate transformation such that the related forecast problem becomes easier, i. e. it evolves more smoothly over time. Here, we propose an integrated neural network approach which combines state space reconstruction and forecasting. This is referred to as unfolding in space & time [ZN00a, ZN01, p. 259-64 and p. 342-50].

Experimental results with real-world financial data indicate, that the incorporation of the different first principles leads to a strong improvement of the econometric model building.

Chapter 5: Training of Neural Networks by Error Backpropagation

Chapter 5 deals with the training of neural networks. Training is a synonymical expression for optimizing the network parameters, such that the network error (i. e. the difference between the network outputs and related observations) is minimal [Zim94, p. 36-7 and p. 40]. First, we discuss standard error backpropagation for the learning of feedforward neural networks [Wer94, p. 270-9]. Furthermore, we explain the shared weights extension of error backpropagation for the training of unfolded time-delay recurrent networks [RHW86, p. 354-7].

Since backpropagation is only an efficient way of calculating the partial derivatives of the network error function with respect to the network weights, we also introduce two stochastical learning rules, namely pattern-by-pattern and vario-eta learning [Zim94, NZ98, p. 40-51 and p. 395-7]. These learning rules allow the adaptation of the weights such that the overall error function of the network is minimal. In addition, we focus on the problem of outliers in the training data, which may cause unreliable model forecasts. Here, robust error and activation functions can be seen as a remedy [NZ98, p. 387-8]. Furthermore, we discuss the partial learning weight modification rule and weight pruning techniques. These methods can be utilized to optimize the structure of neural network [ZGT02, NZ98, p. 407-12 and p. 405-410].

We conclude chapter 5 with a comparison of pattern-by-pattern and vario-eta learning. More precisely, we consider the estimation of the maximum intertemporal connectivity (MIC) [ZN01, p. 326]. The MIC is the finite truncation length of an unfolding in time recurrent neural network. The estimation of the MIC is illustrated in an empirical study modeling the short-term German interest rate. We show that a renormalization of the gradients as performed during vario-eta learning supports the estimation of the MIC, because we are able to detect long-term dependencies in the data. This also improves the performance of our model.

Chapter 6: Multi-Agent Market Modeling – A Guide to Literature

Chapter 6 gives a survey of recent developments in the research field of multi-agent market modeling. In recent literature, multi-agent models are used to analyze real-world market phenomena (e. g. volatility clustering, fat tails or complexity [Lux98, p. 143-7 and p. 155-62]), which are difficult to explain by the standard equilibrium theory of financial markets [Far98, Far99a, LM99, Art95, p. 3-4, p. 9992, p. 498 and p. 20].

Instead of assuming market efficiency [Fam70, Fam91] or rational expectations [Mut61, Luc72], agent-based models incorporate theories of human behavior from psychology or sociology [Far98, p. 5]. This allows us to study a market price dynamics from the perspective of interacting agents and their psychological behavior [PAHL98, Far98, p. 27-8 and p. 4-5]. We will describe this direction of research and argue, that multi-agent models may help to gain a deeper understanding of real-world financial markets [Far99a, LAHP97, p. 9991-2 and p. 1-2].

The research field of multi-agent market modeling is steadily growing and it would be impossible to mention the entire spectrum of papers in the frame of this thesis. Therefore, we decided to focus our attention on major design questions, which have to be answered during the modeling of an agent-based market. The design issues are related to the micro- and macroeconomic structure of the market [LeB01a, WHD02, BMT96, p. 255-9, p. 88-93 and p. 175].

On the microeconomic level, one has to consider the modeling of the agents. Major design issues are e. g. the decision processes of the agents, decision heterogeneity or adaptive decision making [WHD02, LeB01a, p. 90-92, p. 255-56]. The macroeconomic side addresses the modeling of the market structure and the interaction of the agents. Major design issues are the traded assets, the organization of the market, and the market price formation mechanism [WHD02, LeB01a, p. 89-90 and p. 92-93, p. 256-58]. We show how these design issues are addressed in recent multi-agent literature and compare our multi-agent approaches to the surveyed models.

Chapter 7: Multi-Agent FX-Market Modeling by Feedforward Neural Networks

In chapter 7 we introduce a multi-agent model on the basis of feedforward neural networks [ZNG01d]. The basic idea of this approach is that the decision making scheme of an agent can be modeled by a single neuron [Zim89, Zim94, p. 496-7 and p. 3-6]. From this point of view, a neural network reflects the interaction of a large number of decision makers resp. a market process [Zim89, Zim94, p. 497-8 and p. 7-10]. The work of chapter 7 builds upon (*i.*) the feedforward neural networks outlined in chapters 2 and 3, and (*ii.*) the learning algorithms introduced in chapter 5.

Let us assume, that the decision making scheme of an agent consists of three stages. These stages are (*i.*) information filtering, (*ii.*) market evaluation and (*iii.*) acting based on the rated information [Zim94, p. 3-4]. As an example of this decision process, one may refer to a trader dealing in a FX-market. Typically, the trader has access to various information sources (e. g. newspapers or data information systems). In order to handle this flood of information, the agent has to focus his attention on a couple of news, which seem to have the deepest impact on the market. This is referred to as information filtering. By analyzing the collected pieces of information, the agent performs a market evaluation. Based on this evaluation the agent executes an action [Zim94, p. 3-4].

The preceding decision process can be modeled by a single neuron [Zim89, Zim94, p. 496-7 and p. 3-6]: Each input signal of the neuron is associated with a particular weight. Information filtering takes place by adjusting the weights of non-essential information towards zero, while important data may be emphasized by higher weighting factors. The market evaluation is represented by the adder of the neuron, which generates a superposition of information (net input). The net input of the neuron is compared to a bias and transferred through the activation function. The resulting output signal can be seen as the action of the trader.

Since a neuron reflects the decision making of a single agent, a neural network with hundreds of neurons can be seen as the interaction of many economic decisions. In other words, the combination of "the buy / sell decisions of many agents results in a market model" [Zim89, p. 497].

In this approach of agent-based modeling, we discuss two different types of market price formation: The 'explicit price dynamics' considers the impact of the microeconomic market structure (agents' behavior) on the macroeconomic dynamics (price formation). In this case, the price change is a reaction to the observed market excess (i. e. demand – supply) [LeB01a, p. 256]. The 'implicit price dynamics' presumes, that the market directly influences the decision making behavior on the microeconomic level. This means, that the price is a control parameter of the market to avoid situations of market dis-equilibria [Weid00, ZNG01d, p. 200 and p. 739-41]. Both market price mechanism are integrated into the feedforward neural network framework.

As a specialty, our considerations are *not* limited to the analysis of a single market. We extend our investigations to a multiple market approach, which allows to treat several markets simultaneously. Thus, an agent has to handle a portfolio of assets and has to find the most profitable allocation of funds among them [ZNG01d, p. 736-7].

In an empirical study, we apply our multi-agent approach to real-world FX-markets (USD, DEM and YEN). In a multiple market analysis it turns out, that our model is superior to more conventional forecasting techniques.

Chapter 8: Multi-Agent FX-Market Modeling based on Cognitive Systems

Chapter 8 describes a multi-agent approach on the basis of time-delay recurrent neural networks. The recurrent networks are utilized to model the cognitive systems of the agents. This chapter builds upon (*i.*) the time-delay recurrent neural networks developed in chapter 4 and (*ii.*) the learning algorithms discussed in chapter 5.

In this multi-agent model, the decision-taking process of each agent is based on an elementary cognitive system with three basic features: perception, internal processing and action [RN95, LN77, Per01, Bar97, p. 31-48, 508-19, p. 587-619, p. 391-412 and p. 393-419]. These features are necessary conditions of a cognitive system [Cra43, RSMH86, RN95, p. 57, p. 40-4 and p. 13-4]. Let us briefly examine the properties of the cognitive system.

We distinguish conscious from unconscious perception [Bar97, RN95, Per01, p. 411-9, p. 32-3 and p. 393-5]. During conscious perception the cognitive system compares selective observations to certain internal expectations. Only the resulting difference has an impact on the system. Unconscious perception enters the cognitive system directly. This corresponds to stimulus-response. The internal processing mainly consists of an internal model of the external world [RN95, Bar97, LN77, p. 203, p. 404-6 and p. 589-95]. The internal model balances conscious perception with internal expectations. To generate internal expectations, an internal memory is required [LN77, p. 303-8]. This memory can be seen as a merging of all relevant past information from which the internal expectation is formed. Actions are initiated by evaluating the internal expectations with an objective function (e. g. utility maximization). Hence, actions are always goal-oriented [RN95, p. 31 and p. 41-45].

We will discuss two modeling approaches: First, we give an *inductive* description of perception, internal processing and action [ZNG01b, p. 768-70]. As a structural representation of the inductive approach to cognitive systems, we propose error correction neural networks [ZNG01b, p. 767-68]. Second, we derive the three features of the cognitive system *deductively* from the assumption of homeostasis [ZGTN02]. Given a changing environment, homeostasis can be seen as the attempt of a cognitive system to maintain an internal equilibrium [CMcV01, RN95, p. 11-7 and p. 35]. We model the deductive approach to cognitive systems by zero-neurons within a time-delay recurrent neural network. Zero-neurons are input-output neurons with constant, task invariant target values and fixed input connections [ZGTN02]. We provide experimental results with real-world financial data for both modeling approaches.

Chapter 9: Summary and Conclusion

We conclude this work by summarizing the major results and findings of the different chapters. Furthermore, we emphasize the major contributions of this work to the research fields of multi-agent modeling and econometrics. In an outlook, we point to directions of future research.

Chapter 2

NEURAL NETWORKS: INTRODUCTION & HISTORICAL NOTES

In this chapter, we deal with a comprehensive foundation of neural networks. We start off with a basic introduction to neural networks (sec. 1.). Thereafter, we describe the well-known 3-layer feedforward neural network (sec. 2.) and summarize recent historical developments in the research field (sec. 3.).

More precisely, section 1. deals with the biological foundation of neural networks. Since our main interest is on the modeling of markets and the decision making of the market participants (agents), we quickly turn away from the biological motivation of neural networks and focus on an uncommon interpretation of neurons as elementary models of decision making [Zim89]. We will point out, that a neural network can be seen as the interaction of many decisions. This corresponds to a market process.

Section 2. introduces the well-known 3-layer feedforward neural network. As we will show, 3-layer feedforward neural networks have several drawbacks. For instance, a complicated preprocessing of the raw inputs is required, temporal structures can only be represented in an indirect way and large 3-layer networks not only learn the underlying dynamics, but also fit the noise in the data (dilemma of overfitting). In this connection, we also deal with possible solutions of these issues. As we will point out, this leads to the incorporation of prior knowledge into the modeling (see chapters 3 and 4) [NZ98, ZN01, p. 375-86 and p. 321-50].

In section 3. we focus on important milestones within the research field of neural networks. Beginning with the classic work of McCulloch and Pitts (1943), we consider Hebb's unsupervised learning rule for neural networks (1949), Rosenblatt's perceptron (1958) and the related concept of the adaptive linear element, which is developed by Widrow and Hoff (1960). Furthermore, we deal with Minsky and Papert's critics of perceptrons, which is often referred to as the credit assignment problem (1969). Although solutions of the credit assignment problem were formulated in the 1970s, Minsky and Papert's criticism caused a downturn in the interest of neural networks. However, the research on neural networks revived in the beginning of the 1980s. For instance, based on the work of Willshaw and von der Malsburg (1973), Teuvo Kohonen developed the theory of self-organizing feature maps (1982). Further on, the research group headed by David Rumelhart popularized the standard error backpropagation algorithm (1986), which was firstly discovered by Paul Werbos (1974). Inspired by the idea of an energy function of a physical dynamical system, John Hopfield (1982) introduced a fixpoint recurrent neural network. The Boltzmann machine (1985) invented by Ackley et al. can be seen as a generalization of Hopfield's idea. In 1988, Broomhead and Lowe suggested radial-basis function networks, while among others, Jeff Elman (1990) developed a time-delay recurrent neural network architecture. Finally, we refocus on the dilemma of overfitting and purely data driven modeling, which arose in the beginning of the 1990s.

1. FROM BIOLOGICAL TO ARTIFICIAL NEURONS

Intending to give a basic introduction to artificial neural networks, let us start off with the biological motivation of this research area. Since the primary interest of our work is confined to the modeling of dynamical systems with applications in economics, we will also point out, that a neuron can be seen as an elementary model of economic decision making [Zim89, p. 496-8 and p. 3-6]. We begin our studies by describing the basic analogy between biological and artificial neural networks. Subsequently, we discuss a constitutive model of an artificial neuron that can be seen as the foundation of our neural network architectures (see chapters 3 and 4).

1.1 BIOLOGICAL NEURONS

Neural networks are an approach of understanding the brain within an algebraic framework. In other words, neural networks are formal mathematical models of brain mechanisms [Cow90, p. 830]. The brain can be seen as a complex, nonlinear and parallel information processing system [Hay94, p. 1]. The structural constituents of the brain are called neurons. These neurons are efficiently organized in networks in order to perform a broad spectrum of computations (e. g. pattern recognition, audio and visual processing or motor control) [Hay94, p. 1]. To give a comprehensive description of neurons, let us embed a single neuron into the complete nervous system of a human being [Hay94, p. 6-7]. Basically, the human nervous system consists of three major stages: receptors, the neural network and effectors. These stages are depicted in Fig. 2.1 [Hay94, p. 6].

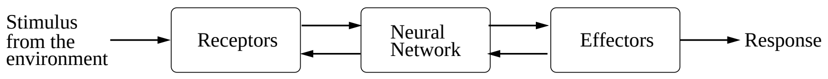


Figure 2.1 Schematical illustration of the human nervous system [Hay94, p. 6]

As shown in Fig. 2.1, the human nervous system receives input information from the *receptors*. The receptors transform stimuli from the environment into electronical impulses [Hay94, p. 6]. This electronical stimulus is transmitted to the *neural network*. The neural network is elementary for the human nervous system: it is the representation of the human brain. The neural network evaluates the incoming input information from the receptors and initiates actions [Hay94, p. 6]. These actions are forwarded in form of electronical impulses to the *effectors*. The effectors transform the electronical impulses of the neural network into perceivable responses resp. system outputs [Hay94, p. 6]. The forward transmission from the receptors to the effectors refers to the flow of information signals through the human nervous system. Besides the forward information flow, there is also a feedback in the system [Hay94, p. 6]. This is indicated in Fig. 2.1 by the arrows pointing from right to left.

Regarding the three-stage scheme of the human nervous system (see Fig. 2.1), a neuron is a single information processing unit, which is elementary for the functioning of the neural network. Neurons occur in different shapes and sizes [CS97, p. 53-5 and p. 400]. Fig. 2.2 illustrates the shape of a common type of cortical neurons, which is the so-called pyramidal cell [CS97, Hay94, p. 56 and p. 7-8].[1]

The pyramidal cell (Fig. 2.2) consists of the cell body (nucleus), the dendrites, and the axon [Hay94, p. 6-9]. The dendrites are the receptive zone of the neuron, which receive input signals from other neurons. The appearance of the dendrites is similar to fine filaments with an irregular surface consisting of so-called dendritic spines. The axon is the transmission line of the neuron. It is characterized by a high electrical resistance and capacitance. Compared to the dendrites, the axon has a greater length and its surface is much smoother. The dendrites conduct electronical impulses towards the nucleus of the neuron, whereas the axon transfers electronical impulses away from the nucleus.

Generally spoken, if the net excitation of the neuron reaches a certain level, the neuron emits an output. The output of the neuron is basically

[1]Fig. 2.2 is taken from *The Electronic Image Bank*, General Psychology, Brown Widdison, McGraw-Hill Higher Education, Nov. 1995, ISBN 0697296474.

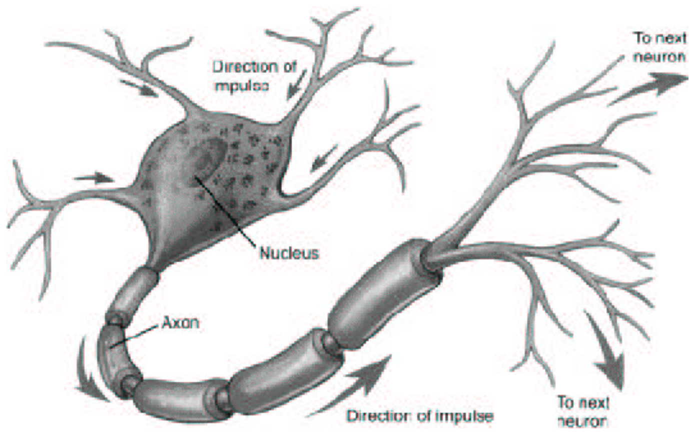


Figure 2.2 Biological neuron: The pyramidal cell

encoded in form of a series of short electronical impulses. These electronical signals are known as action-potentials or 'spike' trains [Hay94, p. 7].

The signal flow between adjacent neurons is organized by so-called synapses. Synapses are connections between the axon of one neuron and the dendrites of another neuron. The most common type of such a connection is the chemical synapse. It is established by the release of a small amount of a chemical substance (so-called neurotransmitter) between two adjacent neurons. For details on synapses see Churchland and Sejnowski (1997) [CS97, p. 55-78].

1.2 ARTIFICIAL NEURONS

Likewise to a biological neuron, an artificial neuron is an elementary information processing unit that is a basic building block of an artificial neural network [CS97, Hay94, p. 53, 101-2 and p. 10]. Now, the question arises, how artificial neurons and artificial neural networks are composed. Fig. 2.3 depicts a model of a nonlinear artificial neuron [Zim94, Hay94, p. 4 and p. 11].

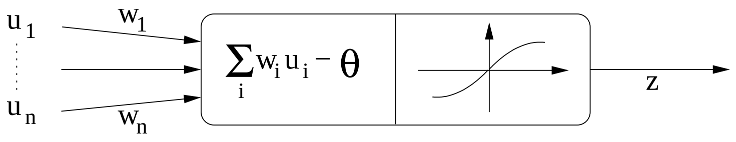


Figure 2.3 A nonlinear artificial neuron

The artificial neuron (Fig. 2.3) is a mathematical model of a biological neuron (Fig. 2.2): Input signals u_i are transferred into the neuron by the connections w_i. Each connection is associated with a weight w_i, which indicates the strength of the input signal u_i. The weights w_i may have negative as well as positive values. The connections w_i represent the synapses of the biological neuron [Hay94, p. 10-1].

Inside the neuron, an adder sums the weighted input signals $w_i u_i$ [Hay94, p. 11]. This linear combiner is the simplest way of constituting an interdependency among the input signals [Zim94, p. 4]. The resulting superposition is the net input of the neuron. The net input of the neuron is lowered by the bias θ [Hay94, p. 11].

If the net input (adjusted by the bias θ) reaches a certain activation level, the neuron emits an output z. The switching behavior is adapted by a so-called activation or squashing function [Zim94, Hay94, p. 4-5 and p. 11-2]. Depending on the activation function, the output of the neuron is either continuous or binary valued [Hay94, p. 12]. The activation function limits the range of the neuron's output signal. In the majority of cases, the finite output range is the unit interval $[0, 1]$ or alternatively $[-1, 1]$ [Hay94, p. 10].

Typically, continuous and differentiable nonlinearities are used as activation functions [ZaRe99, p. 24]. The most common types are sigmoidal functions (e. g. hyperbolic tangent) [Zim94, p. 5-6]. However, also non-continuous and non-differentiable functions may be applied [Bis95, p. 82]. For example, one can refer to the Heaviside step function [Zim94, p. 5].

The usage of a nonlinear activation function is essential to model the switching behavior of the neuron [Zim94, p. 4 and p. 6]. As we will point out later, the switching can be seen as a crossover from a market evaluation to an economic decision [Zim94, p. 3-4 and p. 6].

Let us describe the artificial neuron (Fig. 2.3) in mathematical terms. A neuron k is specified by the following two equations [Hay94, p. 11]:

$$\text{netin}_k = \sum_{i=1}^{n} w_{ki} u_i - \theta_k \tag{2.1}$$

$$z_k = f(\text{netin}_k) \tag{2.2}$$

In Eq. 2.1, the net input (netin_k) of the neuron is computed by adding the weighted input signals $w_{ki} u_i$. Each weight w_{ki} of the neuron is associated with a particular input u_i. The linear combined input signals $w_{ki} u_i$ are lowered by the bias θ_k. The output signal z_k of the neuron is calculated by Eq. 2.2. To compute the output z_k, the net input (Eq. 2.1) is transferred through the activation function $f(\cdot)$ of the neuron.

1.3 COMMON ACTIVATION FUNCTIONS

Among the different activation functions, one typically identifies three common types [Bis95, Smi93, Hay94, Lip87, p. 82, p. 32-5, p. 12-5 and p. 5]: (*i.*) threshold functions (Eq. 2.3), (*ii.*) piecewise linear functions (Eq. 2.4) and (*iii.*) sigmoid functions (Eq. 2.5). Threshold functions are expressed as

$$f(\text{netin}) = \begin{cases} 1 & \text{netin} \geq 0 \\ 0 & \text{netin} < 0 \end{cases} . \tag{2.3}$$

A neuron equipped with a threshold function (Eq. 2.3 [Zim94, Hay94, p. 5 and p. 12]) is often referred to as the McCulloch and Pitts model [MP43]. The output signal z of the neuron is one ($z = 1$) if the net input signal is non-negative. Otherwise, the output signal z is zero ($z = 0$).

An example of a piecewise linear function is

$$f(\text{netin}) = \begin{cases} \dfrac{1}{2} & \text{netin} \geq +\frac{1}{2} \\ \text{netin} & +\frac{1}{2} > \text{netin} > -\frac{1}{2} \\ 0 & \text{netin} \leq -\frac{1}{2} \end{cases} . \tag{2.4}$$

The piecewise linear function in Eq. 2.4 has a linear region in which the output signal z is equal to the net input [Hay94, p. 14]. If the net input of the neuron exceeds a certain activation level, the piecewise linear function reduces to a threshold function.

The group of sigmoid functions is probably the most common type of activation functions [Hay94, p. 14]. Sigmoid functions are defined as strictly increasing functions, which combine linear and nonlinear behavior [Smi93, p. 32]. Typical examples of sigmoid functions are the hyperbolic tangent [Zim94, p. 6] or the logistic function [Smi93, p. 33], which is

$$f(\text{netin}) = \frac{1}{1 + \exp\left(-a(\text{netin})\right)} . \tag{2.5}$$

The parameter a determines the slope of the logistic function. In the limit, as the slope a reaches infinity, the logistic function becomes a threshold function [Hay94, p. 14]. The logistic function is continuous and differentiable [Hay94, p. 14].

The introduced activation functions (Eq. 2.3, 2.4 and 2.5) are depicted in Fig. 2.4, 2.5 and 2.6. In addition, Fig. 5.4 illustrates the shape of the hyperbolic tangent squashing function. Note, that the slope of the logistic function shown in Fig. 2.6 is $a = 4$.

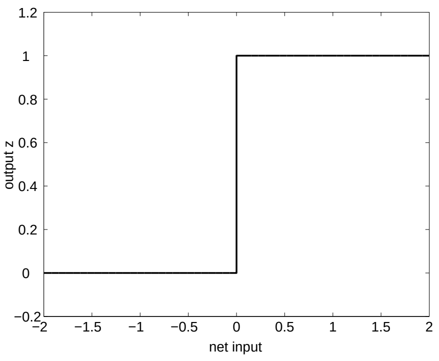


Figure 2.4

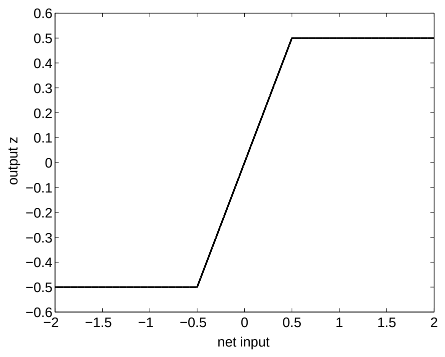


Figure 2.5

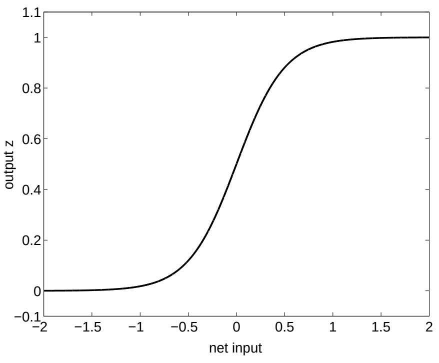


Figure 2.6

Fig. 2.4: Threshold activation function (Eq. 2.3).

Fig. 2.5: Piecewise linear activation function (Eq. 2.4).

Fig. 2.6: Logistic activation function ($a = 4$, Eq. 2.5).

1.4 NEURONS AS ELEMENTARY MODELS OF DECISION MAKING

In contrast to the biological motivation of neural networks (sec. 1.1), we believe, that neural networks have also an economic foundation: On the one hand, neural networks are an ideal framework for the modeling of complex nonlinear dynamical systems [ZN01]. Therefore, it is straightforward to utilize neural networks in the field of econometrics [Zim94, p. 3-4]. On the other hand, neurons can be seen as elementary models of (economic) decision making [Zim89, Zim94, p. 496 and p. 6-10]. Thus, a neural network which may include hundreds of neurons constitutes a market model in which individual traders (neurons) interact. As we will point out, the latter interpretation merges economic theory with the mathematical theory of neural networks. Therefore, the economic interpretation of a neuron is also one of the key concepts of our multi-agent market models (see chapter 7).

To illustrate the analogy between a neuron and an economic decision making process, let us consider a trader dealing at a stock exchange [Zim89, Zim94, p. 496 and p. 3-4, 6-10]. Typically, such a trader has to manage an information overload: Stock market overviews, various technical indicators and other fundamental data are provided by different information sources. Since it is nearly impossible for a human being to handle the latter information flood, the trader has to focus his attention on a couple of news, which seem to be important to him. We call this process of sorting out relevant news for a further evaluation *information filtering*. Information filtering is the first stage of the decision making process.

Having separated out useful information, the trader has to form an opinion about the future development of the stock market. More precisely, the trader has to figure out interdependencies among the selected indicators and has to rate the information by its impact on the stock market. This is usually done by the internal model of the trader, which guides his decision making behavior. In case the trader is a chartist, the market evaluation is performed on the basis of a few technical indicators [Far98, p. 15]. If the trader is a so-called fundamentalist, the decision making is based on fundamental data (e. g. information such as balance sheets, companies income statements, earnings or dividend prospects) [Far98, p. 15]. We refer to this market evaluation as the process of *building a superposition of information.* It is the second stage of the decision making process.

Subsequently, the trader has to come to a decision, which is based on the beforehand market evaluation. Here, the trader has to transform the results of the market evaluation into an action, i. e. he has to decide whether to buy or sell shares. In other words, the market evaluation is transformed into a *yes* or *no* statement. The *initialization of an action* is the third stage of the decision making process.

Now, the question arises, how the outlined decision making process can be modeled in mathematical terms. For this purpose, we suggest to use a single neuron. As we will point out, the neuron precisely reflects the three stages of the decision making process [Zim94, p. 3-4]. Due to visual clarity, the neuron is redisplayed in Fig. 2.7.

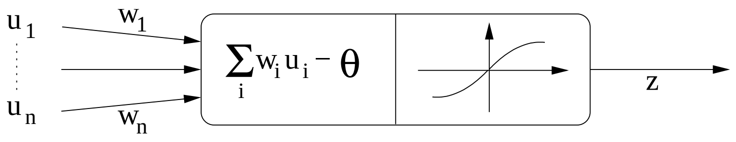


Figure 2.7 A neuron: an elementary model of decision making

Suppose, that the neuron (Fig. 2.7) is supplied with a large number of input signals $u_1, \ldots, u_n$. This can be seen as an information overload, which has to be handled by the neuron. The process of sorting out unneeded information can be adapted by adjusting the weights of the different input signals: While useless information u_i is faded out by setting the corresponding weight w_i to zero ($w_i = 0$), important information u_j can be stressed by adjusting the associated weights w_j to a higher (absolute) level ($|w_j| > 0$). This corresponds to the first stage of the elementary decision making process.

The superposition of information is achieved by the adder of the neuron: The sum of weighted input signals can be seen as the simplest way of building a superposition of information. The weighting of the input signals refers to their importance. Thus, the net input of the neuron can be seen as a quantitative evaluation of the stock market.

The transformation of the quantitative market evaluation into an action is modeled by the activation function and the bias of the neuron: If the net input of the neuron, which is lowered by the bias θ, is e. g. non-negative, the activation function may switch to 1. Otherwise, if the lowered net input is negative, the activation function may return -1. The output z of the neuron, which is either 1 or -1, can be interpreted as the action of the trader. For example, a buying decision corresponds to an output of 1, while a selling decision is analogous to -1. Common examples of nonlinear activation functions creating the required switching behavior are threshold functions or sigmoid functions. In this connection, it is important to note, that only a nonlinear activation function is able to model the crossover from a market evaluation to an economic decision [Zim94, p. 6].

In mind the interpretation of a single neuron as an elementary model of decision making, a neural network can be seen as a model of interacting economic decisions [Zim94, p. 6-10]. In contrast to the biological facets of neural networks, the latter interpretation provides an interface between economics and the mathematical theory of neural networks: If we assume, that a single neuron reflects the decision making scheme of a single trader, a neural network with hundreds of neurons describes a market process. We will refocus on this interpretation of a neural network in chapter 7, since this is one foundation of our multi-agent models.

2. 3-LAYER FEEDFORWARD NETWORKS

Up to now, we merely focused on single neurons, which are elementary building blocks of neural networks. In a neural network, the neurons are basically organized by so-called *layers* or *clusters* [Bis95, Hay94, p. 116-18 and p. 21]. A layer consists of a number of neurons, which share the same characteristics. In general, we have *three* different types of neural network layers [Hay94, p. 21-2]: (*i.*) input layers, (*ii.*) hidden layers and (*iii.*) output layers. An input layer solely gathers the information from outside the neural network. Hidden layers, which typically lie inbetween input and output layers, are the computational units of the neural network. In larger neural networks, several hidden layers may be consecutively linked to each other. The output layers provide the output of the neural network, i. e. the response of the network to a certain activation pattern of input signals.

2.1 ARCHITECTURE OF THE STANDARD MODEL

As a first approach to neural networks, let us examine a 3-layer feedforward neural network [ZaRe99, Wer94, p. 22-23 and p. 272-3]. In a feedforward neural network, the information flow is stringently from the neurons of the input layer to the neurons of subsequent layers. This means that there are no feedback loops or connections within the same layer [Hay94, p. 22-3].[2] The 3-layer feedforward network is depicted in Fig. 2.8 [ZaRe99, Lip87, p. 23 and p. 15-6].

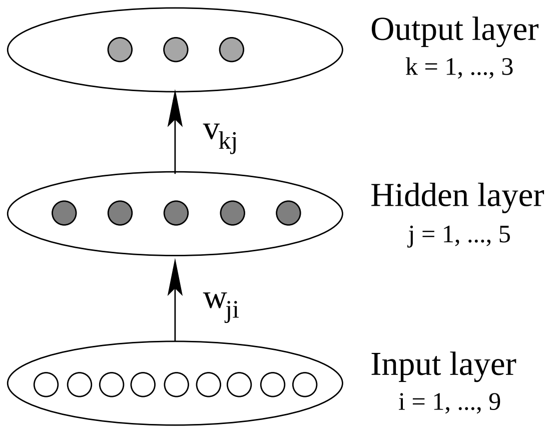


Figure 2.8 A 3-layer feedforward neural network

[2]For further details on recurrent neural networks incorporating feedback loops, the reader is referred to the book edited by Kolen and Kremer (2001) [KK01]. Time-delay recurrent neural networks are introduced in chapter 4.

The 3-layer feedforward neural network (Fig. 2.8) consists of one input layer, one hidden layer and an output layer. The layers of the 3-layer feedforward network are connected as follows [Hay94, p. 22]: First, the input layer is connected to the hidden layer. Second, the hidden layer is connected to the output layer.

The input layer of the network supplies an input vector (often referred to as activation pattern) to the hidden layer. The output of the computational hidden layer is utilized as input for the subsequent output layer. Based on the information flow from the hidden layer, the output layer writes out the overall response of the network to the activation pattern [Hay94, p. 22].

In our example (Fig. 2.8), the input layer consists of *nine* input neurons i ($i = 1, \ldots, 9$), while the hidden layer is formed by *five* hidden units j ($j = 1, \ldots, 5$) and the output layer is constituted by *three* output neurons k ($k = 1, \ldots, 3$). The arrows in Fig. 2.8 indicate the forward oriented connections between the different layers. We assume, that the feedforward neural network is fully connected [Hay94, p. 22 and p. 159]. This means that all neurons of a certain layer are connected with all neurons of the directly sequencing layer. If some of the connections are missing, we speak of a partially connected neural network [Hay94, p. 22]. Since the signal flow of the 3-layer network (Fig. 2.8) is strictly feedforward, there are no feedback loops or connections between neurons within the same layer [Hay94, p. 21-3].

The weights of the 3-layer neural network (Fig. 2.8) are w_{ji} and v_{kj}. A particular weight w_{ji} connects an input neuron i to a hidden neuron j, while a weight v_{kj} connects a hidden neuron j to an output neuron k. The weights w_{ji} and v_{kj} are the turnable parameters of the neural network. Due to visual clarity, we omitted any bias connections in the neural network (Fig. 2.8). However, bias connections to the hidden layer as well as to the output layer can easily be established by assuming a single input neuron, which is directly connected to these layers. The output of the latter input neuron is equal to 1 at any time. [Bis95, Zim94, p. 142 and p. 40]

Let us briefly describe the output y_k, $k = 1, \ldots, 3$, of the 3-layer neural network (Fig. 2.8) in mathematical terms [Bis95, p. 118-9]:

$$y_k = \sum_{j=1}^{5} v_{kj} \tanh\left(\sum_{i}^{9} w_{ji} x_i\right) . \qquad (2.6)$$

For the computation of the network output (Eq. 2.6), we assume that the hidden neurons of the network are equipped with hyperbolic tangent activation functions (Fig. 5.4). The calculation of the network output

(Eq. 2.6) is often referred to as the forward path of the neural network. It is also a major part of the standard error back-propagation algorithm [Zim94, p. 37]. Now let us discuss some remarkable properties of the 3-layer feedforward network (Fig. 2.8).

2.2 UNIVERSAL FUNCTIONAL APPROXIMATION ABILITIES

It is well-known, that a 3-layer feedforward neural network with a sufficiently large hidden layer may in principle model any continuous functional relationship on a compact domain [HSW92, p. 14-20]. With regards to this so-called approximation theorem, each time series identification task is transferred into a pattern recognition approach [NZ98, Hay94, p. 373-4 and p. 67-70]. In doing so, one implicitly assumes that a complete description of the underlying time series dynamics is included in the training data set. Now, if *not* all the information required to describe the dynamics is contained in the training data, a model building, which is solely based on the approximation theorem, is clearly disadvantaged [NZ98, p. 373-4]. To overcome this drawback, we propose neural networks which include additional prior knowledge about the application in form of architectural elements [ZN01, ZNG01a, p. 312-321 and p. 247-261]. Examples of this model building philosophy are given in chapters 3 and 4.

2.3 REPRESENTATION OF TEMPORAL STRUCTURES

Another crucial issue of the 3-layer feedforward neural network is the representation of intertemporal structures [Zim94, ZN01, p. 26 and p. 312, 321-2]: Temporal structures and intertemporal relationships can only be invented by a complicated input preprocessing with preset time lags. Feedforward neural networks only map an input vector to a corresponding output vector [Hay94, p. 66-8]. In other words, a feedforward network is not able to set up a memory (superposition of past information) in order to model the kinetics of a dynamic system directly. Furthermore, temporal structures are *not* reflected by the architecture of the neural network.

As a remedy, chapter 4 introduces time-delay recurrent neural networks. These networks allow the incorporation of memory effects into the modeling. Temporal structures can be directly represented in the network architecture by using unfolding in time and shared weights [ZN01, p. 321-8].

2.4 INPUT PREPROCESSING

A feedforward neural network requires a complicated preprocessing of the raw input information in order to generate an appropriate description of the present time state [Zim94, ZN01, p. 26 and p. 324-5]. This issue is closely related to the problem of representing temporal structures within a feedforward network.

Examples of complicated preprocessing functions are mainly found in financial applications. Here, sophisticated technical indicators are used in the preprocessing of the raw input data. An overview of such preprocessing functions is given in Zimmermann (1994) [Zim94, p. 20-8]. In contrast, we propose to use rather simple preprocessing transformations, which are also able to capture the dynamics of the raw input time series [NZ98, p. 374-5].

The preprocessing of the input data is basically done by using *two* simple transformations: (*i.*) the *momentum* and (*ii.*) the *force* of each raw input time series. If we have a prediction horizon of n time steps, a raw time series u_t is preprocessed as follows [NZ98, p. 374]:

$$\text{momentum:} \quad \tilde{u}_t = \text{scale}\left(\frac{u_t - u_{t-n}}{u_{t-n}}\right) \tag{2.7}$$

$$\text{force:} \quad \hat{u}_t = \text{scale}\left(\frac{u_t - 2u_{t-n} + u_{t-2n}}{u_{t-n}}\right) \tag{2.8}$$

In Eq. 2.7 we compute the relative change of the time series. The relative change (Eq. 2.7) is an indicator for the inertia (or momentum) of the time series. Referring only to the momentum of the time series (Eq. 2.7), leads to poor models which only follow obvious trends [NZ98, p. 374-5]. To avoid this, the forces (Eq. 2.8) are an important additional measure to characterize the turning points of the time series. The forces (Eq. 2.8) can be seen as the normalized curvature of the time series. Note, that both preprocessing functions include an additional scaling of the transformed input data. The scaled input signals have a mean value of zero and a statistical variance of one [PDP00, p. 73]. We found that a scaling of the data fits best to the numerics of hyperbolic tangent squashing functions [ZNG01a, p. 247].

Economic time series (and also many other) are likely to contain inflationary (or linear) trends [NZ98, p. 374-5]. As a consequence, the time series do not satisfy the conditions of (weak) stationarity [PDP00, p. 96]. The stationarity of the data is a common assumption in time series analysis [PDP00, p. 96]. In general terms, "a process is said to be nth order weakly stationary if all its joint moments up to order n exist and are time invariant, i. e., independent of time origin" [Wei90,

p. 8]. Thus, "a second order weakly stationary process will have constant mean and variance, with the covariance and the correlation being functions of the time difference alone" [Wei90, p. 8]. If the time series is not stationary, it can be transformed to stationarity by differencing the data [PDP00, p. 96-7]. Differencing a time series u_t means, that we calculate the differences $u_t - u_{t-1}$. If the time series contains a nonlinear trend, it has to be differenced more than once [PDP00, p. 97]. The stationarity of a time series can be examined with the so-called *Dickey-Fuller* and *Augmented-Dickey-Fuller* test [PDP00, p. 97].[3] Note, that the momentum (Eq. 2.7) incorporates a one step differencing of the raw data, while the force (Eq. 2.8) contains a two step differencing. Thus, both preprocessing functions can be used to transform a non-stationary time series to stationarity [NZ98, p. 374]. Concerning financial data, one difference is usually sufficient to ensure stationarity [PDP00, p. 97].

2.5 THE DILEMMA OF OVERFITTING

The functional approximation abilities of neural networks impose another well-known problem, which is called *overfitting* [Zim94, Bis95, p. 58-60 and p. 11]. Overfitting is the loss of the network's generalization performance due to learning by heart. Let us exemplify the problem of overfitting [Zim94, p. 58-60]: Especially in economics, highly nonlinear structures have to be identified out of small data sets. The data is often covered with noise or contains outliers. The modeling of such a dynamic system requires a complex neural network incorporating a relatively high amount of free parameters. The required complexity leads to an overfitting of the underlying time series. The neural network does not only fit the underlying structure of the time series, but also the noise included in the data. Overfitting is dangerous because the network loses its generalization abilities. For example, the neural network generates predictions that are far beyond the range of the training data.

There are several approaches to prevent overfitting. First, overfitting can be slowed down or even avoided by using large training data sets [Zim94, p. 58-60]. A general rule is to have at least 10 times as many training examples as there are free network parameters. Unfortunately, large training data sets are often not available in economic applications. Furthermore, a large data set is no guarantee for preventing overfitting.

Since large training data sets are often not available, we have to look for other techniques that prevent overfitting. The problem of overfitting is typically approached by a specific stopping criterion for the learning

[3]For details on these stationarity tests see Poddig et al. (2000) [PDP00, p. 352-61].

(e. g. early or late stopping) together with techniques for the optimization of the network structure (e. g. a penalty terms, weight or node pruning methods) [NZ98, Zim94, p. 405-17 and p. 60-77]. An additional remedy is to deal with time-delay recurrent neural networks. As we will point out in chapter 4, time-delay recurrent networks only use a moderate amount of free parameters. Hence, overfitting is not a serious problem of the modeling [ZNG01a, ZN01, p. 241-49 and p. 323-8]. In the following we discuss the concepts of early and late stopping. Furthermore, we point to related techniques for the regularization of the network structure.

2.6 EARLY STOPPING

A well-known technique for solving the overfitting problem is the early stopping procedure. The early stopping concept is as follows [Zim94, NZ98, Bis95, p. 60-1, p. 410-12 and p. 343-5]: Typically, we randomly initialize the neural network with small weights. In the simplest approach of early stopping, we observe the network error on a validation set to determine the beginning of overfitting. At the beginning of the learning, the training and the validation error usually decrease, because the network learns valuable structures out of the data. If the validation error tends to increase, the training of the neural network is stopped.

A more sophisticated early stopping procedure is to train the neural network until overfitting occurs. Afterwards, a part of the neural network structure (e. g. 10% of the weights) is eliminated by using standard weight pruning techniques.[4] Then, the pruned neural network is reinitialized and the training continues. The outlined sequence of learning and pruning is iterated until a stable neural network structure with no overfitting is achieved [NZ98, p. 410-2]. This variant of early stopping is illustrated in Fig. 2.9 [NZ98, p. 411].

A major drawback of early stopping is that the stopping point may appear after a few learning epochs. In this case, the resulting neural network is more or less restricted to the class of semi-linear models, because the weights remain small and thus the internal signal flow of the network is still in the linear range of the hyperbolic tangent activation functions. In other words, the learning has no chance to develop complex nonlinear structures in the neural network [NZ98, p. 410-1]. In Neuneier and Zimmermann (1998) [NZ98] it is reported, that the early stopping

[4]Common weight pruning methods for the regularization of the neural network structure are introduced in chapter 5. The reader is also referred to Haykin (1994) [Hay94, p. 218-26] and Neuneier, Zimmermann (1998) [NZ98, p. 405-18].

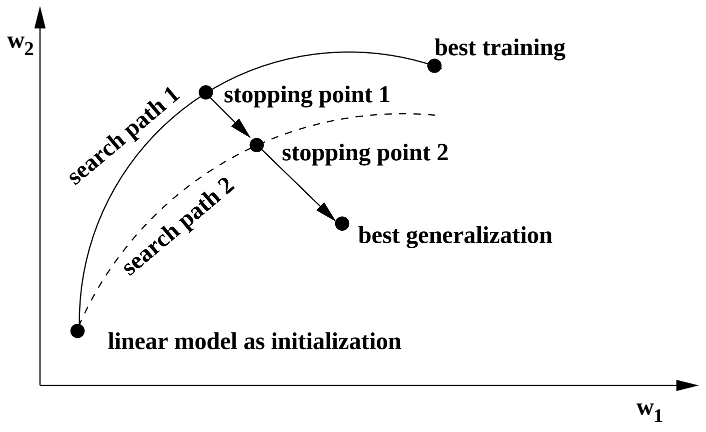


Figure 2.9 The early stopping concept: First of all, the network weights are randomly initialized with small values. Thereafter, the network is trained until overfitting occurs (see 'stopping point 1'). At this point, the neural network structure is optimized by e. g. weight pruning. The learning continues with the reinitialized neural network until overfitting reemerges (see 'stopping point 2'). This variant of early stopping is carried out until a stable neural network without overfitting is obtained (see 'best generalization') [NZ98, p. 411].

concept will generate models with good generalization abilities. However, according to their experiments, the authors point out that early stopping also fails to do so in many cases [NZ98, p. 410-1].

2.7 LATE STOPPING

The concept of late stopping tries to solve the overfitting problem in a different way [NZ98, p. 410-12]: At the beginning of late stopping, the network is randomly initialized with small weights. Afterwards, the neural network is trained until a minimal training error is achieved. At this point, the network most likely shows typical overfitting behavior. To re-increase the generalization performance, the structure of the neural network is optimized by e. g. weight pruning methods (see chp. 5). Likewise to the procedure in early stopping (Fig. 2.9), a pruning step should remove 10% of the network parameters. Thereafter, the pruned neural network is trained until the error level converges again. The late stopping concept should end up with a stable neural network that does not show overfitting behavior. The procedural steps of late stopping are illustrated in Fig. 2.10 [NZ98, p. 412].

As an advantage of late stopping, the learning can explore a maximum amount of nonlinearity out of the training data. Afterwards, the

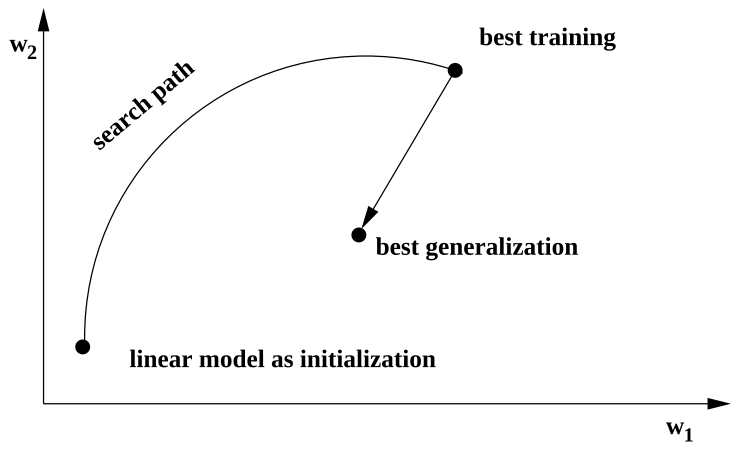


Figure 2.10 The late stopping concept: Starting off with a random initialization of the neural network within its linear range, we train the neural network until convergence. Afterwards, the network structure is optimized by using weight pruning methods and the training continues. The late stopping procedure is iterated until a stable model without overfitting behavior is obtained [NZ98, p. 412].

complexity of the neural network is reduced by e. g. weight pruning in order to establish a model with a good generalization performance (Fig. 2.10). Interestingly, the two parts of late stopping, "learning to the minimal training error and extracting an appropriate generalizing model [by weight pruning], can be understood as a generation of a structural hypothesis followed by a falsification of the generated structure" [NZ98, p. 412]. Further details on the learning of neural networks and the related falsification of the generated structural hypothesis can be found in chapter 5. A more elaborated training procedure for neural networks is developed in Neuneier and Zimmermann (1998) [NZ98, p. 418].

Finally it should be noted, that the stopping point of the learning can be delayed by (*i.*) noise on the input data or (*ii.*) by a penalty term [Zim94, Bis95, p. 62-4 and p. 338-43, 346-9]. By adding noise to the inputs, one generates an infinite training data set: In each learning epoch through the training data, each training pattern is manipulated by a varying noise term. Thus, the neural network cannot fit each point of the training sample by heart, but is forced to extract general structures out of the training data. Noise on the inputs works best, if the distribution of the noise term corresponds to the distribution of the different input signals. Instead of using an equally distributed noise term, one should refer to a normal distribution. Furthermore, the noise term should not exceed 10% of the input signal variance.

Penalty terms can also be used for the regularization of network structure [Bis95, Smi93, p. 338-43 and p. 192-5]. For example, the weight decay method enhances the overall error function of the network by an additional penalty term, which enforces small weights [Smi93, p. 193-4]. Likewise to early stopping, this regularization technique has a bias towards linear models [NZ98, p. 410-1]. Other penalty terms declare additional constraints on the model building [NZ98, p. 411]. A more detailed overview is given in Zimmermann (1994) [Zim94, p. 62-4].

3. HISTORICAL NOTES ON NEURAL NETWORKS

Let us proceed with some historical notes concerning recent developments in the application field of neural networks. Due to the broad spectrum of developments in the theory of neural networks, we focus our attention to subjectively important milestones of the research field. These milestone are

- the McCulloch and Pitts model of a neuron (1943),
- the Hebbian learning rule (1949),
- the perceptron of Rosenblatt (1958),
- the adaptive linear element by Widrow and Hoff (1960),
- Minsky and Papert's critics of perceptrons (1969),
- Kohonen's self-organizing maps (1982),
- Hopfield networks (1982),
- the Boltzmann machine (1985),
- the standard error backpropagation algorithm (1974 and 1986),
- radial basis function networks (1988),
- Elman's recurrent neural networks (1990) and
- the dilemma of overfitting and purely data driven modeling (1990s).

More comprehensive surveys of the topic, which also include some interesting anecdotes and reminiscences, are given in e. g. Haykin (1994) [Hay94, p. 38-44] or Cowan (1990) [Cow90]. Some other good introductory material is provided in Churchland and Sejnowski (1997) [CS97, p. 1-21].

3.1 MCCULLOCH AND PITTS MODEL OF A NEURON (1943)

In 1943 Warren McCulloch and his colleague Walter Pitts introduced a formal mathematical treatment of brain mechanisms [MP43]. Their artificial neural networks unite studies of neurophysiology and logical calculus [Hay94, p. 38]. McCulloch and Pitts used neural networks to model logical operators (e. g. OR, NOT or AND expressions) by arithmetical quantifiers. The models of McCulloch and Pitts "permit the framing of sharp hypotheses about the nature of brain mechanisms, in a form equivalent to computer programs" [Cow90, p. 830].

More generally, one of the main results of McCulloch and Pitts is, that any well defined input to output relationship (resp. a computable function) can be modeled by a formal neural network, which consists of a sufficient number of neurons and synaptic connections [Hay94, p. 38].

The McCulloch-Pitts model of a neuron is depicted in Fig. 2.11 [Smi93, p. 3]. The neuron emits an output z if the sum of certain excitatory input signals u_i exceeds the threshold θ [Bis95, Hay94, p. 83-4 and p. 10-4]. The neuron incorporates a simple threshold activation function (see Eq. 2.3 and Fig. 2.4). Consequently, the output of the McCulloch-Pitts neuron is either $z = 0$ or $z = 1$. This is often referred to as the all-or-none property of the McCulloch-Pitts model [Hay94, p. 14]

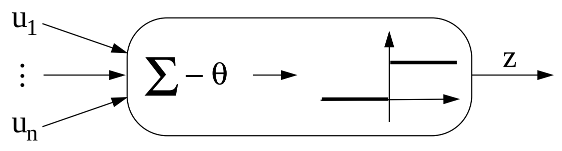


Figure 2.11 Model of a neuron suggested by McCulloch and Pitts [Hay94, p. 13]

The ideas of McCulloch and Pitts have to be seen in parallel to the work in the field of computer science, i. e. the design of digital computers [Smi93, p. 4]. For instance, John von Neumann proposed switch-delay circuit elements, which have several correspondences to the artificial neurons suggested by McCulloch and Pitts [Hay94, Neu86, p. 38].

3.2 HEBBIAN LEARNING RULE (1949)

One of the most favorable characteristics of neural networks is their ability to learn from training data in order to perform specific tasks. On this topic Donald Hebb firstly outlined a physiological learning rule for synaptic modifications [Heb49]. The proposed learning rule can be seen as the basis for further developments in the field of learning and adaptive behavior [Cow90]. Hebb pointed out, that modifications of the brain connectivity continually take place while an individual learns miscellaneous tasks. In addition, changes in the connectivity of the brain during the learning cause neural assemblies [Hay94]. Hebb's postulate of learning can be formulated as a two-part rule [Hay94, p. 55]:

1. If two neurons on either side of a synapse (connection) are activated simultaneously (i. e. synchronously), then the strength of that synapse is selectively increased.
2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

Translated into the language of artificial neural networks, the weight of an input signal should be increased (decreased) in order to accentuate the coherency (incoherency) between that input and the output of the neuron [Bis95, p. 319]. For further details on the Hebbian learning rule and its formal mathematical formulation, see Haykin (1994) [Hay94, p. 55-8] or Churchland, Sejnowski (1997) [CS97, p. 325-30].

3.3 PERCEPTRON OF ROSENBLATT (1958)

Almost nine years after Hebb, the next milestone in the history of neural networks is probably the introduction of the perceptron by Frank Rosenblatt in 1958 [Ros58]. The key to Rosenblatt's work is the question how the brain groups similar external stimuli and differentiates them from dissimilar stimuli [Smi93, p. 4-5]. For this purpose, the perceptron is originally designed as a (linear) pattern classifier. In other words, "[t]he goal of the perceptron is to correctly classify the set of externally applied stimuli [...] into two classes" [Hay94, p. 136]. The classes are separated by a linear decision hyperplane [Hay94, p. 135-6].

Basically, the perceptron incorporates a single non-linear McCulloch-Pitts neuron [Hay94, Bis95, p. 135-7 and p. 98-100]: Input signals u_i, $i = 1, \dots, N$, are weighted by associated parameters w_i. An adder sums the weighted input signals. The resulting sum is compared with a threshold θ and transferred through a threshold activation function (see Eq. 2.3 and Fig. 2.4). As a consequence of the threshold function, the perceptron outputs 1 (-1) in case the net input is positive (negative).

From this it follows, that the perceptron can be seen as a simple two-class pattern-classifier [Hay94, p. 135-6]: For each pattern of input signals u_i, $i = 1, \dots, N$, the output of the perceptron is either $z = 1$ or $z = -1$. If the output of the perceptron is $z = 1$, the related input pattern belongs to class A. On the other hand, if the output signal z is equal to $z = -1$, the corresponding pattern is assigned to class B. As an example, Fig. 2.12 depicts the decision plane of a simple perceptron in case of a two-dimensional, two-class pattern-classification task [Hay94, p. 137-8].

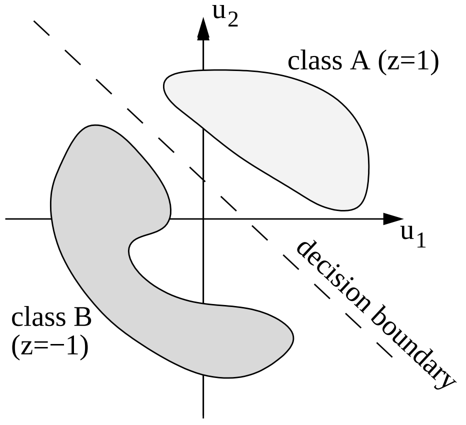


Figure 2.12 The perceptron acts as a linear pattern classifier of two classes, A and B: The decision boundary of the perceptron is a straight line, which is given by $w_1u_1 + w_1u_2 - \theta = 0$. The two classes A and B are separated by the decision boundary. An input pattern (u_1, u_2) induces either an output of $z = 1$ or $z = -1$. If the output z is $z = 1$ $(z = -1)$, the related input pattern (u_1, u_2) is assigned to class A (B). Input patterns (u_1, u_2) of class A (B) are located above (below) the decision boundary. Note, that the bias θ merely shifts the decision boundary away from the origin.

Rosenblatt pointed out, that the perceptron is able to generalize. If a previously unknown pattern resembles to a known one, the output of the perceptron is also similar in both cases [Smi93, p. 4-5].

Remarkably, Rosenblatt also invented a simple iterative learning procedure for the training of the perceptron and proofed the convergence of the learning algorithm. The iterative learning procedure uses an error correction rule (resp. an approach of supervised learning) and is often referred to as the *perceptron convergence algorithm*. For details on the perceptron convergence algorithm, see Haykin (1994) [Hay94, p. 137-43] and Bishop (1995) [Bis95, p. 100-3].

It is important to note, that the simple perceptron, which merely consists of a single neuron, can only distinguish between two classes [Hay94, p. 138]. However, a perceptron incorporating more than one neuron is in a position to separate multiple classes. Nevertheless, the classes have to be linearly separable in order to ensure that the perceptron is able to perform a correct classification [Bis95, p. 103-5]. In other words, nonlinearly separable classes are "beyond the computing capability of the perceptron" [Hay94, p. 138]. This critical issue is one reason for a great distrust against neural networks in the 1960s [Bis95, p. 103-5].

3.4 ADAPTIVE LINEAR ELEMENT (1960)

In 1960, Widrow and Hoff formulated the concept of the adaptive linear element (adaline), which is closely related to Rosenblatt's perceptron [WidH60]. The only difference between the perceptron and the adaline lies in the application of different supervised learning procedures [Hay94, p. 40]: Adaline utilizes a certain kind of gradient decent (so-called least mean square algorithm (LMS)), whereas the perceptron incorporates an iterative non-gradient based training procedure [Hay94, p. 128-33 and p. 137-43]. Remarkably, the work of Widrow and Hoff pioneered the development of adaptive filter theory [Cow90, Hay94, p. 838 and p. 117-8].

3.5 MINSKY AND PAPERT'S CRITICS OF PERCEPTRONS (1969)

The development of the perceptron caused a great enthusiasm in the 1960s [Hay94, p. 40 and p. 148-50]. However, the enthusiasm for neural networks was abruptly quenched by Marvin Minsky and Saymour Papert. In their book entitled "Perceptrons" Minsky and Papert demonstrated on a formal mathematical basis, that the performance of the perceptron, i. e. its computational capability, is fundamentally limited [MiPa69]. More precisely, Minsky and Papert pointed out that a simple perceptron cannot solve classification problems, which are *not* linearly separable [Hay94, p. 138].

Even more important, Minsky and Papert presumed that it is impossible to train the hidden layer in a multi-layer perceptron [Cow90, p. 838]. This is well-known as the *credit assignment problem* [Hay94, p. 40 and p. 62, 148-50]. In other words, Minsky and Papert stressed that the contribution (credit) of a single hidden neuron to the explanation of the overall network error cannot be determined (assigned) [Cow90, p. 838].

Although solutions of the credit assignment problem were formulated until the mid-seventies,[5] the critique of Minsky and Papert led to a noticeable downturn in the research field of neural networks. According to Cowan (1990) [Cow90, p. 840], there are two other major reasons

[5]For instance, the credit assignment problem was addressed by the standard error backpropagation algorithm, which was firstly introduced by Paul Werbos in 1974 [Wer74]. Unfortunately, the backpropagation algorithm received only insufficient consideration and it took until 1986, when D. Rumelhart, G. Hinton and R. Williams reinvented the ideas of Werbos [RHW86, p. 322-8].

for the downturn in the area of neural networks: First, the absence of a technological infrastructure, i. e. powerful computers and work stations, made it difficult to perform experiments. Second, there are psychological and financial reasons. Minsky and Papert's critique discouraged a lot of researchers and in conjunction, the funding of such research activities dried up.

However, even during the 1970s a lot of important developments emerged in the henceforth unfashionable research field of neural networks. For instance, expert systems dominated the research and development in symbolic information processing. In the 1980s, the interest in neural networks revived.

3.6 SELF-ORGANIZING MAPS (1982)

One of the most remarkable theories, that emerged during the 1980s, is that of self-organizing maps [Hay94, p. 41]. Early work in this area was done in Germany by Christoph von der Malsburg in 1973, who performed computer simulations of self-organization [Smi93, p. 9]. Inspired by biophysical studies of so-called brain maps, von der Malsburg and his colleague Willshaw published a paper on the formation of self-organizing maps in 1976 [Hay94]. Brain maps develop a certain internal representation of external stimuli in a spatial manner. Examples of such maps can be found in the visual, auditory or somatosensoric cortex [Smi93, Lip87, p. 9].

Related to the studies of Willshaw and von der Malsburg, Teuvo Kohonen worked out a theory of self-organizing (feature) maps in 1982 [Koh88, p. 119-55]. Kohonen's self-organizing maps (SOMs) are probably one of the most popular types of artificial neural networks in the category of unsupervised learning. For instance, SOMs are used in data mining applications, financial analysis, image processing and visualization. Generally spoken, SOMs generate an ordered low-dimensional representation of a high-dimensional input data space [Bis95, p. 188-9]. The functioning of a SOM is illustrated in Fig. 2.13 [Hay94, Koh88, p. 445 and p. 130-7].

Self-organizing maps use an unsupervised (or self-organized) learning algorithm [Hay94, p. 64-6 and p. 446-8]. In unsupervised learning algorithms, the neural network is *not* provided with target values. Instead, the learning process is guided by a task-independent measure of the network's representation quality. Furthermore, competitive learning rules, e. g. the winner-takes-all-principle, are frequently used in unsupervised learning algorithms. In contrast, supervised learning techniques typically measure the difference between the model output and correspond-

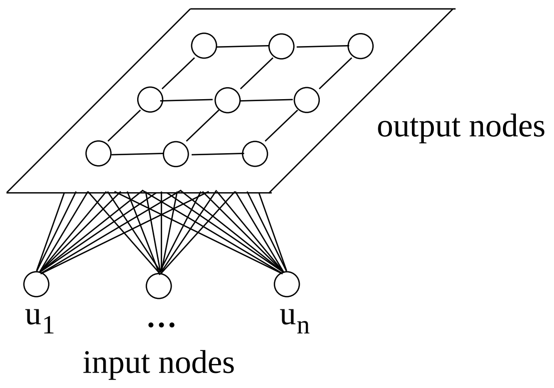


Figure 2.13 Structure and functioning of a self-organizing feature map [Hay94, Koh88, p. 446-54 and p. 122-37]: Typically, a SOM consists of an input layer and an output layer. Each output neuron can be seen as a prototype of a class of similar inputs. The output neurons are usually organized in form of a two-dimensional grid, i. e. they form a feature map. In order to set up the two-dimensional grid, each output neuron is interconnected with a subset of adjacent output neurons. The local connections among the output neurons are fixed, i. e. they are only used to determine the spatial architecture of the SOM. The input neurons are fully connected to the grid of output neurons. The weights of these connections are the only turnable parameters of the SOM. During the training, different input vectors are sequentially presented to the SOM. By a certain similarity measure (e. g. Euclidean distance), the best match between each input vector and the weight vectors of the output neurons is identified. Having detected the most similar output neuron in the feature map, the weights of the winning output neuron are adjusted towards the concerned input vector. As a specialty, not only the weights of the winning output neuron, but also the weights of its neighboring output neurons are modified. Typically, the neighborhood of each output neuron is determined by a Gaussian neighborship function. At the beginning of the learning, the neighborhoods of the output neurons are fairly large, whereas at the end of the training, the neighborhoods are small. This indicates, that a topological ordering of the feature map has taken place during the learning.

ing target values by the application of specific cost functions [Hay94, p. 63-4]. Weight modifications are performed to fit the network output to the given target values (see chp. 5).

3.7 HOPFIELD NETWORKS (1982)

In 1982, John Hopfield utilized the idea of a Lyapunov energy function to derive a novel approach to the understanding of recurrent neural networks with symmetric connectors [Hay94, Hop92, p. 41]. If a Hopfield network is started in a random state, it moves on to a final stable state. This corresponds to the physical behavior of a dynamic system falling into a state of minimal energy. The Hopfield network is illustrated in Fig. 2.14 [Hay94, p. 681].

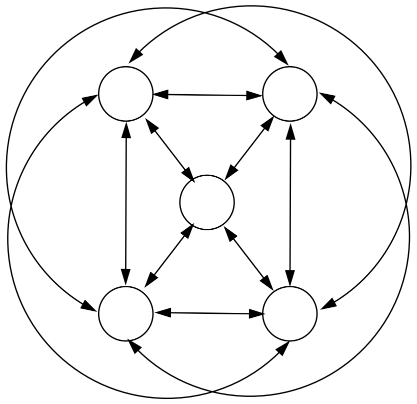


Figure 2.14 A Hopfield network consists of input-output neurons, which are fully connected to each other by multiple feedback loops. The output of a single neuron is fed back to all other neurons of the network. The Hopfield network incorporates symmetric connections without self-connectivity, i. e. $w_{ij} = w_{ji}$ with $w_{ii} = 0$. Each neuron of the network is equipped with a nonlinear squashing function. The Hopfield network employs Hebbian learning and may either operate in a continuous or discrete mode. For an explicit treatment of the Hopfield model see e. g. Churchland, Sejnowski (1997) [CS97, p. 111-9] and Haykin (1994) [Hay94, p. 680-90].

Remarkably, Hopfield firstly described the way in which information is stored in a recurrent neural network in detail. The work of Hopfield not only revived the interest in recurrent neural networks, but also managed to involve the community of physics into the research field [CS97, p. 111-4]. – The Hopfield model establishes "the isomorphism between [...] a recurrent network and an Ising model used in statistical physics" [Hay94, p. 41].

3.8 BOLTZMANN MACHINE (1985)

The stochastical Boltzmann machine was invented in 1985 by Ackley, Hinton and Sejnowski [AHS85] and can be seen as the *first* successful approach to the learning of hidden neurons [Hay94, p. 42]. It is named after the physicist Ludwig Boltzmann, because of the formal equivalence between Boltzmann's work on thermodynamics and the dynamics within the neural network. More precisely, the development of the Boltzmann machine was inspired by a new optimization algorithm called simulated annealing, which was invented by Kirkpatrick, Gelatt and Vecchi in 1983

[KGV83]. Likewise, simulated annealing has its origins in statistical mechanics.

A Boltzmann machine constitutes a stochastic model of neural networks with feedback [Zim89, p. 492]. It can be seen as a generalization of a Hopfield network [Zim89, Hay94, CS97, p. 492-6, p. 562-9 and p. 119-20]: The Boltzmann machine consists not only of input and output units, but also of hidden units. Similar to a Hopfield network, the neurons of the Boltzmann machine are fully interconnected with each other. An example of the Boltzmann machine is depicted in Fig. 2.15 [Hay94, p. 563].

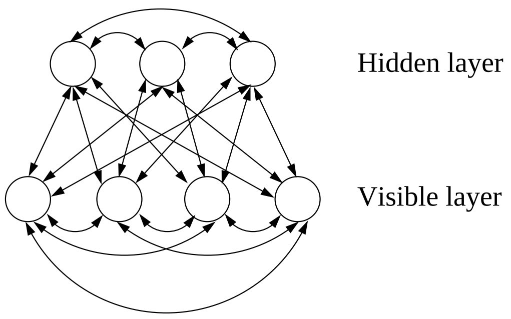


Figure 2.15 The Boltzmann machine is a stochastical neural network with full recurrent connections. It learns a probabilistic model for the data patterns, i. e. the resulting dynamics is stochastic rather than deterministic. The input and output units of the Boltzmann machine are located in the visible layer. The hidden layer of the Boltzmann machine contains the hidden units. These neurons allow us to capture higher order statistics from the data without generating a higher order energy function.

On a probabilistic basis, the output units of the Boltzmann machine either output 1 or 0. This means, that the activation level of a unit is *not* determined by a deterministic activation function but by a probability function. Inspired by thermodynamics and simulated annealing, the Boltzmann machine is said to have a temperature. The temperature of the Boltzmann machine is related to the probability [Zim89, CS97, p. 494-5 and p. 119-20]: A high temperature implies that the probability, that an output unit is in one or another state, is by 0.5 uniformly distributed. During the operation of the Boltzmann machine, the temperature decreases and thus, the behavior of the output units becomes more and more like simple (deterministic) perceptrons. If the temperature is high, the Boltzmann machine randomly bounces between the local energy minima. The high temperature enables the system to explore the search space. If the system is slowly annealed, it is more likely, that it will move to an energy minimum without being able to escape. A thermal equilibrium is reached, if the temperature of the Boltzmann machine remains constant. The annealing resp. temperature reduction procedure corresponds to simulated annealing [KGV83].

Remarkably, the learning algorithm of the Boltzmann machine "broke the psychological logjam by showing that the speculation in Minsky and Papert (1969) was incorrectly founded" [Hay94, p. 42]. Unfortunately, the learning algorithm is computationally more expensive than standard error backpropagation. A detailed description of the Boltzmann machine and the related learning algorithm can be found in Haykin (1994) [Hay94, p. 562-9].

Interestingly, the inner structure of the Boltzmann machine and its dynamical behavior have a direct economic interpretation: "Through the feedback structure, the network can be interpreted as an interaction model of communicating agents [(neurons)] in a generalized form of equilibrium. Even simulated annealing has economic correspondences and consequences. It is this sort of dynamics, that allows the system to find a global and not only a local optimal behavior" [Zim89, p. 505].

3.9 STANDARD ERROR BACKPROPAGATION (1974 & 1986)

In 1986, David Rumelhart, Geoffrey Hinton and Ronald Williams reinvented the error backpropagation algorithm for the training of neural networks [RHW86]. Error backpropagation was originally introduced by Paul Werbos in his PhD thesis at Harvard University in 1974 [Wer74, Wer94]. The backpropagation algorithm solved the credit assignment problem and thus, truly revived the interest in the theory of neural networks [Hay94, p. 156-7].

The two-volume book entitled "Parallel Distributed Processing: Explorations in the Microstructure of Cognition" of the group of researchers headed by David Rumelhart was probably one of the most influential publications within the field of neural networks in the 1980s [Rum86].

For instance, Rumelhart et. al. also outlined a concept for the training of time-delay recurrent neural networks [RHW86, p. 354-7]. The resulting backpropagation through time algorithm is based on the idea, that every recurrent neural network can be represented by an identically behaving feedforward neural network [RHW86, p. 354-5]. The system identification task of the recurrent neural network is transformed into a spatial architecture using unfolding in time and shared weights (see chapters 4 and 5).

3.10 RADIAL BASIS FUNCTION NETWORKS (1988)

In 1988, Broomhead and Lowe introduced a novel class of feedforward neural networks called radial-basis function (RBF) networks [BrLo88, p. 321-55]. Pioneering work in this research area was also done by John Moody and Christian Darken, who developed a heuristic method for the estimation of the basis function parameters using least squares estimators [MoD89, p. 281-94]. Another noteworthy contribution to RBF networks is provided by Tomaso Poggio and Federico Girosi [PG90, p. 1481-97], who applied regularization theory to RBF networks. By this, the smoothness properties of the mapping can be controlled such that the generalization performance of the RBF network is improved [Bis95, Hay94, p. 171-5 and p. 277-8]. Early work on this subject is reviewed in Powell (1985) [Pow85, p. 143-67].

RBF networks can be used for a broad spectrum of applications, e. g. function approximation, regularization, density estimation or pattern classification [Bis95, p. 164]. Fig. 2.16 depicts a basic model of an RBF network [Hay94, Bis95, p. 278 and p. 169].[6]

Likewise to multi-layer perceptrons, RBF networks are universal function approximators [Bis95, Hay94, PS93, p. 182-3, p. 290-3 and p. 305-316]. RBF networks generate *local* approximations to nonlinear input-output mapping, whereas multi-layer perceptrons form *global* approximations to nonlinear input-output mapping [Hay94, p. 293]. In other words, multi-layer perceptrons have *global* segmentation properties, whereas RBF networks have *local* segmentation properties [Fla97, NZ98, p. 145-8 and p. 378]. Detailed comparisons between RBF networks and multi-layer perceptrons are given in Haykin (1994) [Hay94, p. 293-4] and Bishop (1995) [Bis95, p. 182-3].

3.11 ELMAN'S RECURRENT NEURAL NETWORKS (1990)

In 1990, Jeff Elman introduced a recurrent neural network, which uses so-called context units [Elm90]. Context units supply the network with information about previous activation values. In other words, the information of delayed (previous) activation values is given to the network in order to compute current activation values [CK01, p. 21-2]. Therefore, the usage of context units can be seen as a clarification of time-delay recurrent connections [CK01, p. 22]: First of all, an activation value is fed

[6]Note, that more general RBF architectures with more than one hidden layer are usually not considered [Bis95, p. 168].

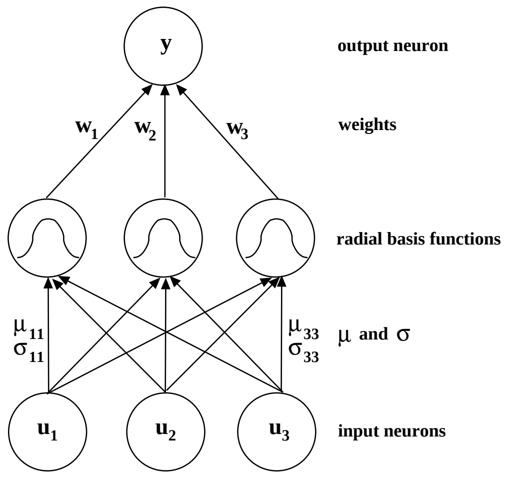


Figure 2.16 A basic model of an RBF network [Smi93, Bis95, p. 188-92 and p. 167-71]. In its most basic form, an RBF network consists of an input layer, a single hidden layer and an output layer. The input layer connects the RBF network to the environment. The hidden layer involves a nonlinear transformation of the input patterns to a (high-dimensional) hidden space, whereas the linear output layer supplies the network response to a certain activation pattern [Hay94, p. 256]. The hidden neurons of the RBF network are equipped with the radial basis functions [Bis95, p. 168]. The most common type of radial basis functions are Gaussian bumps [Bis95, p. 168]. Other types are e. g. multiquadrics or inverse multiquadrics [Hay94, p. 259]. In case of Gaussian basis functions, the turnable parameters of the RBF network are the weights w_i, the centers μ and the widths σ. The connections from the input to the hidden neurons determine the parameters of the radial basis functions, e. g. the centers μ and the widths σ of the Gaussian bumps [Bis95, p. 168-9]. Note, that due to visual clarity bias connections θ are omitted. Generally spoken, the training procedure of an RBF network consists of two stages [Bis95, Hay94, p. 170-6 and p. 298-305]: First, the parameters of the radial basis functions are determined (e. g. the centers μ and the widths σ of the Gaussian basis functions). Techniques for the latter optimization are discussed in e. g. Bishop (1995) [Bis95, p. 183-91]. Having calculated optimal values for the parameters of the radial basis functions, the weights w_i are determined. Here, linear optimization strategies are applied [Hay94, p. 298].

back into a context unit. The context unit delays the transmission of the activation value and sends the delayed information to the network. The latter connection also incorporates a weighting of the previous activation value. In the network, the context is derived from the activation values of the hidden neurons. Elman's time-delay recurrent neural network is depicted in Fig. 2.17 [CK01, p. 22].

Elman's network is often referred to as the simple recurrent network (SRN). The recurrent network is trained by a modified version of back-

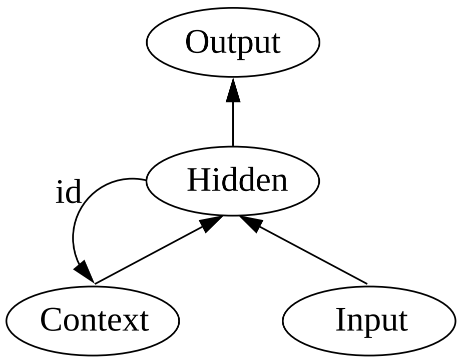


Figure 2.17 Elman's recurrent neural network [CK01, p. 21-2]: The network consists of an input layer, a hidden layer, a so-called context layer and an output layer. As a specialty, the activation values of the hidden layer are transferred to the context layer by a fixed identity connection *id*. The context layer delays the activation values and transfers the delayed information back to the network in form of additional input data. Besides the feedback connection from the hidden to the context layer, all network connections are turnable. Note, that the number of context units is equal to the number of hidden units [CK01, p. 21-2].

propagation through time, in which the temporal history is not considered [CK01, p. 22]. Similar types of time-delay recurrent neural networks are developed in parallel by Jordan, Williams and Zipser or Giles et. al. [Elm90, WiZi89, Gil91]. Jordan invented a "traditional three-layer-feedforward architecture with a set of context units that mirror the output layer activation and feed it back into the hidden layer" [CK01, p. 19]. A comprehensive description of Jordan's network is provided in Churchland and Sejnowski (1997) [CS97, p. 159-62]. Williams and Zipser proposed a single layer neural network with fully recurrently connected neurons [CK01, p. 23]. Giles et. al. introduced a so-called second-order recurrent neural network [CK01, p. 23-4]. The network only incorporates a single layer in which the "processing units are connected to each other and to the input nodes through a series of multipliers" [CK01, p. 23]. The term 'second-order' refers to the way in which the net input of the neurons is defined [Hay94, p. 737]. Details on second-order neural networks can be found in Churchland, Sejnowski (1997) [CS97, p. 162-3] and Haykin (1994) [Hay94, p. 737-9].

3.12 DILEMMA OF OVERFITTING & PURELY DATA DRIVEN MODELING

In the early 1990s another critical issue emerged, which addresses the generalization performance of neural networks. According to the well-known theorem of Hornik, Stinchcombe and White, a sufficiently large three layer feedforward neural network can approximate any continuous functional relationship on a compact domain [HSW92, p. 14-20]. However, neural networks may also suffer from their functional approximation abilities, because they might lose their generalization performance. This effect is known as overfitting [Zim94, p. 58-60].

Overfitting means that a complex neural network with too many free parameters not only learns the underlying dynamics of the target time series, but also fits the noise in the data [Zim94, p. 58]. Especially in economics, overfitting is a crucial issue. In economic applications, one typically has to identify highly nonlinear dynamic systems out of small data sets [Zim94, p. 58-9]. Under such conditions, the required complexity of the network is likely to cause overfitting. It is therefore not astonishing, that overfitting caused another downturn in the research field of neural networks.

Nevertheless, there are several approaches to avoid overfitting. For instance, one may think of methods for the optimization of the network structure (e. g. weight pruning, node pruning or weight decay), model selection criteria (e. g. specific cost functions, fitness and performance measures) or specific training procedures (e. g. early and late stopping) [Zim94, NZ98, Bis95, ZaRe99, p. 60-77, p. 405-18, p. 195-200 and p. 37-57]. The basic idea of these methods is to reduce the complexity of the model such that the training data allows a reliable fitting [NZ98, p. 413-17].

The standard way of modeling which is based on the functional approximation abilities of neural networks implies, that a time series identification task is transferred into a pattern recognition approach [NZ98, p. 373-4]. Unfortunately, the data is often covered with noise or there are unknown external influences. Hence, the data does not include all information which is required to describe the underlying dynamics. In this case, a pattern-recognition approach is clearly disadvantaged. In other words, the characteristics of the available data determine the quality of the neural network [NZ98, p. 373-4]. A solution to the outlined problem is provided by Hans Georg Zimmermann and his research group at Siemens Corporate Technology [NZ98, ZN01, ZNG01a].

According to Zimmermann, neural networks can be seen as a joint framework for the modeling of complex systems: Besides the learning from data, Zimmermann proposes to integrate prior knowledge about the dynamical system into the modeling. Furthermore, the modeling is enhanced by the integration of first principles, e. g. the separation of variants and invariants in high-dimensional dynamical systems (see chp. 4). The additional elements of the modeling are incorporated into the neural network in form of architectural building blocks. In the next chapter, we consider the 11-layer feedforward neural network, which provides deeper insights into this model building philosophy [ZN01, NZ98].

4. PRELIMINARY CONCLUSION

In this chapter we provided a basic introduction to neural networks. Starting off with the biological motivation of this research area, we pointed out, that neural networks are an approach of the embodiment of mind within an algebraic framework. Since the main interest of our work is confined to the modeling of economic market processes, we presented an uncommon interpretation of a neural network: A single neuron precisely reflects an elementary decision making process of a single market participant (agent). Correspondingly, a neural network can be seen as a model of interacting (economic) decisions or as a market process. As we will show in chapter 7, this interpretation provides the appealing crossover between the mathematical theory of neural networks and economics.

The 3-layer feedforward neural network is a well-known standard model. The network architecture consists of an input layer, a hidden layer and an output layer. The signal flow in the network is strictly feedforward. The modeling of a dynamic system with a simple 3-layer feedforward neural network is a critical issue: First, the task of capturing the underlying system dynamics is transferred into a pattern recognition approach. By this, we implicitly assume, that all relevant information needed to describe the dynamics is included in the training data. In case, the data is noisy or does not contain all information about the dynamics, a pattern recognition approach is clearly disadvantaged. Furthermore, 3-layer feedforward networks do not allow an explicit modeling of temporal structures. In a feedforward network, a certain input pattern is only fitted to an associated target pattern. As a consequence, a complicated preprocessing of the raw input data is required to achieve an appropriate description of the present time state.

An additional problem of (large) 3-layer feedforward neural networks is overfitting: Overfitting is a synonym for learning by heart. Due to their computational capabilities, 3-layer feedforward networks do not only fit the underlying structure of the time series, but also the noise included in the data. This leads to a poor generalization performance. Especially in economic applications, overfitting is a critical issue. Overfitting can be attacked by specific learning schemes (e. g. early or late stopping) together with optimization techniques for the network structure (e. g. weight or node pruning). Furthermore, advanced neural network architectures, which incorporate prior knowledge about the underlying application, are a possible solution for the overfitting dilemma. By the application of these neural networks, we also turn away from a purely data driven model building.

Chapter 3

MODELING DYNAMICAL SYSTEMS BY FEEDFORWARD NEURAL NETWORKS

In this chapter we deal with the modeling of dynamical systems by feedforward neural networks. A standard way of modeling is based on the well-known theorem, that a sufficiently large three layer neural network can in principle model any continuous functional relationship on a compact domain [HSW92, p. 14-20]. Relying on this standard paradigm, the time series identification problem is transferred into a pattern recognition approach [ZN01, Hay94, p. 311 and p. 67-8]. Typically, one starts off with a complicated preprocessing of the raw data in order to find an appropriate description of the present time state, which is given to the model in form of external input signals [NZ98, p. 373-4]. A major disadvantage of the latter preprocessing is that the representation of intertemporal structures can only be realized by preset time-lags of the used preprocessing functions [ZN01, p. 311-2]. Since a 3-layer feedforward network cannot represent temporal structures through a memory (superposition of past information), the identification of the underlying system dynamics is limited to a pure input to output data mapping [ZN01, p. 312].[1]

In the pattern recognition approach, one implicitly assumes, that a complete description of the underlying system dynamics is included in the training data set [NZ98, p. 373-4]. In other words, the result of the modeling depends solely on the data which is used for the fitting of the model, i. e. the characteristics of the available data determine the quality of the resulting model [ZN01, p. 311-2]. If *not* all information about the system dynamics is contained in the data or the data is covered with noise, a model building, which is solely based on the approximation abilities of neural networks, is clearly disadvantaged [NZ98, p. 373]. For instance, the amount of data might be too small in comparison to the complexity of the neural network, the data is covered with noise or

[1]In contrast, a time-delay recurrent neural network incorporates a memory [ZNG01a, p. 240 and 242]. If the system identification task is solved by an unfolding in time, the network architecture explicitly reflects temporal structures [RHW86, p. 354-57]. We will explain the modeling of dynamical systems by time-delay recurrent networks in chapter 4.

the available data does not sufficiently cover the high-dimensional input space and so forth [ZN01, NZ98, p. 311-2 and 373-4].

Following this reasoning, we believe that a model building which solely relies on the information contained in the data is often misleading [NZ98, p. 373]. This is especially true for economic applications: Here, one typically has to identify a high-dimensional non-linear dynamic system on the basis of noisy data or insufficiently small data sets [Zim94, p. 29]. Instead of referring to the outlined pattern recognition approach, we propose neural networks which include prior knowledge of the dynamic system in form of architectural enhancements. By this, the model building can be enhanced in the sense that more relevant information about the underlying dynamical system is included in the modeling [NZ98, p. 374-5]. A feedforward neural network architecture, which is based on the latter philosophy of model building, is the so-called 11-layer architecture [NZ98, p. 375-86].

The 11-layer feedforward neural network is especially designed for the modeling of dynamic systems. In particular, the 11-layer network consists of several building blocks, which incorporate prior knowledge about the dynamic system in form of architectural extensions of the underlying feedforward neural network [ZN01, NZ98, p. 318-21 and p. 383-6]. In other words, the 11-layer architecture handles different problems of data modeling in different parts of the network [NZ98, p. 383-6]. Remarkably, the building blocks complement one another such that an advanced feedforward neural network for the modeling of dynamic systems is created. The 11-layer architecture achieves only a maximal effect of synergy by the joint application of all building blocks [ZN01, p. 318-21].

Among the different building blocks of the 11-layer architecture, a network internal preprocessing based on hyperbolic tangent squashing [Bis95, p. 127] is used for the handling of outliers in the data set (sec. 1.). The preprocessing of the raw input data is simplified. Only the momentum and force of each raw input time series are calculated with respect to different time horizons [NZ98, PDP00, p. 374-5 and p. 102-5]. The 11-layer architecture is prepared for a similarity or discrimination analysis by the integration of a square layer [Fla97, NZ98, p. 146-8 and p. 378-9]. Using square values of the input signals within an additional hidden cluster of the 11-layer architecture, we merge the (global) difference analysis of multi-layer perceptrons and the (local) similarity analysis of RBF networks (sec. 2.). Additional information of the underlying dynamic system is incorporated into the 11-layer architecture by using the concept of forces and embeddings [NZ98, p. 383-6], which allows a complete characterization of the time series (sec. 3.). This idea is based on the well-known Takens theorem [Tak81]. As a side effect, the forces and embeddings generate additional error flows during the training of the network. Hence, the network signal flows of the forward and backward pass are implicitly balanced [ZW97, NZ98, p. 292-3 and p. 384-5]: The high-dimensional input vector of the neural network is *not* only provided with the gradient information of a low-dimensional output vector, but also with the error flow generated by the forces and embeddings. Finally, the 11-layer architecture uses the technique of ensemble forecasting [NZ98, Perr93, p. 381-83]: A series of forecasts is computed out of the forces and embeddings. In case, the errors of these forecasts are uncorrelated, an averaging over the series improves the model prediction ability (sec. 4.). In the following, we introduce the different building blocks and show how these features are integrated into a unified approach, the 11-layer architecture (sec. 5.) [ZN01, NZ98, p. 318-21 and p. 383-6].

As a result, we obtain a powerful class of functions for the modeling dynamical systems [ZN01, p. 312 and p. 321]. In other words, the 11-layer architecture is able to provide us with a better description of the underlying system dynamics and thus, improves forecasting. The principle task of the 11-layer architecture is the analysis of (financial) time series [NZ98, p. 374-5].

We illustrate the forecasting ability of the 11-layer architecture by its application to a real world problem (sec. 6.). Forecasting the semi-annual development of the German bond index, it turns out, that the 11-layer neural network is superior to more conventional forecasting techniques [NZ98, p. 419-20].

1. INTERNAL PREPROCESSING

Compared to traditional feedforward neural networks, the 11-layer architecture uses a relative simple preprocessing of the raw input signals [ZN01, NZ98, p. 312-3 and p. 374-5].[2] More precisely, only (*i.*) the *momentum* and (*ii.*) the *force* of each raw input time series are computed with respected to different time horizons (see Eq. 2.7 and Eq. 2.8). While the momentum describes the relative change in the underlying input time series, the force is used to characterize its turning points. In combination with each other, these preprocessing functions provide an appropriate description of the raw input signals [NZ98, p. 374-5]. Due to numerical reasons, the preprocessing also includes a scaling of the transformed

[2]An overview of other preprocessing techniques is provided in e. g. Bishop (1995) [Bis95, p. 296-300] or Zimmermann (1994) [Zim94, p. 20-28].

input data, such that the signals have a mean of zero and unit variance [PDP00, NZ98, p. 73 and p. 374-5].

However, outliers in the data set may have a large impact on the modeling, because the learning of the network parameters is somewhat interferenced [NZ98, Wei90, p. 375 and p. 195-200]. Especially in economics, outliers are common problem, e. g. unexpected shocks (political or economic crises) or so-called 'information shocks' (news, not in line with the market expectations) [ZN01, NZ98, p. 314 and p. 387].

For the handling of outliers, we propose an additional network internal preprocessing of the scaled input signals u_i. The following transformation is applied to each input signal u_i [NZ98, p. 375]:

$$u_i' = \tanh(w_i u_i) \ . \tag{3.1}$$

As stated in Eq. 3.1, the scaled input signals are transferred through a hyperbolic tangent. Small inputs pass the hyperbolic tangent nearly unchanged due to its linear range near the origin, whereas outliers are damped by its saturation level [NZ98, ZN01, p. 375-6 and p. 313-4]. A neural network formulation of the outlier cancellation (Eq. 3.1) is depicted in Fig. 3.1 [NZ98, p. 376].

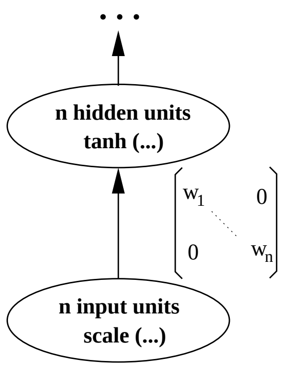


Figure 3.1 Network internal preprocessing for the handling of outliers. An input layer consisting of n input units is connected to a hidden layer with n hidden units. The connection is square diagonal matrix with positive weights $w_i > 0$. The activation function of the hidden layer is a hyperbolic tangent [ZW97, NZ98, p. 295-6 and p. 375-6].

The weights of the diagonal connector between the input and the hidden layer should be initialized with small values of $w_i = 0.1$ [ZW97, NZ98, p. 295 and p. 376]. This ensures, that at the beginning of the training all of the input signals pass the hyperbolic tangent nearly unchanged. If the outliers should be absorbed right from the beginning of the training, the weights of the diagonal connector have to be initialized with larger values, e. g. $w_i = 1$ [ZN01, NZ98, p. 314 and p. 376]. For economic applications, small initial values of the weights are preferable for short-term forecasting (e. g. modeling of daily stock returns), while larger values are recommended for long-term predictions (e. g. a six month forecast horizon based on monthly observations). The reason for this procedure is that monthly economic data is more likely to contain unexpected (political) shocks [NZ98, p. 376].

During their experiments, Neuneier and Zimmermann (1998) discovered, that the training stability of the preprocessing layer can be improved by (*i.*) allowing only positive weights in the diagonal connection ($w_i > 0$) and by (*ii.*) the exclusion of bias weights in the preprocessing layer [NZ98, p. 376]. Positive weights $w_i > 0$ in the diagonal connection ensure that the sign of the input signals is not affected by the internal preprocessing. The exclusion of bias weights prevents numerical ambiguities.

Apart from the latter constraints, the weights of the diagonal connection are trained in the same way as all other parameters of the neural network [ZW97, p. 296]. In real world applications, Neuneier and Zimmermann observed large, medium and small values for the weights of the diagonal connector [NZ98, p. 376]. While large weight values indicate, that a larger proportion of the input signal is damped by the hyperbolic tangent squashing function, small values are an indicator of unimportant inputs, which should be eliminated from the network. Medium weight values indicate that the related input signals pass the network internal preprocessing nearly unchanged.

2. MERGING MLP AND RBF NETWORKS

The 11-layer architecture includes a similarity or discrimination analysis, which combines the global segmentation properties of multi-layer perceptrons (MLP) with the local segmentation properties of radial basis function (RBF) networks [NZ98, p. 378-9]. The idea of global and local decision making is originally proposed by G. Flake [Fla97, p. 146-48]. For a comparison of MLP and RBF networks see Haykin (1994) [Hay94, p. 293-4] and Bishop (1995) [Bis95, p. 182-3].

The integration of global (MLP, sigmoidal activation functions [Hay94, Bis95, p. 156 and p. 116-20]) and local (RBF, Gaussian basis functions [Hay94, Bis95, p. 256 and p. 167-70]) segmentation properties into a neural network is as follows [NZ98, p. 378-79]: First, we introduce a so-called 'square' layer to the network architecture. The 'square' layer is designed as a hidden layer that incorporates a quadratic squashing function. We use the 'square' layer to compute the squared values of the preprocessed input signals [NZ98, p. 378]. Now, if we process the input signals *and* their squared values within an additional hidden layer that is equipped with hyperbolic tangent squashing, the network is enabled to combine global and local decision making [NZ98, p. 378].

Let us describe the combination of global and local segmentation properties in mathematical terms [NZ98, p. 378]: The output z of a standard 3-layer feedforward neural network using hyperbolic tangent tanh(.) squashing can be written as

$$z = \sum_j v_j \tanh\left(\sum_i w_{ji} u_i - \theta_j\right) . \tag{3.2}$$

In contrast, the output z of a comparable RBF network incorporating Gaussian basis functions is

$$z = \sum_j v_j e^{-\frac{1}{2}\sum_i \left(\frac{u_i - \mu_{ji}}{\sigma_{ji}}\right)^2} . \tag{3.3}$$

Now, G. Flake [Fla97, p. 147] combines Eq. 3.2 and Eq. 3.3 in the following way

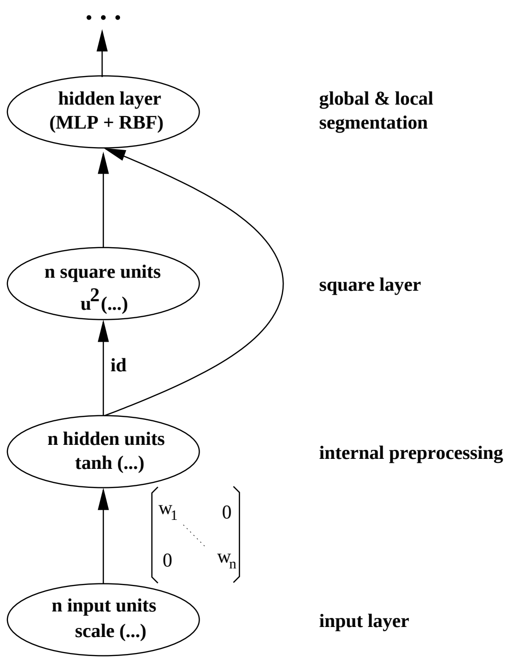
$$z = \sum_j v_j \tanh\left(\sum_i w_{ji} u_i + r_{ji} u_i^2 - \theta_j\right) . \tag{3.4}$$

In Eq. 3.4 the output of the RBF network (Eq. 3.3) is approximated up to a sufficient level by the squared inputs u_i^2 [NZ98, p. 378]. Simultaneously, the original input signals u_i ensure that Eq. 3.4 maps the output of the multi-layer perceptron (Eq. 3.2) [NZ98, p. 378].

A fragment of a feedforward neural network architecture incorporating a network internal preprocessing (sec. 1.), the proposed 'square' layer and an additional hidden layer with tanh squashing is depicted in Fig. 3.2 [NZ98, p. 379].

A combination of the network internal preprocessing layer and the square layer (see Fig. 3.2) has some advantages [NZ98, p. 378]: First, we utilize the automatical handling of outliers in the input data. Second, the exclusion of a particular input leads to a simultaneous elimination of its squared values.

The experiments of Neuneier and Zimmermann indicate, that the additional free parameters of this approach do not cause an acceleration of overfitting [NZ98, p. 378]. Remarkably, the overfitting of such a network is actually slowed down. Furthermore, the authors observed that a good



...
hidden layer (MLP + RBF)
global & local segmentation
n square units u^2(...)
square layer
id
n hidden units tanh (...)
internal preprocessing
w_1 0 0 w_n
n input units scale (...)
input layer

Figure 3.2 Network internal preprocessing combined with a square layer for the preparation of global and local segmentation. The outputs of the network internal preprocessing are fed into the 'square' layer by an identity matrix *id*. The size n of the 'square' layer is equal to that of the preprocessing layer. The 'square' layer incorporates a quadratic squashing function in order to compute the squared values of the preprocessed input signals. The identity matrix between the preprocessing and the squared layer is frozen during the training and recall of the network. The additional hidden layer (tanh squashing) at the top of the neural network fragment is able to create MLP and RBF structures (see Eq. 3.4). Due to visual clarity, the bias connection to the hidden layer merging global and local segmentation is suppressed. Note, that the squared layer does *not* incorporate any bias connections [NZ98, p. 378-79].

generalization performance can be achieved even if the training error of the network is very small [NZ98, p. 378]. This observation contradicts the early stopping criterion for the training of neural networks [NZ98, p. 410-13].

These results can understood by the global and local segmentation properties of the suggested architecture [NZ98, p. 378]: Due to the local properties of the architecture, local structures of the input space can be separated such that they have no global impact on the model. This effect may increase the generalization performance of the neural network.

3. FORCES AND EMBEDDINGS

In this section, we introduce the concept of forces and embeddings, which increases the information flow in the backward path of the network by generating additional error signals for the learning of the weights [NZ98, p. 383-86 and p. 379-80].

A typical problem of standard 3-layer feedforward neural networks is, that a huge number of input signals is only supplied with the error signal of a single output neuron [ZW97, ZN01, p. 292 and p. 316-17]. Since the learning can only utilize the information of a single output neuron for adapting the network weights, the information flow along the forward and the backward path of the network is heavily unbalanced [NZ98, p. 379]. An obvious solution to this problem is to increase the error flow of the network by a number of additional output neurons [NZ98, p. 383-86 and p. 379-80].

Now the question arises, which kind of additional information is appropriate for our task of modeling dynamical systems. We propose to characterize the autonomous part of the dynamics on the output side of the network in a way, which is similar to the Takens' theorem for the notion of state on the input side [Hay94, Tak81, p. 714-6].

Following the Takens' theorem, we refer to the concept of forces and embeddings to characterize the underlying dynamical system [NZ98, p. 384-85]. The forces and embeddings are calculated on basis of m different time slots along the forecast horizon $t + n$. Presuming m different time slots around the forecast horizon $t + n$ the forces r_i can be written as

$$r_i \quad = \quad \frac{-y_{t+n+i} + 2y_{t+n} - y_{t+n-i}}{3y_t} \; , \quad \forall \quad i = 1, \dots m \; , \tag{3.5}$$

whereas the embeddings s_i can be formulated as

$$s_i \quad = \quad \frac{y_{t+n+i} + y_{t+n} + y_{t+n-i}}{3y_t} - 1 \; , \quad \forall \quad i = 1, \dots m \; . \tag{3.6}$$

As shown in Eq. 3.5, the forces r_i are formulated as curvatures of the underlying time series of target values y [ZN01, NZ98, p. 319-20 and p. 384]. By this, we force the network to focus on the turning points of the target time series (see also the preprocessing function of Eq. 2.8). On the other hand, Eq. 3.6 describes a normalized 3-point embedding [ZN01, NZ98, p. 319-20 and p. 384]. Forces (Eq. 3.5) and embeddings (Eq. 3.6) are calculated with respect to the forecast horizon $t + n$. The different time slots m allow a comprehensive description of the features of the underlying dynamic system [NZ98, p. 384]. This corresponds to the idea of characterization by points in Takens' Theorem [Hay94, Tak81, p. 714-6].

We suggest to choose the width m of the forces and embeddings with respect to the forecast horizon $t+n$. For example, in case of a prediction horizon of six time periods ($n = 6$), a value of $m = 6$ should be an appropriate range for the forces and embeddings [NZ98, p. 385].

A neural network architecture incorporating the concept of forces and embeddings is depicted in Fig. 3.3 [NZ98, p. 380 and p. 386].

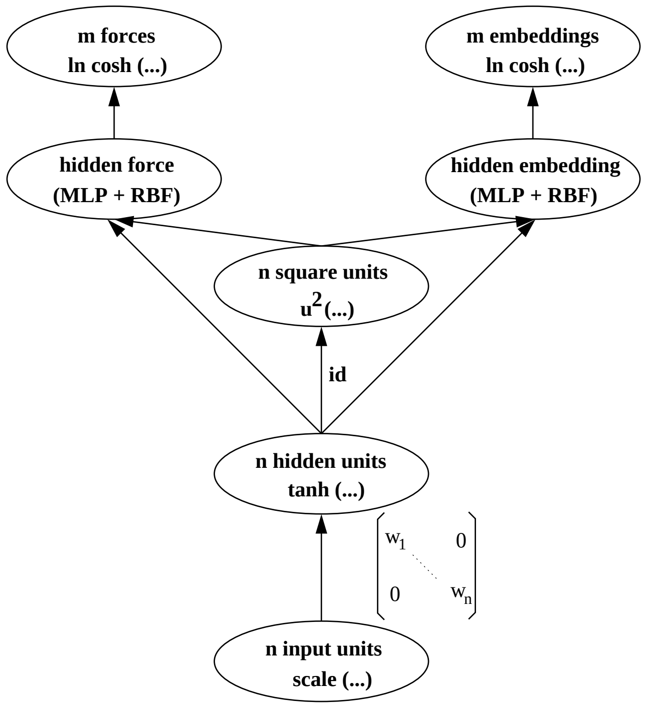


Figure 3.3 Neural network architecture for the estimation of forces (Eq. 3.5) and embeddings (Eq. 3.6). Forces and embeddings are located in two different branches of the neural network in order to avoid interferences during the training [NZ98, p. 384-85]. Again, we use a network internal preprocessing (sec. 1.) for the cancellation of outliers. The input signals and their squared values are fed into two different hidden layers entitled 'hidden force' and 'hidden embedding'. The hidden layers are equipped with hyperbolic tangent squashing and may therefore generate global and local structures [Fla97, p. 146-48]. Forces and embeddings are integrated into separate output layers at the top of the neural network. According to the preset time slot m, the output layer entitled 'forces' includes m output units subjected to Eq. 3.5, whereas the output layer entitled 'embeddings' computes m embeddings subjected to Eq. 3.6. We propose to used the robust cost function $\ln\cosh(.)$ for these output layers [NZ98, p. 387-88].

Remarkably, the sum of a force r_i (Eq. 3.5) and the corresponding embedding s_i (Eq. 3.6) computes to a forecast of our intended target [ZN01, NZ98, p. 320 and p. 385]:

$$r_i + s_i \;=\; \frac{y_{t+n} - y_t}{y_t} \;, \quad \forall \quad i = 1, \ldots m \; . \tag{3.7}$$

According to Eq. 3.7, forces and embeddings can be seen as indirect forecasts of our underlying target [ZN01, NZ98, p. 320 and p. 385]. As we will show later on (sec. 4.), this relationship is the basis for the calculation of an ensemble forecast [ZN01, p. 317-18].

A somewhat similar concept is introduced by Weigend and Zimmermann [ZW97, p. 292-4]. The authors invent a so-called point prediction layer in combination with a subsequent interaction layer [ZW97, p. 292-4]. These output layers enhance the backward flow of the network by additional error signals [ZW97, p. 292]. Other concepts and solutions to balance the signal flow in a neural network are proposed by e. g. Weigend, Rumelhart and Huberman [WRH90, p. 876-79] or Caruana [Car94, p. 658-660].

The concept of the point prediction and interaction layer is as follows [NZ98, p. 379-81]: Suppose, we are interested in forecasting y_{t+n}. The point prediction layer forces the neural network to learn y_{t+n-1}, y_{t+n} and y_{t+n+1}. The interaction layer, which is subsequent to the point prediction layer, computes the differences $(y_{t+n} - y_{t+n-1})$, $(y_{t+n+1} - y_{t+n})$ and the curvature $(y_{t+n+1} - 2y_{t+n} + y_{t+n-1})$.

A neural network architecture incorporating a point prediction and an interaction layer is shown in Fig. 3.4 [ZW97, NZ98, p. 293 and p. 380].

Let us focus on the effect of the point prediction and interaction layer [ZW97, NZ98, p. 293 and p. 380-81]: In case, the point predictions are perfect, the interaction layer has no effect, because the forecasts of the neighboring differences and the curvature are perfect as well. However, as depicted in Fig. 3.5 [ZW97, NZ98, p. 294 and p. 381], the interaction layer has a desirable effect in case of false point predictions.

Fig. 3.5 depicts the point prediction errors of two different models (dotted line: model 1; dashed line: model 2). Model 1 and 2 have identical point prediction errors [NZ98, p. 380-81]. Now, the interaction layer favors model 1, because it predicts the neighboring differences and the curvature appropriately. In this case, the interaction layer does not cause additional error signals [NZ98, p. 380-81]. In contrast, model 2 is penalized by the interaction layer, because the neighboring differences and the curvature are incorrectly predicted. As a result, model 1 is

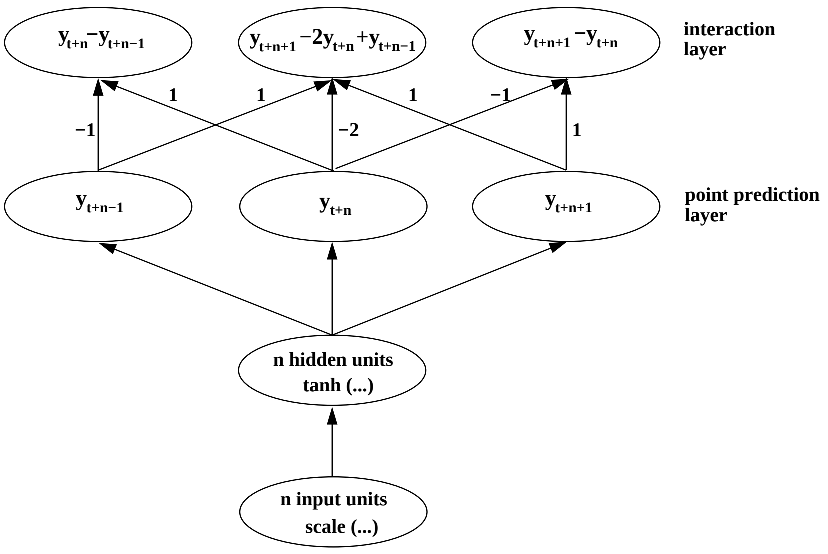


Figure 3.4 Point prediction and interaction layer within a feedforward neural network. The point prediction layer generates error signals for y_{t+n-1}, y_{t+n} and y_{t+n+1}. Based on these forecasts, the subsequent interaction layer computes the next-neighbor differences as well as the curvature. In order to compute the overall error function of the network, one has to sum the (equally weighted) error signals of the point prediction and interaction layer [ZW97, p. 293]. All connections between the point prediction and the interaction layer are frozen during the training and recall of the network [ZW97, NZ98, p. 292-4 and p. 380].

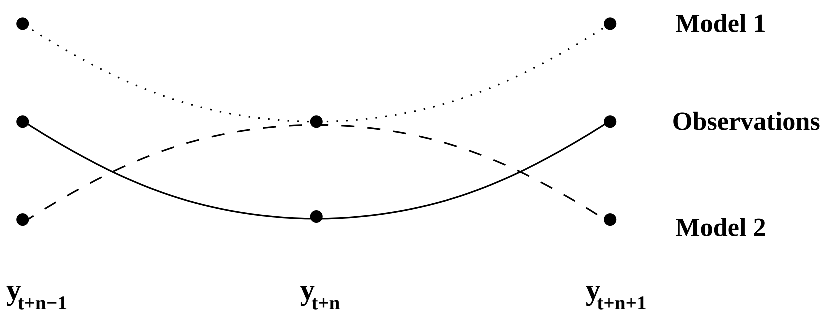


Figure 3.5 Geometric illustration of the impact of the interaction layer on the network overall error function [NZ98, p. 380-81]

preferred by the interaction layer, because the overall error of model 2 is larger than that of model 1 [NZ98, p. 380-81].

Weigend and Zimmermann have discovered in their experiments, that a stand-alone point prediction layer leads to unsatisfactory results, because the point predictions cause only an insufficient amount of interaction within the neural network [ZW97, p. 305]. Remarkably, the interaction layer can also be used to model specific relationships among the targets of the neural network. For example, one may apply an interaction layer to describe the interdependencies between different financial markets [ZN01, NZ98, p. 317 and p. 381].

4. MULTI-FORECASTS AND STATISTICAL AVERAGING

Suppose, we have a number of m forecasts for the same target, which are provided from m different neural networks. A merging of these forecasts into a unified prediction is a well-known technique which is likely to achieve a better generalization performance and more stable results than the single network forecasts [Hay94, NZ98, p. 353 and p. 381]. In other words, an average over the m forecasts may outperform the predictions of the individual networks on the average [NZ98, p. 381]. This technique is also referred to as ensemble forecasting [Hay94, p. 353]. The PhD thesis of M. P. Perrone provides a comprehensive survey of ensemble forecasting and includes also a large bibliography on the subject [Perr93]. Within this work, we will only focus on a simple statistical averaging of a series of forecasts [NZ98, p. 381-2].

"Averaging the output of several networks may give us a better and more stable result" [NZ98, p. 381]. As an explanation for this statement, let us consider the case of m different neural networks that have been trained on a given data set. The neural networks generate different forecasts, because the network designs differ or they converge to different local minima during the training [NZ98, Hay94, p. 381 and p. 353]. The expectation is that the overall prediction accuracy can be improved by a forecast ensemble in form of an average [NZ98, p. 381].

Let us clarify the statistical benefits of such a forecast ensemble by examining the overall error function E_{average} of the m sub-networks [NZ98, p. 381-82]. Given a set of T training patterns, the observed target data y^d_{t+n} and the model forecasts $y_{i,t+n}$ of the $i = 1, \ldots, m$ sub-networks, the mean square error of the forecast ensemble computes to

$$E_{\text{average}} = \frac{1}{T} \sum_{t=1}^{T} \left[\left(\frac{1}{m} \sum_{i=1}^{m} y_{i,t+n} \right) - y^d_{t+n} \right]^2 . \qquad (3.8)$$

Rearranging the terms of Eq. 3.8 leads to

$$E_{\text{average}} = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{m}\left(\sum_{i=1}^{m} y_{i,t+n} - y_{t+n}^{d}\right)\right]^2 . \tag{3.9}$$

Now, let us assume that the individual mean square errors of the m different sub-networks are uncorrelated [NZ98, p. 382]. This can be formulated as

$$E_{\text{model}_i,\text{model}_j} = \frac{1}{T}\sum_{t=1}^{T}(y_{i,t+n} - y_{t+n}^{d})(y_{j,t+n} - y_{t+n}^{d}) = 0 , \tag{3.10}$$

with $\forall i \neq j$ [NZ98, p. 382]. Under the assumption that the errors of the sub-networks are uncorrelated (Eq. 3.10), we can rewrite Eq. 3.9 as follows [NZ98, p. 382]:

$$E_{\text{average}} = \frac{1}{m}\left(\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{T}\sum_{t=1}^{T}(y_{i,t+n} - y_{t+n}^{d})^2\right)\right) . \tag{3.11}$$

Finally, reformulating the terms of Eq. 3.11 leads to

$$E_{\text{average}} = \frac{1}{m}\,\text{average}\left(E_{\text{sub-networks}}\right) . \tag{3.12}$$

According to Eq. 3.12, the overall mean square error of an ensemble of m different neural networks is by a factor of m *lower* than the average error of the sub-networks [NZ98, p. 382]. However, it should be noted, that statistical averaging does *not* provide additional prior knowledge about the underlying dynamic system [NZ98, p. 382]. Nevertheless, statistical averaging may reduce the error which is due to the uncertainty of the training and thus, provides us with better and more stable results [NZ98, Hay94, p. 381 and p. 353].

We propose to combine a statistical averaging with the concept of forces and embeddings [NZ98, p. 384-5]. Already mentioned above (Eq. 3.7), the m forces and embeddings generate m forecasts of our intended target. This series of predictions can be used to form a forecast ensemble [NZ98, p. 384-5]. A segment of a neural network architecture which generates a series of forecasts out of the forces and embeddings and subsequently constructs an average forecast is depicted in Fig. 3.6 [NZ98, p. 382].

Note, that the multi forecast as well as the average forecast layer of the neural network in Fig. 3.6 can be interpreted as interaction layers, which support the identification of the underlying dynamical system [NZ98, p.

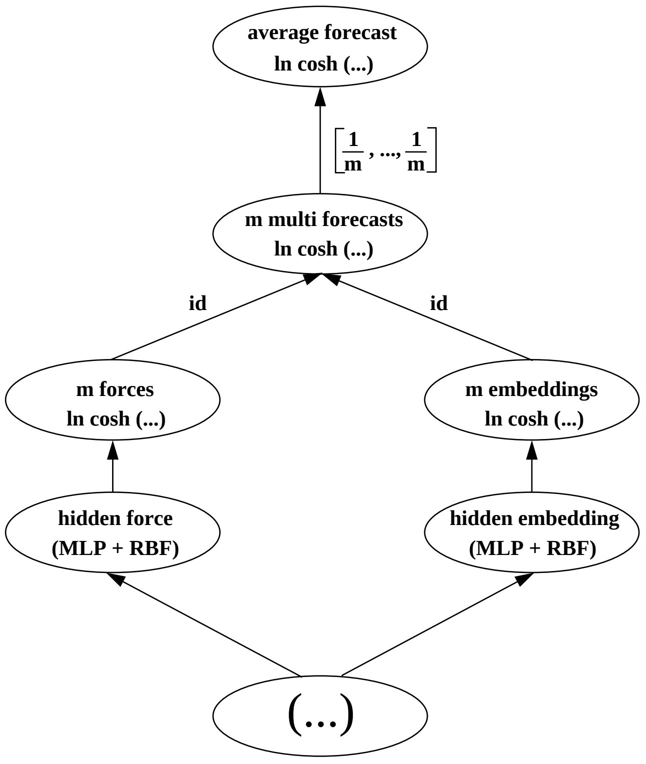


Figure 3.6 Multi forecasting and statistical averaging in a feedforward neural network architecture [NZ98, p. 382]. With respect to a preset width m, the neural network generates m forces and m embeddings (sec. 3., Fig. 3.3). The forces and embeddings are transferred into the multi forecast layer by fixed identity matrices *id*. By combining the forces r_i with the corresponding embeddings s_i, the multi forecast layer computes a series of m forecasts (sec. 3., Eq. 3.7). In order to calculate an average over the m forecasts, we connect the multi forecast layer to the average forecast layer. The particular connection between these layers contains fixed weights w_i. The value of each weight is $w_i = 1/m$. Proceeding this way, the final output of the neural network, i. e. the average over the series of m forecasts, is provided by the average forecast layer. Note, that both, the multi forecast and the average forecast layer are designed as output clusters using the robust cost function $\ln\cosh(\cdot)$ [NZ98, p. 387-88].

385-86]. Furthermore, the forecast ensemble of Fig. 3.6 does *not* increase the number of free network parameters, because the required connections are fixed during the training and recall of the network [NZ98, p. 385-86].

5. THE 11-LAYER NEURAL NETWORK

Up to now, we have introduced several architectural building blocks which support the modeling of dynamic systems by the integration of prior knowledge about the application. In this section, we integrate the architectural building blocks into *one* 11-layer neural network [NZ98, ZN01, p. 383-86 and p. 318-321]. The application field of the 11-layer architecture is the analysis of (financial) time series, i. e. the development of econometric forecast models [NZ98, p. 373-5]. Remarkably, only the joint application of all building blocks in one neural network creates a maximal effect of synergy such that the modeling of dynamic systems is supported [NZ98, p. 374-5 and p. 383-86]. The 11-layer architecture is depicted in Fig. 3.7 [NZ98, ZN01, p. 386 and p. 319].

The 11-layer architecture (Fig. 3.7) includes a relative simple preprocessing of the raw input time series [NZ98, p. 374-5]: For each raw input we calculate only the *momentum* (relative difference, see Eq. 2.7) and the *force* (curvature, see Eq. 2.8). In combination with each other, these simple preprocessing transformations provide an appropriate description of a raw input signal [NZ98, p. 374]. Note, that the transformed input signals are scaled. A scaled input time series has a mean of zero and a statistical variance of one [NZ98, PDP00, p. 374-5 and p. 73].

The transformed input time series are fed into a network internal preprocessing layer. The network internal preprocessing is used for the cancellation of outliers. According to the n input signals, the network internal preprocessing layer consists of n hidden neurons, which are equipped with hyperbolic tangent squashing. Outliers are damped by the saturation level of the hyperbolic tangent, while other data points pass the activation function nearly unchanged. The connection between the input and the preprocessing layer is diagonal matrix with positive weights ($w_i \geq 0$) (sec. 1.).

Having passed the network internal preprocessing, the squared values of the input signals are computed by a 'square' layer. The 'square' layer is composed of n hidden neurons, which are endowed with quadratic activation functions. The connection between the network internal preprocessing layer and the 'square' layer is a fixed identity matrix *id*. Feeding the preprocessed input signals and the squared values into a hidden layer with hyperbolic tangent squashing, the network is enabled "to act as a combination of widely known neural networks using sigmoidal activation function (MLP) and radial basis function networks (RBF)" [NZ98, p. 378]. Within the 11-layer architecture, we have two hidden layers (entitled 'hidden force' and 'hidden embedding'), which create MLP and RBF structures [NZ98, p. 384-5].

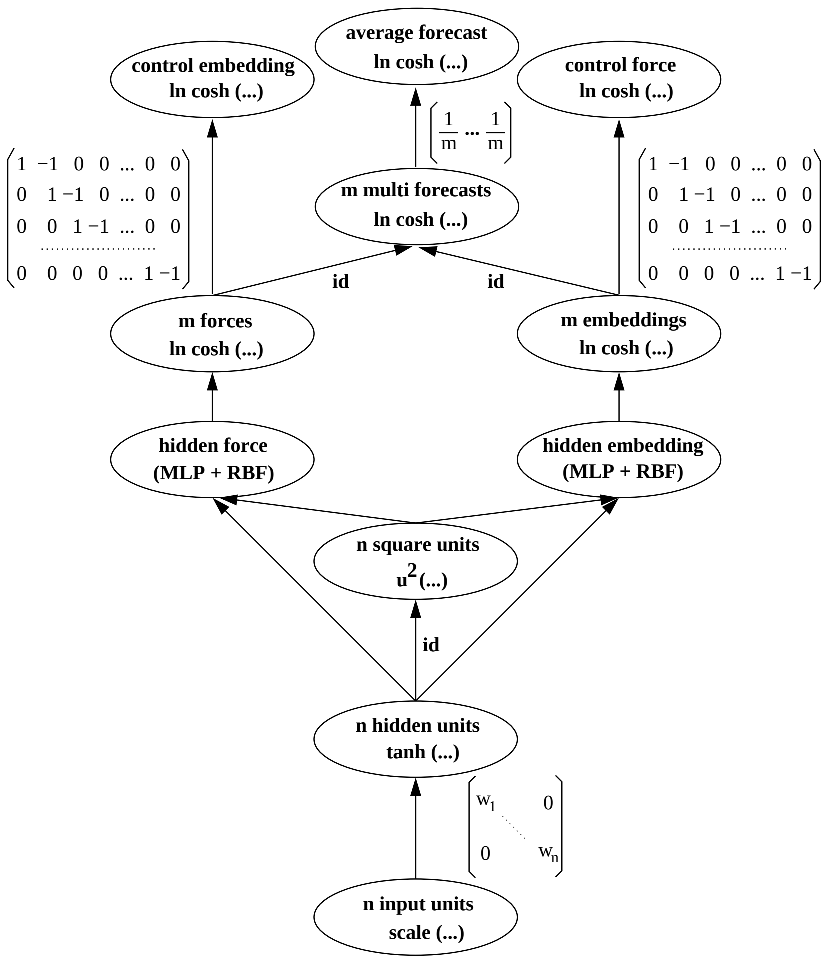


Figure 3.7 The 11-layer feedforward neural network for the modeling of dynamic systems [NZ98, p. 386]. Due to visual clarity, bias connections are suppressed.

Starting with the hidden force and the hidden embedding layer, the 11-layer architecture (Fig. 3.7) is composed of two branches, which are used to calculate the forces and the embeddings of the underlying target time series. The separation of forces and embeddings is performed to avoid inferences during the learning. The concept of forces and embeddings is related to the well-known Takens' theorem [Hay94, Tak81, p. 714-16]. Calculating the forces and embeddings with respect to m different time slots along the forecast horizon $t+n$, we add additional prior knowledge about the underlying dynamic system to the neural network.

Furthermore, the signal flows along the forward and backward path of the network are balanced [ZW97, p. 292]. According to the m preset time slots, we calculate m forces and m corresponding embeddings. The embeddings are described as normalized 3-point embeddings, while the forces are formulated as curvatures (sec. 3., Eq. 3.5 and 3.6). Forces and embeddings are organized in separate output clusters, which are equipped with the robust error function $\ln\cosh(.)$ [NZ98, p. 387-88].

The 11-layer architecture (Fig. 3.7) also includes an ensemble of forecasts, i. e. several forecasts of our intended target are combined to one unified prediction. Presuming that the individual forecast errors are uncorrelated, a statistical averaging provides us with a better generalization performance and more stable results [ZN01, p. 317-18]. We calculate a series of m forecasts of our intended target by combining the m forces and embeddings in pairs: For each time slot m, we add the particular force r_i and the corresponding embedding s_i (sec. 3., Eq. 3.7). To perform the summation, the forces and embeddings are connected to the multi forecast layer using fixed identity matrices. Subsequently, the average over the m forecasts is computed by connecting the multi forecast layer to the average forecast layer. The particular connection consists of fixed weights $w_i = 1/m$ (sec. 4., Fig. 3.6).

As a specialty, we include a control-force and a control-embedding output layer into the 11-layer network [NZ98, p. 385]. These output clusters focus on the difference between neighboring forces resp. embeddings. For instance, the neural network has not only to learn the forces r_i and r_{i+1}, but also the difference $r_i - r_{i+1}$ which is located in the control force layer [NZ98, p. 385]. The same procedure applies for the embeddings [NZ98, p. 385]. Neuneier and Zimmermann added the control-force and the control-embedding layers, because the authors observed, that the network often had to estimate too similar features on the force resp. embedding level [NZ98, p. 385]. "To counteract this behavior we have to add two additional clusters which control the difference between the individual outputs inside the embedding and the force cluster" [NZ98, p. 385].

Although the 11-layer architecture seems to be relatively complicated, most of the network connections are fixed during the learning [NZ98, p. 383-5]. By the application of fixed weights, we only route the information flow within the 11-layer network in order to integrate additional prior knowledge about the underlying task [NZ98, p. 383-5].

The description of the 11-layer neural network clarifies that the different architectural building blocks supplement each other [ZN01, p. 318-21]: For instance, the summation of the forces and embeddings yields a series of forecasts, which is in turn utilized for the construction of an ensemble forecast. Note, that the multi and average forecast layers, the force and embedding layers as well as the control force and control embedding layers can be interpreted as interaction layers [ZN01, p. 320-21]. In combination with each other, these interaction layers support the identification of the underlying dynamical system [ZN01, p. 320].

6. EMPIRICAL STUDY: FORECASTING THE GERMAN BOND RATE

In this section, we apply the 11-layer architecture (Fig. 3.7) and a standard 3-layer feedforward neural network (Fig. 2.8) [Hay94, Bis95, p. 156 and p. 116-20] to the German bond market. Forecasting the semi-annual development of the German bond rate, it turns out, that on the basis of common performance measures (Sharpe ratio, hit rate, realized potential, annualized return and profit & loss curves) the 11-layer architecture is superior to the standard 3-layer neural network (MLP). The concept of this empirical study widely follows Neuneier and Zimmermann (1998), [NZ98, p. 419-20]. Note, that the authors report similar results [NZ98, p. 419-20].

The basics of our empirical study can be characterized as follows: We are working on the basis of monthly data from April 1974 to Oct. 1995 [261 data points] to forecast the semi-annual development of the German bond rate. The complete data set is divided into *two* subsets: the training set (in-sample period) covers the time period from April 1974 to May 1990 [196 data points], while the generalization set (out-of-sample period) covers the time from June 1990 to Oct. 1995 [65 data points]. The training set is used for the learning of the network parameters. The generalization set is used to evaluate the performance of the 11-layer network and the benchmark.

The input data base of both neural networks consists of 10 raw time series. More precisely, we included business cycle, inflation, stock market and foreign exchange indicators of Germany, USA and Japan.

In the preprocessing we calculate the *momentum* (Eq. 2.7) and *force* (Eq. 2.8) of each raw input time series. Additionally, we included a scaling, i. e. the resulting input signals have a mean of zero and a unit variance. Using these preprocessing transformations, we obtain a number of 20 input signals for our neural networks.

The 11-layer architecture which is used to predict the development of the German bond rate on a six month forecast horizon is equivalent to Fig. 3.7. According to the six month forecast horizon, the width m of the forces and embeddings is preset to $m = 6$ (see Eq. 3.5 and 3.6). The hidden force as well as the hidden embedding layer consist of 20 hidden neurons.

As a benchmark for the 11-layer architecture, we employed a 3-layer feedforward neural network (MLP, see Fig. 2.8) [Hay94, Bis95, p. 156 and p. 116-20]. The MLP consists of an input layer (20 input neurons), one hidden layer (20 hidden neurons equipped with hyperbolic tangent squashing [Bis95, p. 127]) and one output layer (1 output neuron with $\ln\cosh(.)$ cost function [NZ98, p. 387-88]).

Both neural networks are trained until convergence by using standard error backpropagation in combination with pattern-by-pattern learning rule [Zim94, p. 37-40 and p. 40-49]. We optimize the MLP by EBD weight pruning [TNZ97, NZ98, p. 670-72 and p. 406-7].

The forecasting accuracy of the neural networks is evaluated by *five* common performance measures: Sharpe ratio, hit rate, realized potential, profit & loss curve and annualized return of investment [PDP00, Ref95, p. 267-8 and p. 70-74]. The Sharpe ratio is defined as the excess return of the model (difference between the model return r_M and a risk-free return r_f) divided by the standard deviation σ_M of the model return, i. e. $SR = (r_M - r_f)/\sigma_M$. Note, that we assume a risk-free return of $r_f = 4\%$. The hit rate counts how often the sign of the relative change in the German bond is correctly predicted. The realized potential is defined as the ratio of the accumulated model return to the maximum possible accumulated return. The related trading strategy is simple: If we expect a positive shift in the German bond rate, we take a long-position in the asset. Negative expected returns are executed as short-positions. This trading strategy is also used for the computation of the profit & loss curve. Finally, the annualized return of investment is the average yearly profit of the outlined trading strategy. All performance measures are calculated only on the out-of-sample period.

The empirical results concerning the performance of the neural networks are summarized in Tab. 3.1.

As shown in Tab. 3.1, the 11-layer architecture outperforms the standard MLP during the out-of-sample period. This general statement is valid for *all* calculated performance measures. For example, the annualized return of the 11-layer architecture is 11.81% on the test set, while the MLP achieves only 4.32%. The profit & loss curves of both neural networks are depicted in Fig. 3.8.

Remarkably, we found that both neural networks encountered difficulties in the first part of the generalization set (June 1990 to June 1993). This time period is related to the German re-unification. We suppose that during this time period the underlying dynamics of the German bond market was somewhat disturbed. In the second half of the gen-

Table 3.1 Performance evaluation based on Sharpe ratio, hit rate, realized potential and annualized return of investment. All calculations are performed on the generalization set.

model	**Sharpe ratio**	**hit rate**	**realized potential**	**annualized return**
11-layer	0.7395	84.41%	74.68%	11.81%
MLP	0.3841	63.01%	44.74%	4.32%

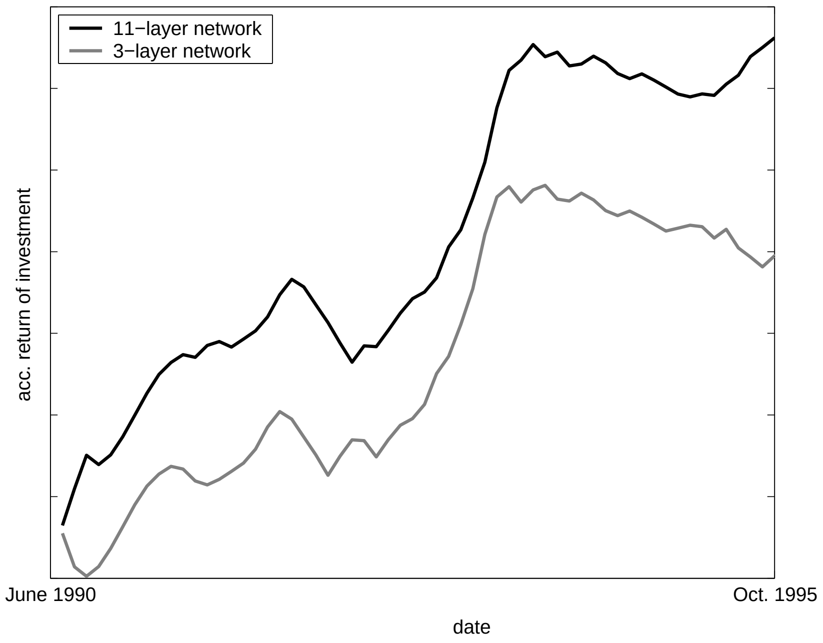


Figure 3.8 Profit & loss curves of the 11-layer architecture and the 3-layer feedforward neural network. The comparison is drawn on the generalization set.

eralization set (July 1993 to Oct. 1995), the dynamics of the German bond market recovered. During this time span, the 11-layer architecture allows a good fitting of the market dynamics, while the forecasts of the MLP are more unreliable [NZ98, p. 419].

7. PRELIMINARY CONCLUSION

Model building on the basis of standard feedforward neural networks typically relies on the well-known theorem, that a sufficiently large 3-layer network can in principle approximate any continuous functional relationship [HSW92, p. 14-20]. As a consequence of this philosophy, the characteristics and quality of the underlying data constrain the performance of the model. If the data is covered with noise or does *not* contain all of the required information, a mainly data dependent model building is clearly disadvantaged. In addition, the functional approximation abilities of feedforward networks often induce overfitting. Overfitting is a critical issue, because the neural network loses its generalization abilities due to learning by heart. In other words, the neural network does *not* only learn the underlying structure of the time series, but also the noise included in the data [Zim94, p. 58-9].

In the face of these problems, we believe that a model building process should incorporate prior knowledge about the specific application. The learning from data is only *one* part of this process, since *not* all information about the underlying dynamics may be contained in the data or the data is often unreliable due to noise, outliers or missing values [NZ98, p. 373]. The 11-layer network is one example of this model building philosophy. Prior knowledge about the specific application is introduced into the 11-layer network by architectural building blocks. For example, an interaction layer allows the modeling of inter-market relationships, while the concept of forces and embeddings provides deeper insights into the underlying dynamics. As a consequence, the model building is not only based on the (unreliable) training data but also on prior knowledge about the underlying dynamic system. In the next chapter, we present time-delay recurrent neural networks that also incorporate prior knowledge in form of architectural enhancements [ZN01, ZNG01a].

The benefits of an integrated model building approach are illustrated by a comparison of the 11-layer architecture and a standard 3-layer feedforward neural network. Experimental results from the German bond market indicate, that the 11-layer network is highly superior the standard MLP.

Remarkably, a model building process, which is based on both time series data and prior knowledge about the application, does *not* only provide superior forecasts, but also allows to gain a deeper understanding of the application. On this basis, it is also possible to analyze and to quantify the uncertainty of the model forecasts. This is especially important for the development of decision support systems [NZ98, p. 386].

Chapter 4

MODELING DYNAMICAL SYSTEMS BY RECURRENT NEURAL NETWORKS

In this chapter we mainly focus on a new recurrent neural network architecture, which includes the previous model error as an additional input. Hence, the learning can interpret the observed model error as an external shock which can be used to guide the model dynamics afterwards.

The description of a dynamic by a recurrent system allows the inclusion of memory effects. A natural representation of such a recursion in form of a neural network is given in the first section (sec. 1.1) [Hay94, p. 735-6]. In this context, the major difficulty is the identification of such a system, which is a parameter optimization task. Instead of solving the latter problem by employing an optimization algorithm we use finite unfolding in time to transform the temporal identification task into a spatial architecture, which can be handled by a shared weights extension of error backpropagation (sec. 1.2) [RHW86, p. 354-57].

Most real world systems are open systems, i. e. they are partly autonomous and partly externally driven. A more detailed study of the learning behavior of neural networks identifying such systems shows that the network learning has a bias on the external driven part. In order to enforce the learning to emphasize on the autoregressive part of the dynamics, we introduce an extension of the unfolding in time architecture, called overshooting (sec. 1.3) [ZN01, p. 326-7]. In addition, overshooting allows improvements in the estimation of the embedding dimension and supports long term forecasting (sec. 1.4). Compared to feedforward neural networks, the identification of relevant inputs is even more complex in case of a recurrent model framework, since delayed influences are caused by the memory of the model. On this problem section 1.5 deals with input feature selection for recurrent networks.

In the second section (sec. 2.) we introduce the concept of error correction neural networks (ECNN). The approach of using observed model errors to improve the identification of a dynamic system is also utilized by other methods, e. g. the parameterization of a Kalman Filter varies with the observed model error, while linear ARIMA models include autoregressive components in combination with a stochastic error correction part [Hay96, Wei90, p. 302-22 and p. 71]. In contrast, our approach consists of a fixed nonlinear autoregressive and error correction system (sec. 2.1).

In section 2.2 the unfolding in time of a related network architecture is presented. This design prevents the ECNN from mapping pseudo causalities caused by external random shocks. Once again, a combination with overshooting improves the long term predictability (sec. 2.3). Furthermore we propose to handle the problem of trend following models by integrating the concept of alternating errors into the ECNN framework (sec. 2.4).

Section 3. consists of two parts: First we introduce the concept of uniform causality, which is the basis for the embedding of discretely measured time systems [ZN01, p. 330-2]. We show that a refinement of the model time grid relative to a wider-meshed time grid of the data provides deeper insights into the dynamics. Undershooting is a recurrent neural network based approach to the principle of uniform causality. Second, we combine the undershooting concept with error correction neural networks [ZNG02, p. 398-9]. This leads to a novel approach which improves the performance of our models by time grid refinements.

In section 4. we discuss an approach to handle the identification of high dimensional dynamic systems. The complexity of such tasks can be reduced significantly if it is possible to separate the dynamics in variants and invariants. Clearly, we only have to predict the variants of the system, because the invariants remain constant over time. A recombination of the variants and invariants provides us with a forecast of the complete system. In sec. 4.1 a neural network solution of the latter problem is described which searches for an optimal coordinate transformation to reduce the dimensionality of the forecasting task [RHW86, p. 335-9]. Dimensionality reduction combined with ECNN is discussed in sec. 4.2.

Section 5. deals with the problem of optimal state space reconstruction [CEFG91, ZN00a, Hay94, p. 57-72, p. 259-63 and p. 714-8]. Starting from a Takens embedding we try to define a time independent, nonlinear coordinate transformation of the state space, which allows us to proceed with a smoother trajectory. In particular, section 5.1 introduces such transformations in the form of neural network architectures. The smoothness of the trajectory is enforced by a penalty function (sec. 5.2). In section 5.3 we show that the combination of the smoothing coordinate transformation and the ECNN can be done in a surprisingly elegant way.

Finally, we present an empirical study modeling the US-Dollar (USD) / German Mark (DEM) FX-market (sec. 6.). We apply the recurrent neural networks described in this chapter to forecast the monthly development of the USD / DEM FX-rate. The empirical study has two research directions: First, we examine the behavior of the different architectural building blocks and point to their contribution to the overall model performance. Second, we present a benchmark comparison. We evaluate the performance of our recurrent neural networks by a comparison with several benchmarks (e. g. linear regression model or 3-layer feedforward network). The benchmark comparison is based on common performance measures (e. g. Sharpe Ratio, realized potential or hit rate).

1. BASIC TIME-DELAY RECURRENT NEURAL NETWORKS

The following set of equations (Eq. 4.1), consisting of a state and an output equation, is a recurrent description of a dynamic system in a very general form for discrete time grids [Hay94, Kol01, p. 641-3, 735-6 and p. 57-8]. The dynamic system is depicted in Fig. 4.1 [ZN01, p. 321].

$$\begin{array}{rcll} s_t &=& f(s_{t-1}, u_t) & \text{state transition,} \\ y_t &=& g(s_t) & \text{output equation.} \end{array} \tag{4.1}$$

The state transition is a mapping from the previous internal hidden state of the system s_{t-1} and the influence of external inputs u_t to the new state s_t. The output equation computes the observable output vector y_t.

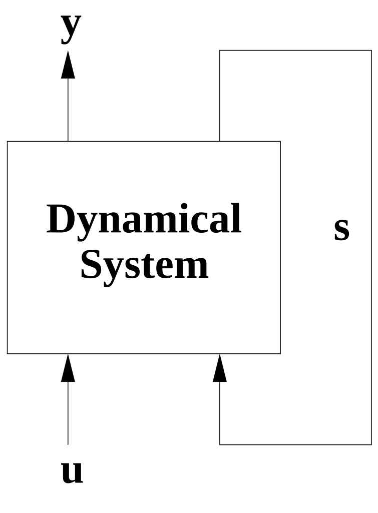


Figure 4.1 The identification of a dynamic system using a discrete time description: input $u_t \in \mathbb{R}^k$, hidden states $s_t \in \mathbb{R}^d$, output $y_t \in \mathbb{R}^n$.

The system can be viewed as a partially observable autoregressive dynamic state s_t which is also driven by external disturbances u_t. Without the external inputs the system is called an autonomous system [MC01, Hay94, p. 44-5 and p. 666]. We will refer to this special case later in order to introduce new mathematical concepts. However, in reality most systems are driven by a superposition of autonomous development and external influences [Bar97, p. 1-6].

The task of identifying the dynamic system of Eq. 4.1 can then be stated as the task to find (parameterized) functions f, g such that an averaged distance measurement (e. g. Eq. 4.2) between the observed data $y_t^d, t = 1, \ldots, T$ and the computed data y_t of the model is minimal (for other cost functions see [NZ98, p. 387-8]):

$$\frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2 \rightarrow \min_{f,g} \ . \tag{4.2}$$

If we ignore the dependency on s_{t-1} in the state transition equation by assuming $s_t = f(u_t)$, and $y_t = g(s_t) = g(f(u_t))$, we are back in the framework where a neural network approach without a recurrence, i. e. a feedforward neural network, can solve the system identification task.

The inclusion of the internal hidden dynamics makes the modeling task much harder to solve, because it allows varying inter-temporal dependencies. Theoretically, in the recurrent framework an event s_t is explained by a superposition of external shocks $\{u_t, u_{t-1}, \ldots\}$ (also referred to as external inputs) from all the previous time steps [Bar97, MC01, Hay94, p. 254-8, p. 44-45 and p. 641-3, 735-6]. In practice recurrent neural networks tend to focus as much as possible on only the most recent external variables. However, to construct a successful forecast system we want to force the networks to learn a strong autonomous sub-dynamic because only this enables us to predict how the system will evolve in the future [Kol01, CK01, p. 57-61 and p. 15-9]. Thus, sections 1.3 and 1.4 deal with techniques how to separate the autonomous dynamics of partially externally driven systems.

1.1 REPRESENTING DYNAMIC SYSTEMS BY RECURRENT NETWORKS

The identification task of Eq. 4.1 and 4.2 can be implemented as a *time-delay recurrent neural network* [ZN01, Hay94, p. 322-3 and p. 739-47]:

$$\begin{array}{rcll} s_t & = & \mathrm{NN}(s_{t-1}, u_t; v) & \text{state transition,} \\ y_t & = & \mathrm{NN}(s_t; w) & \text{output equation.} \end{array} \tag{4.3}$$

By specifying the functions f and g as neural networks with parameter vectors v and w we have transformed the system identification task of Eq. 4.2 into a parameter optimization problem (see chp. 5):

$$\frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2 \rightarrow \min_{v,w} \ . \tag{4.4}$$

The dynamic system described in Eq. 4.5 can be modeled by a neural network architecture as shown in Fig. 4.2 [CK01, ZN01, p. 21-2 and p. 322-3].

$$\begin{array}{rcll} s_t & = & \tanh(As_{t-1} + Bu_t) & \text{state transition,} \\ y_t & = & Cs_t & \text{output equation.} \end{array} \tag{4.5}$$

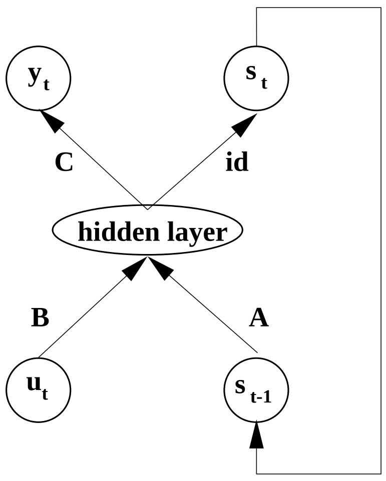


Figure 4.2 A time-delay recurrent neural network. Note, that the cluster s_t is usually not necessary, but included here for visual clarity.

The weights are $v = \{A, B\}$ and $w = \{C\}$. In general one may think of a matrix D instead of the identity matrix (id) between the hidden layer and s_t. This is not necessary because for a linear output layer s_t we can combine matrices A and D to a new matrix A' between the input s_{t-1} and the hidden layer. Note, that the output equation $\mathrm{NN}(s_t; w)$ is realized as a linear function. It is straightforward to show by using an augmented inner state vector that this is not a functional restriction [ZN01, p. 322-3].

After these definitions the next section describes a specific neural network implementation of time-delay recurrent networks.

1.2 FINITE UNFOLDING IN TIME

In this section we discuss an architectural solution technique for time-delay recurrent neural networks [RHW86, CK01, Hay94, p. 354-57, p. 19-20 and p. 751-6]. For an overview on algorithmic methods see Pearlmatter (2001) [Pearl01, p. 182-96] or Medsker and Jain (1999) [MJ99, p. 47-75]. We unfold the network of Fig. 4.2 over time using *shared weight matrices* A, B and C (Fig. 4.3). Shared weights share the same memory for storing their weights, i. e. the weight values are the same at each time step of the unfolding [RHW86, Hay94, p. 355 and p. 752].

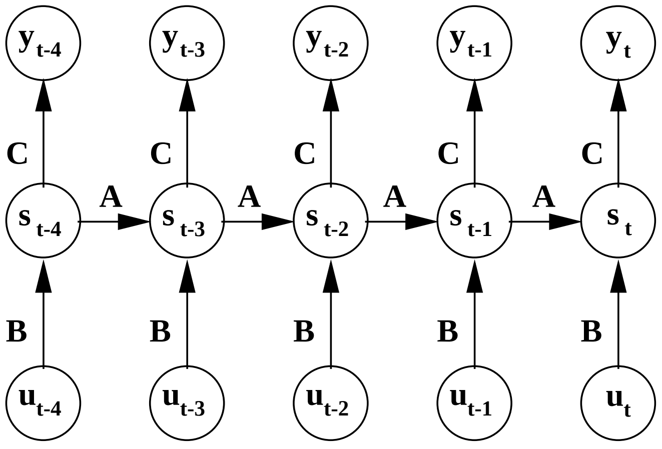


Figure 4.3 Finite unfolding realized by shared weights A, B and C.

The approximation step is the finite unfolding which truncates the unfolding after some time steps (for example, we choose $t-4$ in Fig. 4.3). The important question to solve is the determination of the correct amount of past information to include in the model of y_t [ZN01, p. 324-7]: Typically, one starts off with a given truncation length (in our example we unfold up to $t-4$). Then one can observe the individual errors of the outputs $y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}$ and y_t which are computed by Eq. 4.2. The individual error levels are usually decreasing from left y_{t-4} to right y_t. The reason for this is that the leftmost output y_{t-4} is computed only by the most past external information u_{t-4}, the next output y_{t-3} depends also on its external u_{t-3} but it uses the additional information of the previous internal state s_{t-4}. By such superposition of more and more information the error value will decrease until a minimum error is achieved. This saturation level indicates the maximum number of time steps which contribute relevant information to model the present time state.

In the following we call this number *maximal inter-temporal connectivity (MIC)* [ZN01, p. 324]. For example, if we assume that the error saturation takes place at output y_{t-1}, then we can skip time step $t-4$ (Fig. 4.3). If we did not find a saturation in our truncated unfolded network we would start again with a wider expansion, e. g. up to $t-6$. Assuming e. g. that the saturation takes place at $t-1$ we achieve an unfolding with memory of past time steps $t-3, \dots, t-1$ and the information of the present time step t. For a more detailed description of the estimation of the MIC, the reader is referred to chapter 5.

Having in mind, that we are ultimately interested in forecasting we should ask ourselves what the improvements of the modeling are up to now. If y_t is a description of the shift of a variable (e. g. a price shift $y_t = \ln(p_{t+1}) - \ln(p_t)$ in a financial forecast application), we predict the target variable one step ahead. Thus, the result of the model in Fig. 4.3 is only a sophisticated preprocessing of the inputs $u_{t-3}, \dots, u_t$ to generate a present time state description s_t from which the forecast y_t can be computed.

In contrast to typical feedforward neural networks whose success heavily depends on an appropriate, sometimes very complicated preprocessing, our approach only requires a simple preprocessing of the raw inputs u_τ [ZN01, p. 324-25]. We propose to calculate only the scaled momentum (relative change, see Eq. 2.7) and force (normalized curvature, see Eq. 2.8) of each raw input time series [NZ98, PDP00, p. 374-5 and p. 73]. This releases us from smoothing the data or computing first and higher order moments of the shift variables like it is done in the model building of feedforward neural networks [Zim94, p. 20-8].

A further advantage of the recurrent preprocessing (Fig. 4.3) is the moderate usage of free parameters. In a feed-forward neural network an expansion of the delay structure increases automatically the number of weights. In the recurrent formulation (Fig. 4.3) the shared matrices A, B are reused if more delayed input information is needed. Additionally, if weights are shared more often, then more gradient information is available for the learning of these weights. As a consequence, potential over-fitting is not so dangerous as in the training of feed-forward networks. In other words: due to the inclusion of the temporal structure into the network architecture, our approach is applicable to tasks where only a small training set is available [ZN01, p. 325].

1.3 OVERSHOOTING

An obvious generalization of the network in Fig. 4.3 is the extension of the autonomous recurrence in future direction $t+1, t+2, \ldots$. We call this extension *overshooting* (see Fig. 4.4) [ZN01, p. 326-7]. If this leads to good predictions we get as an output a whole sequence of forecasts. This is especially interesting for decision support systems, e. g. trading system in finance.

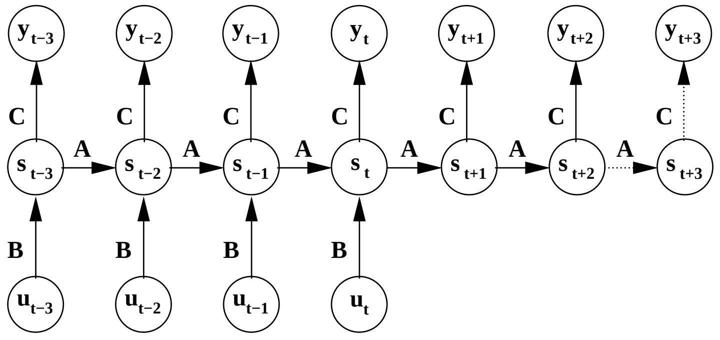


Figure 4.4 Overshooting is the extension of the autonomous part of the dynamics.

In the following we show how overshooting can be realized and we will analyze its properties. First, we discuss how far into the future good predictions can be achieved. It is apparent that the maximum number of autonomous iterations is bounded by the maximum intertemporal dependency length *MIC* which we have found by observing the error saturation as described in section 1.2 [ZN01, p. 326-7].

As long as the number of future outputs y_{t+k} does *not* exceed the *MIC* we iterate the following [ZN01, p. 326-7]: train the model until convergence, if it is over-fitting, then include the next output (here y_{t+k+1}) and train it again. Typically, we observe the following interesting phenomenon [ZN01, p. 326-7]: if the new prediction y_{t+k+1} can be learned by the network, the training error of the newly activated time horizon will decrease. In addition, the error level of all the other outputs $y_t, \ldots, y_{t+k}$ will decrease too, because more useful information (error signals) is propagated back to the weights. We stop extending the forecast horizon if newly added predictions y_{t+k+1} cannot be learned or the training error level even starts to increase. We measure the errors of the different forecast horizons on the training set. All forecast horizons refer to the same error function (e. g. Eq. 4.4). Further details on the truncation length of the overshooting can be found in chapter 5.

Depending on the problem, the *MIC* can be larger than the forecast horizon found by this algorithm [ZN01, p. 326-7]. Note, that this extension does not add new parameters, since shared weights A, C are used. Summarizing, it should be annotated, that overshooting generates additional valuable forecast information about the analyzed dynamical system and acts as a regularization method for the learning.

One may argue, that the overshooting network bears a resemblance to so-called vector autoregressive (VAR) models [Wei90, p. 342-3]. However, it is important to note, that the overshooting network is a state space model, which aggregates information over the state transitions (Eq. 4.5). Furthermore, the overshooting network is a *nonlinear* autoregressive system, while VAR-models include only *linear* components.

1.4 EMBEDDING OF OPEN SYSTEMS

The most important property of the overshooting network (Fig. 4.4) is the concatenation of an input driven system and an autonomous system. One may argue that the unfolding in time network (Fig. 4.3) already consists of recurrent functions and that this recurrent structure has the same modeling effect as the overshooting network. This is definitely not true because the learning algorithm leads to different models for each of these architectures. Backpropagation learning usually tries to model the relationship between the most recent inputs and the output because the fastest adaptation takes place in the shortest path between input and output. Thus, the learning mainly focuses on u_t. Only later in the training process, learning also may extract useful information from input vectors $u_{t-\tau}$ which are more distant to the output. As a consequence the unfolding in time network (Fig. 4.3) tries to rely as much as possible on the part of the dynamics which is driven by the most recent inputs $u_t, \ldots, u_{t-\tau}$. In contrast, the overshooting network (Fig. 4.4) forces the learning through the additional future outputs $y_{t+1}, \ldots, y_{t+\tau}$ to focus on modeling an internal autonomous dynamic. Overshooting therefore allows us to extend the forecast horizon [ZN01, p. 327-8].

The dimension of the internal state space s_t (Eq. 4.1) represents the *embedding dimension* of an autonomous dynamic system [MC01, p. 74-5]. An unfolding in time neural network is able to extract the correct size of the embedding dimension using node pruning techniques [NZ98, Zim94, p. 405 and p. 74-6]. The embedding dimension of a partially autonomous system is typically underestimated by the network architecture depicted in Fig. 4.3. In contrast, the overshooting network shown in Fig. 4.4 estimates the correct dimension, because it is forced to learn long term inter-temporal dependencies in order improve to the autonomous part of the dynamics [ZN01, p. 327-8].

1.5 CAUSALITY & FEATURE SELECTION

The advantages of the overshooting network are especially important for the analysis of causal relationships. Let us characterize causality in form of a reasoning as an 'If-Causality'. Typical examples of such a reasoning can be found in rule based systems (incl. fuzzy logic) or causal graphical networks. On the other hand, a causality in form of a reasoning combined with a temporal delay structure can be called 'When-Causality'. This type is mainly used in control and recurrent systems. Both approaches are sensitive against pseudo correlations [SC00, Bar97, RN95, p. 430-6, p. 255 and p. 209,448].

However, we see a typical difference between human views on causality and the approaches mentioned above: Most of the time, people are considering the causality of events rather than the causality of variables [LN77, p. 501-2]. Especially in the field of financial forecasting, turning points of the time series are of major interest. Instead of modeling an 'Event-Causality' we have to transform the time series of interest into a series of events. Afterwards, we can sort out the relevant causalities by the application of an overshooting network.

The basic idea of this concept can be outlined as follows: During the estimation of the maximal inter-temporal connectivity we search for long-term dependencies between the targets y_τ and the external influences u_τ. In other words, we try to discover the first causal connection between the external inputs and the targets, which improves the description of the dynamics in the present time state. Overshooting enforces the estimation of the MIC. – An overshooting network tries to explain the present time state by exploring external influences, which are located in early stages of the unfolding. This can be interpreted as a *first cause analysis* [ZN01, p. 328]. Based on the information obtained from the overshooting network, one can continue the model building with the original variables. Note, that this strategy does not guarantee to exclude pseudo correlations, but it decreases their influences.

From this point of view, overshooting networks can be utilized for the development of *early warning indicators*. The task is to detect serious changes in the future development of a dynamic system as soon as possible. Especially in economics, early warning indicators are vitally important for the analysis of financial markets or the evaluation of emerging countries [ZN01, p. 328].

2. ERROR CORRECTION NEURAL NETWORKS (ECNN)

If we are in the position of having a complete description of all external forces influencing a deterministic system the equations 4.6 would allow to identify the temporal relationships by setting up a memory in form of a state transition equation [MC01, Hay94, p. 44-5 and p. 641-6, 666]. Unfortunately, our knowledge about the external forces is typically incomplete or our observations might be noisy. Under such conditions learning with finite datasets leads to the construction of incorrect causalities due to learning by heart (overfitting). The generalization properties of such a model are questionable [NZ98, p. 373-4].

$$\begin{aligned} s_t &= f(s_{t-1}, u_t) \ , \\ y_t &= g(s_t) \ . \end{aligned} \tag{4.6}$$

If we are unable to identify the underlying system dynamics due to insufficient input information or unknown influences, we can refer to the observed model error at time period $t-1$, which can be interpreted as an indicator that our model is misleading. Handling the latter error information as an additional input, we extend Eq. 4.6 obtaining Eq. 4.7:

$$\begin{aligned} s_t &= f(s_{t-1}, u_t, y_{t-1} - y^d_{t-1}) \ , \\ y_t &= g(s_t) \ . \end{aligned} \tag{4.7}$$

Keep in mind, that if we have a perfect description of the underlying dynamics, the extension of Eq. 4.7 is no longer required, because the observed model error at $t-1$ would be *zero*. Hence, modeling the dynamical system one could directly refer to Eq. 4.6. In all other situations the model uses it's own error flow as a measurement of unexpected shocks.

This is similar to the MA part of a linear ARIMA model (*ARIMA* stands for *A*utoregressive *I*ntegrated *M*oving *A*verage models, which utilizes both, linear autoregressive components and stochastic moving average-components derived from the observed model error to fit a time series, see [Wei90, p. 71]). As a major difference, the error correction system is a state space model, which accumulates a memory over the state transitions. Hence, there is no need to preset a number of delayed error corrections. Another difference is that ARIMA models include only linear components, whereas our approach is nonlinear.

Our error correction system bears also some resemblance to the so-called Nonlinear AutoRegressive with eXogenous inputs recurrent networks (NARX) [NP90]. These networks "describe a system by using a nonlinear functional dependence between lagged inputs, outputs and / or prediction errors" [MC01, p. 71]. Hence, NARX models can be seen as a generalization of linear ARMA processes [Wei90, p. 56-7]. NARX models are usually approximated by recurrent neural networks, i. e. multi-layer perceptrons incorporating feedback loops from the output to the hidden layer [DO00, p. 336-7]. Dealing with NARX networks, one faces the problem of exploring long-term dependencies in the data [MJ99, p. 136]. As we will show, an error correction neural network is able to handle this issues appropriately (see sec. 2.3). For further details on NARX models, the reader is referred to Medsker and Jain (1999) [MJ99, p. 136-9] or Haykin (1994) [Hay94, p. 746-50].

Another resemblance bears to Kalman Filters where the model error is used to improve the system identification [MJ99, Hay96, p. 255 and p. 302-22]. In contrast to the online adaptation in the Kalman approach, we try to identify a fixed nonlinear system which is able to handle external shocks. By using such a system we can not evade an error when the external shock appears. Thus, our task is to find a fast adaptation to the new situation. Such a strategy should decrease the learning of false causalities by heart. This will also improve the generalization ability of our model.

2.1 APPROACHING ERROR CORRECTION NEURAL NETWORKS

A first neural network implementation of the error correction equations in Eq. 4.7 can be formulated as

$$\begin{aligned} s_t &= \tanh(As_{t-1} + Bu_t + D(Cs_{t-1} - y^d_{t-1}))\ , \\ y_t &= Cs_t\ . \end{aligned} \tag{4.8}$$

The term Cs_{t-1} recomputes the last output y_{t-1} and compares it to the observed data y^d_{t-1}. The matrix transformation D is necessary in order to adjust different dimensionalities in the state transition equation.

It is important to note, that the identification of the above model framework is ambiguously creating numerical problems, because the autoregressive structure, i. e. the dependency of s_t on s_{t-1}, could either be coded in matrix A or in DC. On this problem, one may argue to transform Eq. 4.8 into a well defined form (Eq. 4.9) utilizing $\dot{A} = A + DC$.

$$\begin{aligned} s_t &= \tanh(\dot{A}s_{t-1} + Bu_t + Dy^d_{t-1})\ , \\ y_t &= Cs_t\ . \end{aligned} \tag{4.9}$$

Eq. 4.9 is algebraic equivalent to Eq. 4.8 without taking its numerical ambiguous problems. Unfortunately, by using Eq. 4.9 we loose the explicit information provided by external shocks, since the error correction mechanism is no longer measured by a deviation around zero. Using neural networks with tanh(.) as a squashing function, our experiments indicate, that the numerics works best if the included variables fluctuate around zero, which fits best to the finite state space $(-1;1)^n$ created by the tanh(.) nonlinearity [MC01, p. 50-4].

To overcome the drawbacks of our concretizations in Eq. 4.8 and Eq. 4.9, we propose the neural network of Eq. 4.10 formulated in an error correction form, measuring the deviation of the expected value Cs_{t-1} and the observation y^d_{t-1}. The non-ambiguity is a consequence of the additional nonlinearity.

$$\begin{array}{rcl} s_t & = & \tanh(As_{t-1} + Bu_t + D\tanh(Cs_{t-1} - y^d_{t-1}))\ , \\ y_t & = & Cs_t\ . \end{array} \tag{4.10}$$

The system identification (Eq. 4.11) is a parameter optimization task adjusting the weights of the *four* matrices A, B, C, D (see chp. 5).

$$\frac{1}{T} \cdot \sum_{t=1}^{T} \left(y_t - y^d_t \right)^2 \rightarrow \min_{A,B,C,D}\ . \tag{4.11}$$

2.2 UNFOLDING IN TIME OF ERROR CORRECTION NEURAL NETWORKS

Next, we translate the formal description of Eq. 4.10 into a spatial network architecture using unfolding in time and shared weights (Fig. 4.5) [RHW86, Hay94, p. 354-57 and p. 751-6]. The resulting network architecture is called Error Correction Neural Network (ECNN).

The ECNN architecture (Fig. 4.5) is best to understood if one analyses the dependency between s_{t-1}, u_t, $z_{t-1} = Cs_{t-1} - y^d_{t-1}$ and s_t.

Interestingly, we have two types of inputs to the model: (*i.*) the external inputs u_t directly influencing the state transition s_t and (*ii.*) the targets y^d_{t-1}, whereby only the difference between the internal expected y_{t-1} and the observation y^d_{t-1} has an impact on s_t. Note, that $-Id$ is the fixed negative of an identity matrix.

This design allows an elegant handling of missing values in the series of target vectors: if there is no compensation y^d_{t-1} of the internal expected value $y_{t-1} = Cs_{t-1}$ the system automatically generates a replacement. A special case occurs at time period $t+1$: at this future point in time,

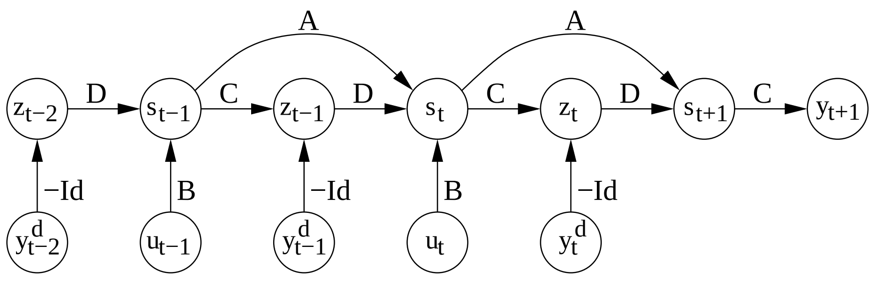


Figure 4.5 Error Correction Neural Network (ECNN)

we have no compensation y^d_{t+1} of the internal expected value, and thus the system is offering a forecast $y_{t+1} = Cs_{t+1}$. A forecast of the ECNN is based on a modeling of the recursive structure of a dynamical system (coded in A), external influences (coded in B) and the error correction mechanism which is also acting as an external input (coded in C, D).

The output clusters of the ECNN which generate error signals during the learning phase are $z_{t-\tau}$ and $y_{t+\tau}$. Have in mind, that the target values of the sequence of output clusters $z_{t-\tau}$ are *zero*, because we want to optimize the compensation mechanism $y_{t-\tau} - y^d_{t-\tau}$ between the expected value $y_{t-\tau}$ and its observation $y^d_{t-\tau}$.

Compared to the finite unfolding in time neural networks (see Fig. 4.3) [CK01, Hay94, p. 19-21 and p. 641-5, 751-5], the ECNN has another advantage: Using finite unfolding in time neural networks, we have by definition an incomplete formulation of accumulated memory in the leftmost part of the network. Thus the autoregressive modeling is handicapped. In contrast, due to the error correction, the ECNN has an explicit mechanism to handle the shock of the initializing phase.

Remind, that the ECNN shares some similarities with ARIMA processes [Wei90, p. 71], Kalman Filters [Hay96, p. 302-22] and NARX networks [MJ99, p. 136-9]. In opposite to these models, the ECNN is designed as a nonlinear state space model, which accumulates a memory over the state transitions. Furthermore, the ECNN is able to explore long-term dependencies in the data. We address this issue in the next subsection.

2.3 COMBINING OVERSHOOTING & ECNN

A combination of the basic ECNN presented in the preceding section 2.1 and the overshooting technique of section 1.3 is shown in Fig. 4.6 [ZN01, p. 326-7].

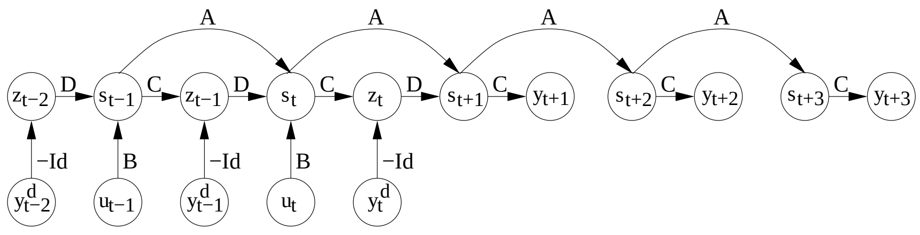


Figure 4.6 Combining Overshooting and Error Correction Neural Networks

Besides all advantages described in section 1.3, overshooting influences the learning of the ECNN in an extended way. A forecast provided by the ECNN is in general based on a modeling of the recursive structure of a dynamical system (coded in the matrix A) and the error correction mechanism which is acting as an external input (coded in C and D). Now, the overshooting enforces the autoregressive substructure allowing long term forecasts. Of course, in the overshooting environment we have to supply the additional output clusters $y_{t+1}, y_{t+2}, y_{t+3}, \ldots$ with target values. Note, that this extension has the same number of parameters as the standard ECNN in Fig. 4.5.

2.4 ALTERNATING ERRORS & ECNN

In this section we propose another extension of the basic ECNN in Fig. 4.5 called alternating errors. The inclusion of alternating errors allow us to overcome trend following models. Trend following behavior is a well-known difficulty in time series forecasting. Typically, trend following models underestimate upwarding trends and vice versa. Furthermore, trend reversals are usually predicted too late.

More formally, a trend following model can be identified by a sequence of *non-alternating* model errors $z_\tau = (y_\tau - y^d_\tau)$ [PDP00, p. 295-7]. Thus, enforcing *alternating* errors z_τ we reduce trend following tendencies. This can be achieved by adding a penalty term to the overall error function of the ECNN (Eq. 4.11):

$$\frac{1}{T} \cdot \sum_{t=1}^{T} \left(y_t - y^d_t \right)^2 + \lambda \cdot (z_t + z_{t-1})^2 \to \min_{A,B,C,D} \quad . \tag{4.12}$$

The additional penalty term in Eq. 4.12 is used to minimize the autocovariance of residual errors in order to avoid trend following behavior. Note, that solutions are *not* sensitive against the choice of λ.

Our experiments with the penalty term of Eq. 4.12 indicate, that the additional error flow is able to prevent the learning of trend following models. In Fig. 4.7 we combined the penalty term of Eq. 4.12 with the basic ECNN of Fig. 4.5.

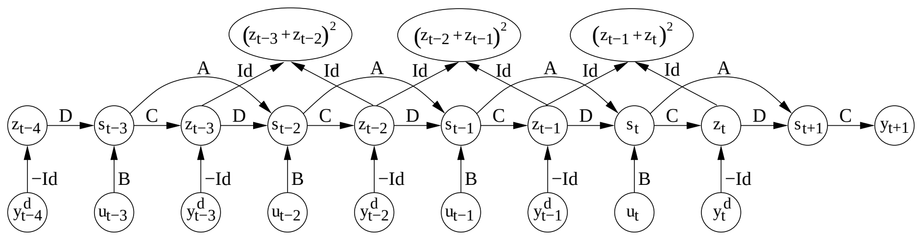


Figure 4.7 Combining Alternating Errors and ECNN. The additional output layers $(y_\tau - y^d_\tau)^2$ are used to compute the penalty term of Eq. 4.12 during the training of the network. We provide the output layers $(y_\tau - y^d_\tau)^2$ with task invariant target values of 0 and apply the mean square error function (Eq. 5.1, [PDP00, p. 413-4]), because we want to minimize the autocovariance of the residuals. Note, that we can omit the calculation of the alternating errors in the recall (testing) phase of the network.

The integration of alternating errors into the ECNN is natural: The error correction mechanism of the ECNN provides the model error $z_\tau = (y_\tau - y^d_\tau)$ at each time step of the unfolding, since the model error is required by the ECNN as an additional input. Thus, we connect the output clusters z_τ in pairs to another output cluster, which uses a squared error function. This is done by using fixed identity matrices *id*. Note, that the additional output layers $(y_\tau - y^d_\tau)^2$ are only required during the training of the ECNN.

Due to the initialization shock, we do not calculate a penalty term for the first pair of model errors. Note, that the proposed ECNN of Fig. 4.7 has no additional weights, because we only use already existing information of model errors at the different time steps of the unfolding.

3. UNDERSHOOTING

Time series analysis deals with building a model of a dynamical system on the basis of observed data [DO00, p. 325-7]. Typically, the time grid of the data is the same as the discretized time grid of the model. We show that a refinement of the model time grid relative to a wider-meshed time grid of the data provides deeper insights into the dynamics [ZN01, p. 328-30]. A recurrent neural network approach called *undershooting* can be derived from the principle of uniform causality [ZN01, p. 330-32]. As a special characteristic, we combine undershooting and time-delay recurrent error correction neural networks.

3.1 UNIFORM CAUSALITY

Uniform Causality provides the basics for the embedding of time discrete systems. Using this principle, we are able to forecast on a finer time grid than that of the data [ZN01, p. 328].

First, let us assume, that we have to identify an autonomous model for a given time series $s_0, s_1, s_2, \cdots, s_T$. The most obvious approach is to build a discrete time model where the time grid of the model is equal to the grid of the data [Hay94, p. 666-7]:

$$s_{n+1} = f(s_n)\,. \tag{4.13}$$

By iteration of f we obtain the flow F

$$s_n = F(s_0, n) := \underbrace{f \circ \cdots \circ f}_{n}(s_0) \quad \text{for} \quad n \in \mathbb{N}\,. \tag{4.14}$$

It is a trivial exercise to work with a wider-meshed time grid for the model, e. g. to use daily data in order to develop a weekly forecast model. Here, we reflect on the consequences of a refinement of the model time grid relative to the time grid of the data, e. g. to build a weekly model from monthly data. In a first shot this may be understood as an interpolation between the discretely measured values s_n [ZN01, p. 328].

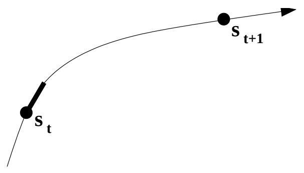


Figure 4.8 Search for a continuous embedding of a discretely measured dynamic system.

There are many possible interpolation schemes. Typically, one uses smooth interpolation techniques, e. g. splines, to find a smooth trajectory along the data points [ZN01, p. 328]. Since we want to analyze dynamic systems we now introduce the principle of uniform causality (Eq. 4.15) in order to further constrain the set of possible interpolations (Fig. 4.8) [ZN01, p. 328-9].

For each m dimensional real vector $s \in \mathbb{R}^m$ and $t, t_1, t_2 \in \mathbb{R}_+$, the *principle of uniform causality* is given by

$$\begin{aligned} \textit{embedding:} \quad && s_t &= f^t(s)\,, \\ \textit{additivity:} \quad && f^{t_1+t_2}(s) &= f^{t_2}\left(f^{t_1}(s)\right)\,. \end{aligned} \tag{4.15}$$

The embedding of Eq. 4.15 can be satisfied by any continuous interpolation. It is formulated as a continuous iteration [KCG90]. The additivity (Eq. 4.15) is a description of *causality*. Intending to follow a dynamics over t_1+t_2, we can start at s and track the system to $s_{t_1} = f^{t_1}(s)$. Then, we begin at s_{t_1} for the rest of the trajectory $s_{t_1+t_2} = f^{t_2}(s_{t_1})$ [ZN01, p. 329].

For the case of $t \in \mathbb{N}$, uniform causality (Eq. 4.15) is self-evident [ZN01, p. 329]: the embedding describes the data and the additivity is a simple iteration of functions. The necessity of $t \in \mathbb{R}_+$ is a strong additional constraint. Let us assume, that we choose f^t as

$$s_t = f^t(s) \ , \text{ with } t \in [0,\epsilon] \text{ and } 0 < \epsilon < 1. \tag{4.16}$$

No matter how small ϵ is, due to the initialization of the first part of the trajectory and due to the additivity condition (Eq. 4.15), there is only one way to follow the path in a causal way.

Modeling a dynamical system by an ordinary differential equation $\frac{ds}{dt} = f(s)$, the principle of uniform causality is always guaranteed [ZN01, p. 329]:

$$\begin{array}{lrcl} \textit{embedding:} & s_t & = & \int_0^t f(s_\tau)\, d\tau \ , \\ \textit{additivity:} & s_0 + \int_0^{t_1+t_2\ldots} & = & \left(s_0 + \int_0^{t_1\ldots}\right) + \int_{t_1}^{t_1+t_2\ldots} \ . \end{array} \tag{4.17}$$

The embedding is given by the integral of the differential equation and the causal additivity is the additivity of the integral. For details of the continuous iteration or continuous embedding of a given time discrete dynamic system see e. g. Kuczma et al. (1990) [KCG90].

We are interested in finding a dynamic law of the discrete dynamics and thus, look for a uniform causal model fitting the data [ZN01, p. 329]. In the standard approach we identify $f(s)$ by e. g. using training data such that

$$s_{n+1} = f(s_n) \ , \ n \in \mathbb{N} \ . \tag{4.18}$$

Thus, we can compute the discrete flow $(s_0, s_1, \ldots)$ by iterations of f

$$s_n = F(s_0, n) = f^n(s_0)\,. \tag{4.19}$$

Note that the standard approach only uses natural numbers as iteration indices. To introduce rational iterations we need the following property of f:

$$f^{tn}(s) = \left(f^t\right)^n(s) \ , \ n \in \mathbb{N} \ , \ t \in \mathbb{R}_+ \ , \tag{4.20}$$

which is a direct consequence of iterating the additivity condition described in Eq. 4.15.

Let us assume, that we are able to identify a function $f^{\frac{1}{q}}(s)$ by

$$s_{n+1} = \underbrace{f^{\frac{1}{q}} \circ \cdots \circ f^{\frac{1}{q}}}_{q}(s_n) \ . \tag{4.21}$$

This is an iterated system where the parameters of $f^{\frac{1}{q}}$ can be estimated by e. g. backpropagation [Hay94, p. 161-75]. We call $f^{\frac{1}{q}}$ a q-root of f [ZN01, p. 329-30]. Referring to $f^{\frac{1}{q}}$, we can identify the trajectory for every rational number to a basis of q. The embedding and additivity conditions of Eq. 4.15 are fulfilled, since operations with rational exponents can be reduced to natural exponents using property 4.20 [ZN01, p. 330]:

$$\begin{array}{ll} \textit{embedding:} & s_{\frac{p}{q}} = f^{\frac{p}{q}}(s_0) = \left(f^{\frac{1}{q}}\right)^p (s_0) \ , \\ \textit{additivity:} & f^{\frac{p_1}{q}+\frac{p_2}{q}}(s_0) = \left(f^{\frac{1}{q}}\right)^{p_1+p_2}(s_0) = f^{\frac{p_1}{q}}\left(f^{\frac{p_2}{q}}(s_0)\right) \ . \end{array} \tag{4.22}$$

By increasing q our approach would lead to more and more uniform causal solutions. Unfortunately, the identification task of Eq. 4.21 becomes also more and more difficult [ZN01, p. 330].

3.2 APPROACHING THE CONCEPT OF UNDERSHOOTING

Undershooting is the refinement of the model time grid by recurrent neural networks using unfolding in time and shared weights [RHW86, p. 354-7]. The concept of undershooting is directly related to the preceding considerations on uniform causality [ZN01, p. 330].

Let us consider the undershooting neural networks depicted in Fig. 4.9 [ZN01, p. 331]. Suppose, that the task of the networks is to forecast a price shift $\ln(p_{t+1}/p_t)$ one step ahead.

For simplicity, delayed inputs are ignored, i. e. the input is only given by u_t. The input u_t is transferred to the recurrent network by matrix B. The network's output is computed by matrix C and compared to the targets $\ln(p_{t+(k+1)/4}/p_{t+k/4})$, $k = 0, \ldots, 3$ in order to generate an error flow. The recursive structure of the dynamics is coded in matrix A.

If we introduce *three* intermediate time steps ($q = 4$) in the description of the dynamics, we get the architecture of Fig. 4.9, left. This network cannot be trained, because none of the intermediate targets

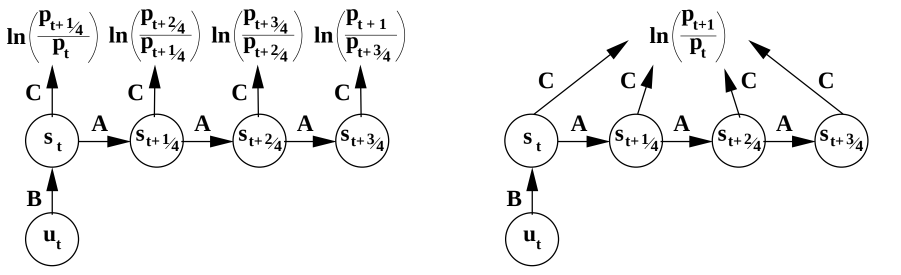


Figure 4.9 Since the intermediate targets $\ln(p_{t+(k+1)/4}/p_{t+k/4})$, $k = 0, \cdots, 3$, are not available, the left-hand network cannot be trained. The transformation, right, allows to train the network even if the intermediate targets are not available. We use the target $\ln(p_{t+1}/p_t)$ for all intermediate states [ZN01, p. 331].

$\ln(p_{t+(k+1)/4}/p_{t+k/4})$, $k = 0, \ldots, 3$ is available. We solve this problem by exploiting the identity of

$$\ln\left(\frac{p_{t+1}}{p_t}\right) = \sum_{k=0}^{3} \ln\left(\frac{p_{t+(k+1)/4}}{p_{t+k/4}}\right). \tag{4.23}$$

This leads directly to the redesigned architecture of Fig. 4.9, right.

In such a recurrent interpolation scheme (Fig. 4.9), there is in principle no need to use shared weights A, C. However, by the application of shared weights, we introduce an important regularization property: For all sub-intervals we assume the same underlying dynamics. Another advantage is that multiple gradient information for each shared weight are generated (see chp. 5). This enables us to estimate our models on the basis of very small data sets [ZN01, p. 331-2].

A successful application of the undershooting concept is reported in Zimmermann et al. (2002) [ZNG02, p. 397-8]. The authors forecast the annual development of business rentals in Munich from 1982 to 1996. The applied undershooting neural network is able to fit the rental dynamics more accurately than a preset benchmark (recurrent network without undershooting).

3.3 COMBINING ECNN AND UNDERSHOOTING

In this subsection we combine the basic ECNN depicted in Fig. 4.5 with the principle of undershooting. According to our experiments, the ECNN is appropriate for the modeling of dynamical systems in the presence of external shocks or noise. The additional regularization of the ECNN by undershooting should enable us to improve the forecast performance (see chp. 6.). The combined neural network architecture is depicted in Fig. 4.10.

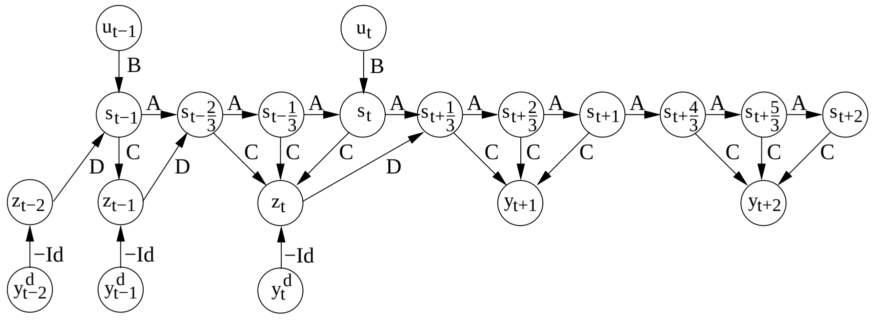


Figure 4.10 Combining Error Correction Neural Networks and Undershooting.

As depicted in Fig. 4.6, undershooting is integrated into the ECNN by a redesign of the autonomous substructure. Assuming a uniform structure in time, the underlying system dynamics of the ECNN is divided into autonomous sub-dynamics. For example, in Fig. 4.10, we added *two* intermediate states $s_{\tau+1/3}$, $s_{\tau+2/3}$. The output of the undershooting ECNN is computed by gathering the information of the intermediate states.

4. VARIANTS-INVARIANTS SEPARATION

In this section we deal with the modeling of high-dimensional dynamical systems [ZN01, p. 341]. The prediction of such a system can be simplified, if it is possible to separate the dynamics into time variant and invariant structures. In this case, the dimensionality of the time variants should be lower than that of the complete dynamic system.

As an introductive example for the separation of variants and invariants, let us think about how to model a pendulum (Fig. 4.11) [ZN01, p. 341].

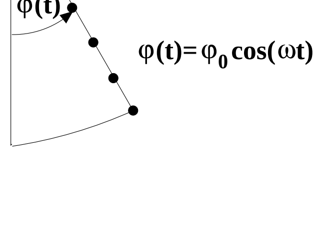


Figure 4.11 The dynamics of a pendulum can be separated in only one variant, the angle φ, and in infinite number of invariants.

A complete description of the pendulum dynamics has to state the coordinates of all the points along the pendulum line, an infinite number of values, and their temporal shifts. Nobody would solve the modeling of a pendulum in the preceding way. Instead, we can significantly simplify the description by a separation of the dynamics in only one variant (the angle φ) and by a description of the invariants (all points of the pendulum line) [ZN01, p. 341].

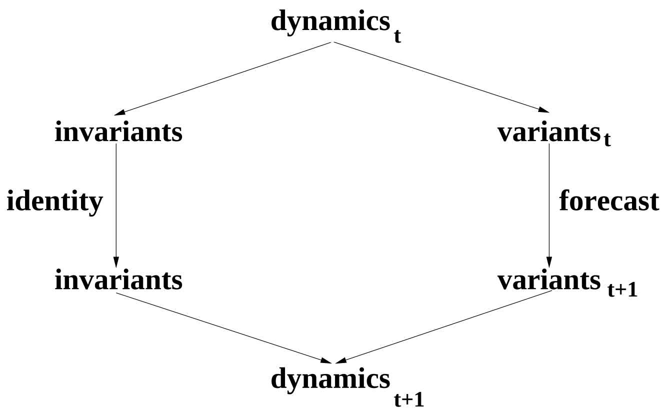


Figure 4.12 Variants-invariants separation of a dynamics.

We learn from this exercise that the separation of time variants and invariants may allow us to simplify the modeling of high-dimensional dynamic systems (Fig. 4.12). – The complexity of such systems can be reduced, if it is possible to separate the dynamics into time variant and invariant structures. Clearly, only the variants have to be forecasted, while the invariants remain constant over time. The recombination of predicted variants and unchanged invariants provides us with a forecast of the high-dimensional dynamics [ZN01, p. 341].

There are many economical applications where this idea can be exploited. An example is the forecasting of yield curves where we can compress the interest rates for different maturities to a small number of variants (in case of the German yield curve only 3 variants are important [ZNG00c, p. 266]). In the broadest sense, the variants are comparable with the principle components of the yield curve dynamics. Since the yield curve can be reconstructed from the variants and invariants, we only have to forecast the variants in order to predict the complete yield curve development [ZNG00c, p. 265-7]. Another example is tactical portfolio management. Instead of forecasting every individual stock, we compress the task by e. g. only predicting market sectors [ZN01, p. 341].

4.1 VARIANTS-INVARIANTS SEPARATION BY NEURAL NETWORKS

We propose a suitable coordinate transformation to separate time variant from invariant structures [RHW86, Spe00, CS97, ZN01, p. 335-9, p. 121-4, p. 101-7 and p. 341-2]. The coordinate transformation allows us to translate the high-dimensional space of observed state representations Y into a low-dimensional inner state space S (i. e. $dim(Y) > dim(S)$) [ZN01, p. 341]. A coordinate transformation in form of a neural network architecture is depicted in Fig. 4.13 [ZN01, p. 342].

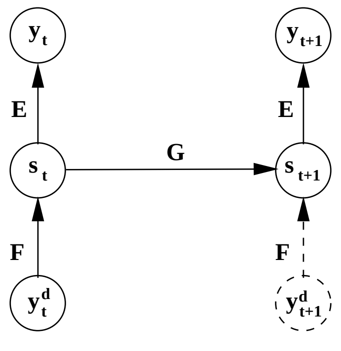


Figure 4.13 Variants-invariants separation by neural networks

Consider a single time step t of the unfolding in time network architecture shown in Fig. 4.13. The internal state s_t plays the role of the variants. The compression matrix F separates the observed state representations y^d_t into time variant and invariant structures ($dim(Y) > dim(S)$). The decompression matrix E combines the variants with the unchanged invariants to reconstruct the complete dynamics y_t.

The matrix G is used to forecast the internal state variables s_t. Remarkably, we forecast only the smaller number of identified variants. At the next time step of the unfolding ($t + 1$), the predicted state variables s_{t+1} (variants) are recombined with the unchanged invariants using the decompression matrix E. By this, we get a forecast of the complete dynamics y_{t+1}. Note, that E and F are shared weight matrices. We can omit the compression phase at $t + 1$ (dotted connection in Fig. 4.13), since we have no observations y^d_{t+1} of the dynamics at future time.

The general description of the variants-invariants separation can be stated without referring to neural networks. Nevertheless the realization is a typical neural network approach due to its property to concatenate functional systems [ZN01, p. 342]. Learning is then adequately implemented by error backpropagation (chp. 5).

4.2 COMBINING VARIANTS-INVARIANTS SEPARATION & ECNN

In this section we want to integrate the dimension reduction concept described in the previous section into the basic ECNN depicted in Fig. 4.6.

One might expect that merging both networks leads to a very large interconnected structure. In Fig. 4.14 we want to propose a slightly different approach. Although the compressor-decompressor network on the left hand-side seems to be disconnected from the forecasting branch, both networks are interrelated: Since we use shared weights the two subsystems influence each other without having an explicit graphical interconnection. Therefore, the bottleneck network and the ECNN can interact to search for an optimal coordinate transformation supporting the system identification.

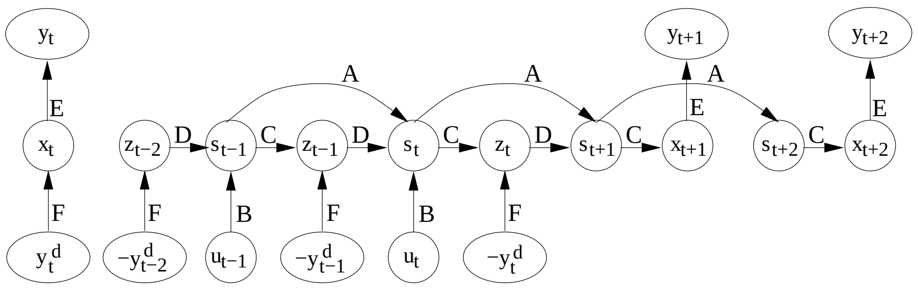


Figure 4.14 Combining the concept of variants-invariants separation with ECNN. Remind, that the observed dynamics y^d_τ is high-dimensional (see Fig. 4.13). For instance, one may refer to the different interest rates of a yield curve [ZNG00c, p. 265-7].

In contrast to the basic ECNN shown in Fig. 4.6, we have two new matrices E, F. While matrix F codes the compression, matrix E is associated with the decompression. Proceeding this way, the ECNN has to predict a coordinate transformed low dimensional vector x_t instead of the high dimensional vector y_t. It is important to note, that the ECNN requires $-y^d_t$ as input in order to generate $-x^d_t$ in the z_t layer. This allows the compensation of the internal forecasts $x_t = Cs_{t-1}$ by the transformed target data $-x^d_t = F(-y^d_t)$.

In our experience we found, that the training of the latter neural network (Fig. 4.14) is very robust, i. e. the coordinate transformation and the forecasting can be trained in parallel. Furthermore, node pruning can be applied to the middle layer of the bottleneck network (left hand-side of Fig. 4.14) [NZ98, Zim94, p. 405 and p. 74-6]. Since we use shared weights, the result of pruning in the bottleneck network is also transfered to the ECNN branch (right hand-side of Fig. 4.14).

A successful application of the variants-invariants separation is reported in Zimmermann et al. (2000c) [ZNG00c, p. 265-7]. The authors use the extended ECNN depicted in Fig. 4.14 to forecast the complete German yield curve, which is composed of 10 different interest rates. In a benchmark comparison based on a common performance measure (realized potential [Ref95, p. 71]), it turns out that the extended ECNN is able to outperform the benchmarks (naive strategy [PDP00, p. 415-6], 3-layer MLP) on different forecast horizons (monthly, quarterly and semi-annual predictions).

5. OPTIMAL STATE SPACE RECONSTRUCTION FOR FORECASTING

In section 1. we described the modeling of a dynamic system by unfolding in time. Now, we introduce the concept of *unfolding in space* **and** *time* [ZN00a, ZN01, p. 259-64 and p. 342-50]. The proposed neural network architecture implements a state transformation of a dynamic system. The aim is to specify this transformation such that the related forecast problem becomes easier because the transformed system evolves more smoothly over time. We achieve this by integrating state space reconstruction and forecasting in a neural network approach.

Let z_t, with $t = 1, \ldots, T$, be a sequence of observations generated by an autonomous dynamic system. According to Takens' theorem [Hay94, Tak81, p. 714-6], the unobserved state of the system can be reconstructed by a finite vector $x_t = (z_t, z_{t-1}, \ldots, z_{t-m})$ if m is sufficiently large, leading to the system description $x_{t+1} = F(x_t)$. But why should the reconstruction result in an appropriate formulation for the transformation function $F(\cdot)$?

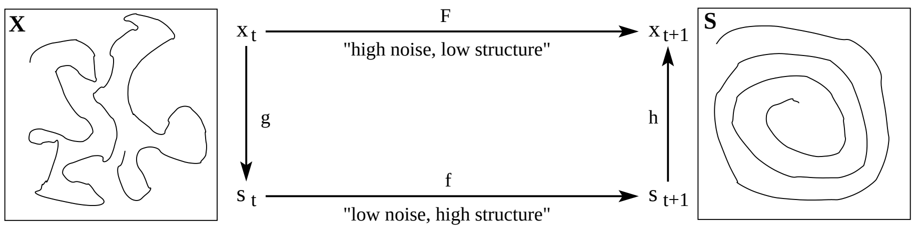


Figure 4.15 The time series of the observed state description x_t may follow a very complex trajectory. A transformation to a possible higher-dimensional state space may result in a smoother trajectory.

There are many papers on optimal state space reconstruction, e. g. Packard et al. (1980) [PCFS80], Casdagli et al. (1991) [CEFG91], Gibson et al. (1992) [GFC92], Ott and Sauer (1994) [OSY94] or Sauer et al. (1991) [SYC91]. In contrast to this work we evaluate the reconstruction with respect to its forecast abilities [ZN00a, p. 259-61].

To achieve a useful reconstruction one may transform the state space (see its complex trajectory in Fig. 4.15, left) into a new (possibly higher dimensional) space hoping that the state transition will become smoother and thus make the identification task $s_{t+1} = f(s_t)$ easier (Fig. 4.15, right). Note, that in the decomposition of F

$$x_{t+1} = F(x_t) \quad \Rightarrow \quad x_{t+1} = h \circ f \circ g(x_t) \tag{4.24}$$

the transformations h, g are static mappings while f contains the dynamics over time for the new state space.

The system identification task can now be stated in two phases [ZN00a, p. 260]: First, search for an unfolding procedure $g : X \to S$ which transforms the original state space X into the higher-dimensional space S with smoother trajectories, and a folding $h : S \to X$ for the transformation back to X such that

$$\begin{aligned} \|x_t - h \circ g(x_t)\| &\to \min_{g,h} \\ \text{subject to} \quad s_t = g(x_t), t &= 1, \ldots, T \text{ is `smooth'} \ . \end{aligned} \tag{4.25}$$

As a constraint we require, that the transformed dynamic system s_t, with $t = 1, \ldots, T$, which results from $s_t = g(x_t)$, is smooth over time.

Second, after having learned the appropriate transformations g and h, the new system dynamics $f : S \to S$ has to be determined such that

$$\|x_{t+1} - h \circ f \circ g(x_t)\| \to \min_{f} \ . \tag{4.26}$$

Instead of solving the difficult problem of identifying F in Eq. 4.24 directly, we divide the task into two subproblems, namely an identification of static mappings g, h (phase 1), and, using these coordinate transformations, an identification of the dynamics f (phase 2). We hope that this separation simplifies the problem because it allows us to optimize in a state description whose trajectories have lower noise and higher structure (as indicated in Fig. 4.15).

5.1 FINITE UNFOLDING IN SPACE & TIME

The search for the static transformations g and h can be easily realized by an auto-associator network as shown in Fig. 4.16.

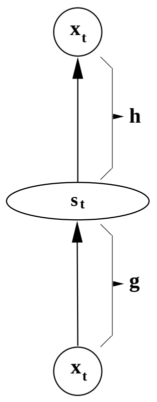


Figure 4.16 The function g maps the observed states x_t to an inner state s_t, h is a coordinate transformation back to the observation x_t. In the following, g and h are coded by shared weight matrices A and B.

In opposite to this standard approach of an auto-associator network [RHW86, p. 335-9], we are not mainly interested in a low-dimensional middle layer which acts as a compression-decompression network. Instead, we want to implement a state transformation, in order to favor the internal time series s_t to be *smooth* [ZN00a, p. 261-3]. Smoothness is a property which is defined over neighboring time points, e. g. over s_{t-1}, s_t, s_{t+1}. This feature is controlled by the architecture in Fig. 4.17 [ZN00a, p. 261].

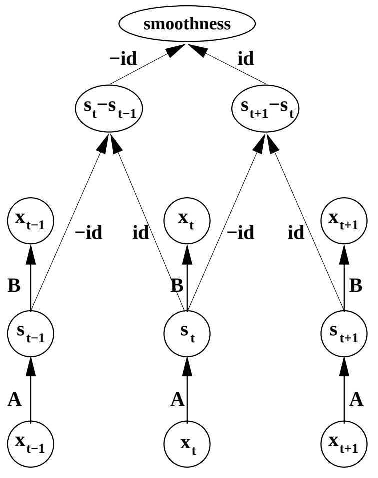


Figure 4.17 An architecture for phase 1, $\|x_t - h \circ g(x_t)\| \to \min_{g,h}$, which forces smooth transitions.

Remember, that we want to divide the problem of identifying the dynamics F by splitting the learning into two phases. The implementation of phase 1, using shared weight matrixes A and B is shown in Fig. 4.17. The (re-)transformations g, h are actually an implementation of neural network autoassociators. The multiple replication of such autoassociators allows the computation of succeeding states like s_{t-1}, s_t, s_{t+1}. Smoothness can then be enforced by superposing a penalty function on the differences $s_t - s_{t-1}$ and $s_{t+1} - s_t$ such that the overall error function consists of two terms, namely the identification error of the auto-associator and the smoothness penalty (see below) [ZN00a, p. 261].

One can assume that a solution for the coordinate transformations g, h exists if the dimension of the inner state space S is higher than or equal to the observed state space X. In this case the state representation in the middle layer is not unique but the smoothness penalty selects the preferred solution according to Eq. 4.25 [ZN00a, p. 260].

The error information coming from the smoothness condition applies to the middle layer of the network (Fig. 4.17). In phase 1 the weight matrices A and B are learned. During phase 2, A and B are fixed.

In phase 2 we learn an autonomous, potentially nonlinear dynamics f by adding a horizontal unfolding in time branch in our network architecture [ZN00a, p. 260-2].

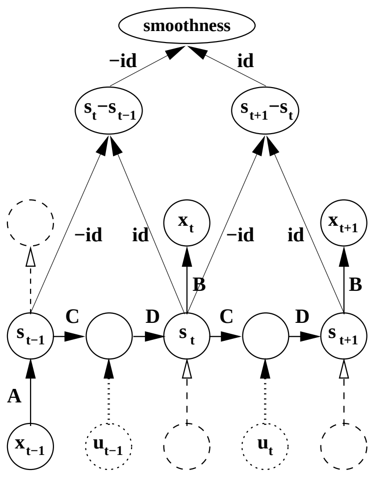


Figure 4.18 Phase 2 of the training optimizing f by $\|x_{t+1} - h \circ f \circ g(x_t)\| \to \min_f$ (see text for more details on the signal flow).

The architecture of a recurrent network in Fig. 4.18 first computes the internal state s_{t-1} using the input x_{t-1} and the fixed weight matrix A. Then we make a two step forecast from s_{t-1} over s_t to s_{t+1} using the shared weights C and D which are not fixed. The observed states x_t (x_{t+1}) are derived from s_t (s_{t+1}) using the fixed weight matrix B. Since the network is now unfolded over the 3 time points $(t-1, t, t+1)$ we can apply the smoothness penalty via the differences $s_t - s_{t-1}$ and $s_{t+1} - s_t$.

Up to now, we focused on autonomous systems. The extension to the analysis of partially externally driven systems is obvious [ZN00a, p. 260]: We have to integrate the external influences u_{t-1} and u_t into the forecasting branch of the network (Fig. 4.18, dotted marks).

5.2 SMOOTHNESS

In the following, we discuss three implementations for the smoothness penalty used for the architectures shown in Fig. 4.17 and 4.18 [ZN01, ZN00a, p. 338-41 and p. 261-3]. Let $u = s_t - s_{t-1}$ and $v = s_{t-1} - s_{t-2}$.

The first smoothness penalty has the form of a normalized curvature [ZN01, ZN00a, p. 339 and p. 261-2]:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{(u-v)^2}{u^2+v^2} \rightarrow \min \ . \tag{4.27}$$

By enforcing the numerator to be minimal, we get a curvature penalization, while the denominator prevents the trivial solution of u, v being small. Besides the smoothness of the trajectory the penalty term of Eq. 4.27 favors constant speed along the flow ($u \approx v$).

An alternative smoothness penalty can be constructed by the scalar product of the velocity vectors u and v [ZN01, p. 340]:

$$\frac{1}{T}\sum_{t=1}^{T} u \cdot v = \frac{1}{T}\sum_{t=1}^{T} \|u\| \cdot \|v\| \cdot \cos\angle(u,v) \ \rightarrow \max \ . \tag{4.28}$$

By maximizing the cos function we achieve the smoothing. If the trajectory is disturbed by noise, which results in zig-zag formations ($\angle(u,v) > 90°$), the penalty term of Eq. 4.28 slows down the motion such that the noise is suppressed.

It is apparent, that the smoothness penalty of Eq. 4.27 is inappropriate if we face zig-zag formations in the trajectory. On the other hand, the curvature penalty of Eq. 4.28 is not appropriate for smooth trajectories ($\angle(u,v) < 90°$), since the absolute velocity is increased to its maximum (only limited by tanh state space).

The following smoothness penalty tries to combine the advantages of both former penalty terms (Eq. 4.27 and Eq. 4.28) without taking their disadvantages:

$$-\frac{1}{T}\sum_{t=1}^{T}\frac{u \cdot v}{\left[u^2+v^2\right]^{(1+\cos\angle(u,v))/2}} \rightarrow \min \ . \tag{4.29}$$

In case of a smooth flow ($\angle(u,v) \approx 0°$) the above curvature penalty reduces to the normalized curvature of Eq. 4.27. Otherwise, if the trajectory is disturbed by noise, i. e. $\angle(u,v) \approx 180°$, the penalty term of Eq. 4.29 reduces to Eq. 4.28. Thus, the smoothness function of Eq. 4.28 combines the advantages of Eq. 4.27 and Eq. 4.28.

Further types of smoothness penalty functions are discussed in Zimmermann and Neuneier (2001) [ZN01, p. 338-41].

5.3 COMBINING STATE SPACE RECONSTRUCTION & ECNN

In figure 4.19 we propose a combination of the basic ECNN (Fig. 4.6) and state space reconstruction (Fig. 4.17 and 4.18).

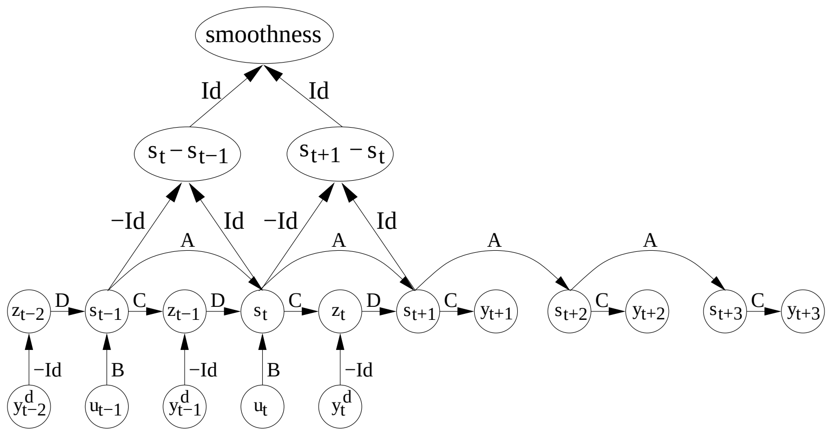


Figure 4.19 Unfolding in Space and Time by Neural Networks

Different to section 5.1 we do not start with a state vector, which is constructed with reference to Takens' Theorem [Hay94, Tak81, p. 714-6]. Instead we generate the internal state description of the underlying dynamical system by the accumulation of information in an ECNN framework. This is especially more convenient, if we have to model an open system, which is partially driven by external influences. Our effort is still concentrated on an improvement of the internal flow s_τ. Using the penalty functions discussed in section 5.2 we get an improvement of smoothness, velocity control and noise reduction.

6. EMPIRICAL STUDY: FORECASTING THE USD / DEM FX-RATE

In this section we present an empirical study modeling the US-Dollar (USD) / German Mark (DEM) FX-market. We apply the different time-delay recurrent neural networks described in this chapter to forecast the monthly change in the USD / DEM FX-rate. The performance of the time-delay recurrent networks is evaluated by a comparison with several benchmarks (e. g. linear regression model or 3-layer feedforward network). The benchmark comparison is based on common performance measures (e. g. Sharpe Ratio, realized potential or hit rate).

The goal of the study is not only to evaluate the performance of our models in contrast to the benchmarks, but also to study the benefits of the different architectural building blocks (e. g. alternating errors, undershooting or unfolding in space *and* time).

This section is composed of two parts: First, we introduce the outline of the empirical study. Thereafter, we present the empirical results modeling the USD / DEM FX-market.

6.1 OUTLINE OF THE EMPIRICAL STUDY

In the empirical study we apply the different time-delay recurrent neural networks described in this chapter to the USD / DEM FX-market. The task of the modeling is to forecast the monthly development of the USD / DEM FX-rate.

6.1.1 DATA SET

The basics of our empirical study can be characterized as follows: We are working on the basis of weekly data from Jan. 1991 to Aug. 1997 [343 data points] to predict the monthly development of the USD / DEM FX-rate. The data is divided into two subsets: the training set (in-sample period) covers the time period from Jan. 1996 to Dec. 1995 [258 data points], while the generalization set (out-of-sample period) ranges from Jan. 1996 to Aug. 1997 [85 data points].

6.1.2 INPUT SIGNALS & PREPROCESSING

As a further decision, we composed a data set of external influences, which serve as inputs for *all* our models. The used input time series are assembled in Tab. 4.1.

As it can be seen from Tab. 4.1, we use specific economic data of the USA, Japan, Germany and Great Britain. The data set consists of 16 raw input time series. The selection of these input signals is guided by the idea, that a major currency is driven by at least *four* influences [Mur91]. These influences are the development of (*i.*) other major currencies, and the development of the national and international (*ii.*) bond, (*iii.*) stock and (*iv.*) commodity markets.

The preprocessing of the raw input data is simple: We calculate the momentum (relative change, see Eq. 2.7) and force (normalized curvature, see Eq. 2.8) of each raw input time series [NZ98, p. 374]. In combination with each other, these preprocessing functions enable us to capture the underlying dynamics of the raw input signals [NZ98, p. 374-5]. The transformed time series are scaled such that they have a mean of zero and a variance of one [PDP00, p. 73].

Note, that raw financial time series are likely to contain inflationary (or linear) trends [NZ98, p. 374-5]. From this it follows, that the condi-

Table 4.1 Raw input time series used for all models of the empirical study

Model Inputs	
Japanese Yen / USD FX-rate	Japanese Yen / DEM FX-rate
British Pound / USD FX-rate	British Pound / DEM FX-rate
3-month interest rate USA	3-month interest rate Germany
3-month interest rate Japan	10-years interest rate USA
10-years interest rate Germany	10-years interest rate Japan
Dow Jones Stock Index	German DAX 30
Nikkei 225	FTSE 100
CRB-Future Index	Gold Price per Oz. in USD

tions of (weak) stationarity are violated [PDP00, p. 96]. Non-stationary time series can be transformed into stationary signals by differencing the data [PDP00, p. 96-7]. The momentum (Eq. 2.7) incorporates a one step differencing, while the forces (Eq. 2.8) contains a two step differencing of the raw data [PDP00, p. 96-7]. Thus, both preprocessing functions can be used to transform non-stationary input signals to stationarity. In case of financial time series, a one step differencing is usually sufficient to ensure stationarity [PDP00, p. 97].

6.1.3 NETWORK ARCHITECTURES & BENCHMARKS

The time-delay recurrent neural networks and benchmarks used in the empirical study are outlined in Tab. 4.2, 4.3 and 4.4.

First let us point to the time-delay recurrent neural networks considered in this study. As we will explain, the design of the empirical study allows us to examine the contribution of the different architectural building blocks to the model performance. In other words, we will study the benefits of e. g. unfolding in space *and* time, alternating errors or undershooting.

We start off with a basic unfolding in time network architecture (RNN, see Tab. 4.2 and Fig. 4.4). The second network is based on the preceding architecture, but includes also the concept of unfolding in space *and* time (Tab. 4.2). In other words, both neural networks are unfolded over the same time horizon $(t-6,\dots,t+6)$, contain an identical overshooting

Table 4.2 Description of the recurrent networks used in the empirical study (part 1)

Recurrent Neural Networks I	
Model	*Description*
Recurrent neural network (RNN)	We use the overshooting neural network shown in Fig. 4.4. The unfolding in time contains 12 time steps. Each time step of the unfolding represents one week. The time interval is from $t-6$ to $t+6$. This means, that we have six past time steps and an additional overshooting branch of six future time steps. The length of the unfolding in time is determined by the estimation of the *MIC* [ZN01, p. 323-5]. The network is trained by error backpropagation using shared weights [RHW86, p. 354-57]. We employ the *vario-eta* learning rule [Zim94, p. 48-49].
RNN with unfolding in space & time	The network architecture is based on the preceding RNN. As an additional feature, the recurrent neural network incorporates the concept of unfolding in space *and* time. The smoothness penalty is calculated via the differences $s_t - s_{t-1}$ and $s_{t+1} - s_t$. The smoothness of the internal flow is enforced by the penalty function described in Eq. 4.29.

branch $(t+1, \ldots, t+6)$ and are trained with *vario-eta* learning. This design enables us to isolate the effect of unfolding in space *and* time.

The third model is a standard error correction neural network (Tab. 4.3 and Fig. 4.6). The settings of the ECNN correspond to the basic RNN. Hence, one may study the effect of the error correction mechanism by comparing these networks.

We enhance the standard ECNN by the application of the unfolding in space *and* time concept (Tab. 4.3 and Fig. 4.19). Once again, it is important to note, that the design of the extended ECNN bears resemblance to the other network architectures. Thus, one may compare this model with the extended RNN in order to analyze the effect of the error correction mechanism. Furthermore, the benefit of the unfolding in space *and* time can be analyzed by comparing the network with the standard ECNN.

In addition, we evaluate an ECNN which incorporates alternating errors (Tab. 4.3 and Fig. 4.7). The basic design of this model (e. g. unfolding in time, overshooting and training) corresponds to the standard

Table 4.3 Description of the recurrent networks used in the empirical study (part 2)

Recurrent Neural Networks II

Model	*Description*
Error correction neural network (ECNN)	The architecture of the ECNN is depicted in Fig. 4.6. The truncation length of the finite unfolding in time is equal to that of the RNN, i. e. we choose $t-6$. The ECNN also incorporates an overshooting branch of six future time steps $t+1,\ldots,t+6$. The network is trained by vario-eta learning and standard error backpropagation using shared weights.
ECNN with unfolding in space & time	The network architecture is shown in Fig. 4.19. The extended ECNN is unfolded over the time interval from $t-6$ to $t+6$. This means, that we have six overshooting time steps. The truncation length of the unfolding is determined by estimating the *MIC*. The smoothness penalty is computed on the basis of the differences $s_t - s_{t-1}$ and $s_{t+1} - s_t$. We apply the penalty term described in Eq. 4.29. The extended ECNN is trained with vario-eta learning.
ECNN with alternating errors	The ECNN incorporating the concept of alternating errors is depicted in Fig. 4.7. The model incorporates a number of 12 unfolding time steps ($t-6$ to $t+6$). The overshooting branch consists of six future time steps ($t+1$ to $t+6$). We computed the alternating errors along the past time steps of the unfolding. The network is trained with vario-eta learning.
ECNN with undershooting	The model architecture corresponds to Fig. 4.10. As a specialty, the data time grid of the model is given by *monthly* observations. The undershooting network describes the FX-market dynamics internally by *four* intermediate time steps. This means, that we model the weekly development of the FX-rate on the basis of monthly observations. Since we use monthly observations, the ECNN is only unfolded over 6 time steps (i. e. 6 month from $t-3$ to $t+3$). The overshooting branch provides us with a sequence of three forecasts ($t+1,\ldots,t+3$).

ECNN. This enables us to investigate the contribution of alternating errors to the model performance.

As a specialty, we also examine a combination of ECNN and undershooting (Tab. 4.3 and Fig. 4.10). In contrast to the other models, the undershooting network is based on *monthly* observations. However, the internal time grid of the model refers to a weekly time scheme: We describe the FX-market dynamics (measured by monthly observations) internally by *four* intermediate time steps (weeks). A comparison between the undershooting model and the standard ECNN enables us to measure the benefit of such a time-grid refinement.

All recurrent neural networks (Tab. 4.2 and 4.3) incorporate an oversized overshooting branch. Although we only refer to the one month forecast horizon, the additional future time steps provide us with valuable information about the analyzed dynamical system and act as a further regularization method for the learning. Note, that all networks are endowed with the robust error function $\ln\cosh$ (Eq. 5.9) [NZ98, p. 387-88].

Another direction of the empirical study is a benchmark comparison. Here, we evaluate the performance of the recurrent neural networks by a comparison with *three* benchmarks (Tab. 4.4): (*i.*) a linear regression model, (*ii.*) a 3-layer feedforward neural network (MLP) and (*iii.*) the 11-layer architecture (Tab. 4.4 and Fig. 3.7). The linear regression model and the MLP are frequently used in econometrics. The 11-layer architecture is a feedforward neural network which is especially designed for the modeling of dynamical systems [NZ98, p. 383-86]. Therefore, a comparison of this benchmark with the time-delay recurrent networks should be interesting.

6.1.4 PERFORMANCE MEASURES

The benchmark comparison is performed on the basis of *five* common performance measures: (*i.*) hit rate, (*ii.*) Sharpe ratio, (*iii.*) accumulated return of investment, (*iv.*) realized potential and (*v.*) annualized return [Ref95, p. 70-74].

First, we employ a 'hit rate' counting how often the sign of the FX-rate shift is correctly predicted [Ref95, p. 69].

Second, we refer to the well-known Sharpe ratio [EG95, PDP00, p. 639 and p. 267-8]. The Sharpe ratio is defined as the excess return of the model (difference between the model return r_M and a risk-free return r_f) divided by the standard deviation σ_M of the model return, i. e. $SR = (r_M - r_f)/\sigma_M$. The measure is often referred to as an excess return to variability ratio. Note, that the average return of a risk-free asset is 4% during the out-of-sample time period (Jan. 96 – Aug. 97).

The accumulated return of investment is based on a simple technical trading strategy, which utilizes the model forecasts [Ref95, p. 70]: If we

Table 4.4 Description of the benchmarks used in the empirical study

Benchmarks	
Model	*Description*
Linear regression	We performed a linear regression analysis to model the linear dependence between the FX-rate shift and the external influences [PDP00, p. 201-11]. We apply t-tests to evaluate the statistical significance of the regression coefficients [PDP00, p. 279-83]. Furthermore, we refer to the Durbin-Watson test statistic to detect the presence of autocorrelation in the model residuals [PDP00, p. 297-307]. We estimated the regression model by ordinary least squares (OLS) [PDP00, p. 211-20].
3-layer MLP	The 3-layer network incorporates a hidden layer with 20 neurons (MLP, see Fig. 2.8 [Hay94, Bis95, p. 156 and p. 116-20]). The hidden neurons are equipped with tanh squashing functions [Zim94, p. 5-6]. The output layer provides us with a one month forecast of the USD / DEM FX-rate shift. Error signals are generated by the robust cost function $\ln\cosh(.)$ (Eq. 5.9). The network is trained by error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 161-68, p. 173-75 and p. 395-99]. We optimized the MLP by EBD weight pruning [TNZ97, p. 670-72].
11-layer network	The 11-layer architecture is depicted in Fig. 3.7 [NZ98, ZN01, p. 383-86 and p. 318-321]. According to the weekly data and the one month forecast horizon, the width m of the forces and embeddings is preset to $m = 4$ (see Eq. 3.5 and 3.6). The hidden force as well as the hidden embedding layer consist of 20 neurons. We trained the 11-layer network by error backpropagation in combination with pattern-by-pattern learning rule.

expect a positive (negative) return, we execute a long (short) position. The net return of the trading strategy (accumulated return) is calculated by cumulating the returns of the different trades. For simplicity, transaction costs are neglected.

The realized potential is defined as the ratio of the accumulated model return to the maximum possible accumulated return, which is achieved by assuming perfect forecasts [Ref95, p. 71].

The annualized return is the average yearly profit of the preceding trading strategy [NZ98, p. 419].

Note, that all performance measures are calculated only on the basis of the generalization set (out-of-sample period).

6.2 RESULTS OF THE EMPIRICAL STUDY

In the following we present the results of our empirical study. The performance measures calculated for the different models on the out-of-sample period are summarized in Tab. 4.5. The accumulated model returns are compared in Fig. 4.20.

Table 4.5 Performance measures of the different models. All performance criteria are measured on the generalization set.

Performance Evaluation				
Model	*Hit Rate*	*Sharpe Ratio*	*Realized Potential*	*Annualized Return*
Recurrent neural network (RNN)	55.29%	0.2829	29.31%	7.95%
RNN with unfolding in space & time	57.65%	0.2829	31.17%	8.36%
Error correction neural network (ECNN)	69.88%	0.5375	55.64%	13.78%
ECNN with unfolding in space & time	69.88%	0.6267	68.17%	15.46%
ECNN with alternating errors	65.06%	0.4399	47.26%	11.72%
ECNN with undershooting	60.66%	0.3653	36.51%	8.94%
linear regression	44.71%	0.0766	9.25%	2.23%
3-layer MLP	50.59%	0.1784	20.95%	5.13%
11-layer network	65.43%	0.3774	45.20%	9.46%

6.2.1 COMPARING THE TIME-DELAY RECURRENT NETWORKS

Let us begin with the evaluation of the recurrent neural network architectures: Across all performance measures, the standard ECNN is superior to the basic unfolding in time network (RNN). This indicates, that the error correction mechanism of the ECNN enables us to augment the performance of the modeling. We may conclude, that the error correction mechanism allows us to handle unknown external influences and the noise contained in the FX-market data. Being so, the fitting of the underlying FX-market dynamics and the model performance are improved.

The desirable effect of the error correction mechanism becomes also apparent by comparing the unfolding in space *and* time RNN and the related ECNN. It turns out, that among the unfolding in space *and* time network architectures the ECNN is superior. Since the only difference between the networks is the error correction mechanism, the benefit of the error correction is re-confirmed.

Comparing the performance of the basic RNN with the unfolding in space *and* time RNN, it turns out that the unfolding in space *and* time network is slightly superior. The desirable effect of unfolding in space *and* time becomes more obvious by looking at the error correction networks. The standard ECNN is clearly outperformed by the ECNN incorporating unfolding in space *and* time. The extended ECNN achieves not only a better Sharpe ratio, but also a higher annualized return and realized potential. Remarkably, the ECNN incorporating unfolding in space *and* time realizes nearly 70% of the maximum reachable performance. This is an illustration of the power of unfolding in space *and* time. We may conclude, that unfolding in space *and* time allows us to reduce the noise level of the FX-rate shifts. The forecast problem becomes easier, because the transformed system evolves more smoothly over time.

As shown in Tab. 4.5, the inclusion of the alternating errors concept has no desirable effect on the modeling. The performance of the ECNN incorporating alternating errors is lower than that of the standard ECNN. To provide an explanation for this observation, one has to analyze the concept of alternating errors on the basis of the USD /DEM FX-market characteristics. We believe that the USD / DEM FX-rate is more likely to oscillate in a certain range than exhibiting a typical trend behavior. Therefore, it is *not* necessary to incorporate the concept of alternating errors into the modeling of the USD / DEM FX-market. Our experiments indicate, that the inclusion of alternating errors is valuable for the modeling of interest rates.

Finally, let us focus on the combination of ECNN and undershooting. Compared to the standard ECNN, the undershooting network exhibits a lower performance. However, it is important to notice, that the modeling task of the undershooting is much harder: Although the data is available on a weekly basis, the undershooting network is restricted to monthly observations. By the application of undershooting, we refine the model time grid relative to the wider-meshed time grid of the data. Similar to the data time grid of the other networks, the undershooting model internally operates on a weekly time scheme. In the light of this model design, we may conclude that the undershooting network achieves a good performance. In other words, undershooting has a desirable contribution to the modeling of the USD / DEM FX-market. The forecast accuracy of the undershooting ECNN becomes more obvious in the benchmark comparison.

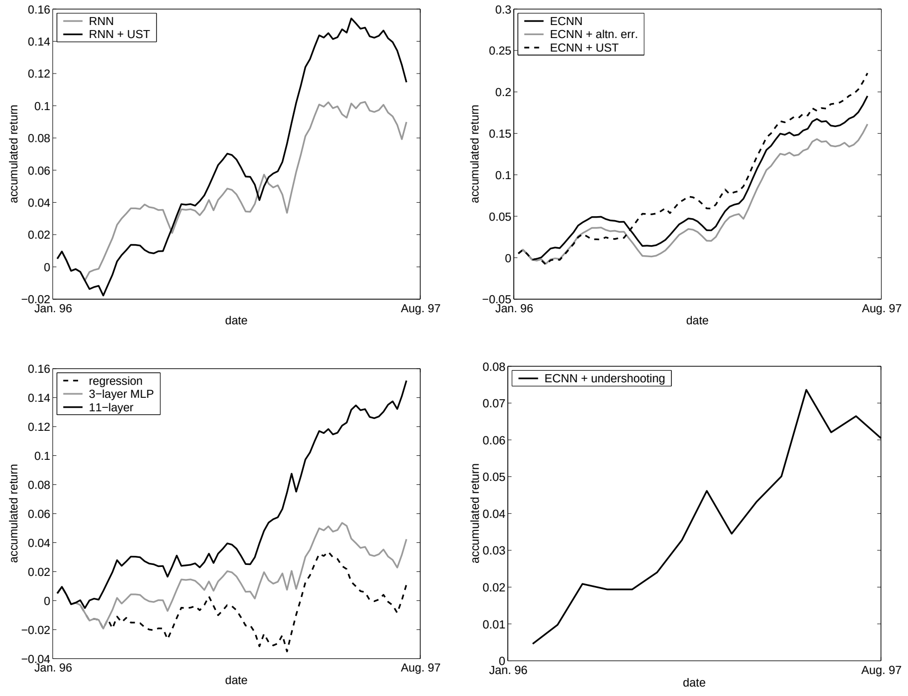


Figure 4.20 Out-of-sample accumulated return of the different time-delay recurrent neural networks and benchmarks. Note, that the data time grid of the ECNN incorporating undershooting is based on monthly data.

6.2.2 BENCHMARK COMPARISON

Let us now compare the recurrent neural networks to the preset benchmarks: As shown in Tab. 4.5, all recurrent network architectures are highly superior to the linear regression model. The poor result of the linear regression analysis indicates that a simple linear model is not able to describe the underlying dynamics of the FX-market appropriately. Note, that the linear regression model is also inferior to the other benchmarks.

Comparing the performance of the different neural networks to the 3-layer MLP, we find that the MLP is also inferior to *all* recurrent network architectures. This is especially true for the Sharpe ratios. The lowest Sharpe ratio calculated for a recurrent network (RNN, $SR = 0.2829$) is almost twice as high as that of the MLP ($SR = 0.1784$). This indicates, that the incorporation of prior knowledge into the modeling helps us to improve the forecast performance.

As shown in Tab. 4.5, it is more difficult to outperform the 11-layer architecture. The performance of the 11-layer network is slightly superior to the basic time-delay recurrent network architectures (Tab. 4.2). In other words, the 11-layer approach allows an appropriate description of the FX-market dynamics, which is comparable to the results of the basic time-delay recurrent networks (RNNs).

However, one should notice that both modeling frameworks have a different view on dynamic systems: In our time-delay recurrent neural networks, the present time state is explained by a superposition of external influences $\{u_t, u_{t-1}, \ldots\}$ and the autonomous development $\{s_t, s_{t-1}, \ldots\}$ from *all* previous time steps. The recurrent system sets up a memory (superposition of information) to identify temporal relationships [CK01, p. 15-6]. In contrast, the 11-layer architecture includes a *preset* description of the autonomous dynamics on the output side of the network. Following Takens' theorem, the underlying dynamics is captured by the concept of forces and embeddings [NZ98, p. 384-85]. Of course both approaches allow us to capture the underlying system dynamics, but the recurrent framework is more flexible in dealing with temporal dependencies.

Remarkably, the 11-layer architecture is outperformed by the standard and the unfolding in space *and* time ECNN. Since the performance of the 11-layer network is similar to a basic RNN, we may conclude that the additional performance of the ECNN is generated by the error correction and the concept of unfolding in space *and* time. This re-indicates the benefits of incorporating prior knowledge into the modeling. Finally note, that the 11-layer network is superior to the other benchmarks.

7. PRELIMINARY CONCLUSION

We believe that recurrent neural networks and especially ECNN provide a very promising framework for the solution of forecasting problems.

If the underlying dynamics is mainly driven by external factors, it is not obvious why a recurrent approach should be superior to a feedforward network. However, if there is a significant autonomous part in the dynamics, the recurrent framework of ECNN seems to be appropriate.

ECNN utilizes the previous model error as an additional input. Hence, the learning can interpret the models misfit as an external shock which is used to guide the model dynamics afterwards. This allows us to prevent the autonomous part of the model to adapt misleading inter-temporal causalities. If we know that a dynamical system is influenced by external shocks, the error correction mechanism of the ECNN is an important prestructuring element of the network architecture to compensate missing inputs.

Extending the ECNN by using techniques like overshooting, alternating errors, variants-invariants separation or unfolding in space and time, one is able to include additional prior structural knowledge of the underlying dynamics into the model. For example, the variants-invariants separation in form of a bottleneck coordinate transformation allows to handle high dimensional problems, while overshooting enforces the autoregressive part of the model.

By the application of undershooting, we introduced another important prestructuring element for the modeling of dynamical systems. In the context of time-delay recurrent neural networks, undershooting allows to model the relationship between the model and the data time grid, such that it is also possible to use a finer model time grid than the data. Interestingly, undershooting provides also a statistical improvement: The networks output is computed as an average which may reduce the noise.

The empirical study modeling the US-Dollar (USD) / German Mark (DEM) FX-market indicated, that the forecast performance can be improved by the incorporation of additional prior knowledge and first principles into the model building. The benchmark comparison underlined that especially the ECNN is highly superior to more conventional forecasting techniques.

Chapter 5

TRAINING OF NEURAL NETWORKS BY ERROR BACKPROPAGATION

Error backpropagation, firstly introduced by Werbos in 1974 [Wer74] and reinvented by D. Rumelhart, G. Hinton and R. Williams in 1986 [RHW86], is the foundation of supervised learning of neural networks [Hay94, p. 63-4 and p. 156-7].

Regarding multi-layer neural networks, the information about the overall network error is *not* directly available for the training of the hidden layers of the network. This is well-known as the credit assignment problem [Hay94, MiPa69, p. 62]. As a remedy, the error backpropagation algorithm provides all parts of the network with information about the network error. More precisely, error backpropagation is a method, which derives the first partial derivations of the neural network error function with respect to the network parameters (weights) [Zim94, p. 38].

On this basis, learning rules optimize the network by adapting the weights such that the differences between the outputs of the network and associated observations are minimal on the average [Zim94, p. 40]. We refer to this principle as *supervised learning*, since the deviation between the networks output and corresponding target values guides the optimization [Bis95, Hay94, p. 10 and p. 63-4].

This chapter consists of *seven* parts. The *first* part deals with the standard backpropagation algorithm. Since outliers in financial data sets are a common problem, the *second* part introduces (*i.*) the error function $\ln\cosh(.)$ and (*ii.*) the activation function $\tanh(.)$. Both functions support a robust estimation of the network parameters in the presence of noise or outliers in the data set [NZ98, p. 375-76 and p. 387-88].

In the *third* part we discuss the so-called *shared weights* extension of standard backpropagation [RHW86, p. 354-7]. This technique is used for the learning of unfolded time-delay recurrent networks. Solving the system identification task of time-delay recurrent neural networks by unfolding in time, the finite truncation length of the unfolding has to be determined [ZN01, p. 324]. On this problem, the *fourth* part proposes a suitable iteration scheme, which can easily be applied to unfolding in time neural networks [ZN01, p. 324 and p. 326-7].

In addition to error backpropagation, learning rules are required, which determine the weight changes [Zim94, p. 40]. Therefore, two stochastical learning rules, namely *vario-eta* and *pattern-by-pattern* learning [NZ98, p. 395-99], are introduced in the *fifth* section. In addition, we discuss their impact on the estimation of the maximum intertemporal connectivity (*MIC*), i. e. the estimation of the finite truncation length of the unfolding in time [ZN01, p. 324]. Interestingly, we find that the learning can support the estimation of the MIC.

Besides the architectural design of the neural network, learning from data is also a major issue of the model building. Here, we are interested in finding time invariant structures out of varying time series. Learning can be seen as the generation of a structural hypothesis, which have to be revised by e. g. weight pruning methods [NZ98, ZGT02, p. 405, 413-17 and p. 409-11]. Based on standard error backpropagation, the *sixth* part develops a new learning rule which evaluates only specific gradients during the training [ZGT02, p. 410-11]. By this, we are able to generate localized structures in the network, which support the optimization of the network architecture. Furthermore, we discuss common weight pruning methods to optimize the neural network structure [NZ98, ZGT02, p. 405-10 and p. 409-10].

Finally, the *seventh* part deals with an empirical study concerning the short-term German interest rate (3-month interbanking offered rate). We forecast the weekly changes in the short-term German interest rate by a time-delay recurrent neural network [ZN01, p. 326]. The system identification task is solved by an unfolding in time. In this connection, we show that the estimation of the *MIC* depends on the employed learning rule. Our results indicate, that a neural network which is trained by *vario-eta* learning is superior to a similar network trained by *pattern-by-pattern* learning, because *vario-eta* discovers additional dependencies in the data.

1. STANDARD ERROR BACKPROPAGATION ALGORITHM

In this section we deal with standard error backpropagation [Hay94, Bis95, p. 161-68, p. 173-75 and p. 140-48]. We exemplify the error backpropagation algorithm on the basis of a 3-layer feedforward neural network consisting of one input layer with $i = 1, \ldots, l$ neurons, one hidden layer with $j = 1, \ldots, m$ neurons and one output layer with $k = 1, \ldots, n$ neurons [Zim94, p. 37]. Due to methodical reasons, we omit any bias connections.[1] The forward information flow of the latter neural network is depicted in Fig. 5.1 [Zim94, p. 37].

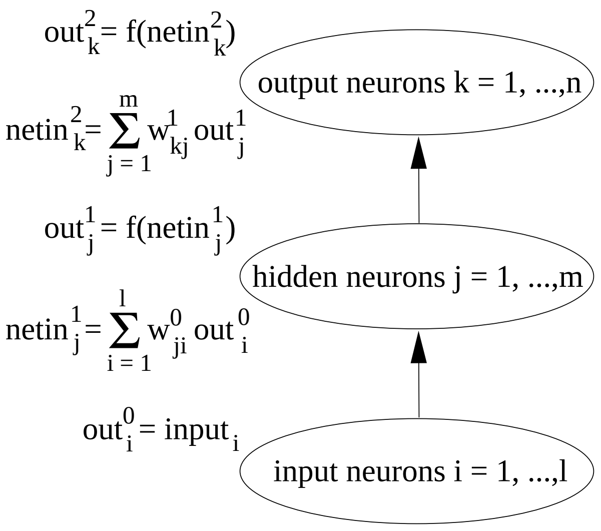

$\text{out}_k^2 = f(\text{netin}_k^2)$
output neurons k = 1, ...,n
$\text{netin}_k^2 = \sum_{j=1}^{m} w_{kj}^1 \text{out}_j^1$
$\text{out}_j^1 = f(\text{netin}_j^1)$
hidden neurons j = 1, ...,m
$\text{netin}_j^1 = \sum_{i=1}^{l} w_{ji}^0 \text{out}_i^0$
$\text{out}_i^0 = \text{input}_i$
input neurons i = 1, ...,l

Figure 5.1 Forward information flow of a 3-layer neural network [Zim94, p. 37]. Let w_{ji}^0 and w_{kj}^1 denote the weights of the network. A weight w_{ji}^0 connects an input i to a hidden neuron j, while w_{kj}^1 connects a hidden neuron j to an output neuron k. The upper indices denote the level of the network. We assume, that the neurons of each layer are fully connected to those of the directly sequencing layer.

We want to minimize the deviation of the network output out_k^2 and associated target values tar_k across a set $t = 1, \ldots, T$ of training pattern. For this purpose, we adjust the weights w_{ji}^0, w_{kj}^1 such that the overall mean-square error function (MSE) [Wei90, p. 179] of the network is minimized (Eq. 5.1).

$$\text{MSE} = \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{n} \frac{1}{2} \left(\text{out}_k^2 - \text{tar}_k\right)^2 \rightarrow \min_{w_{ji}^0, w_{kj}^1} \tag{5.1}$$

Due to visual clarity, we suppressed the subindex t denoting the patterns of the training set. Note, that we multiplied the squared deviations of the network output out_k^2 and associated target values tar_k by 0.5. This enables us to simplify the calculations of the partial derivatives of the error function [Zim94, p. 37-8].

[1]Bias connections to the hidden layer as well as to the output layer can easily be established by assuming a single input neuron, which is directly connected to these layers. The output of the latter input neuron is equal to 1 at any time. [Bis95, p. 142]

Error backpropagation is an efficient algorithm to derive the first partial derivatives of the overall network error function with respect to the network parameters w^0_{ji} and w^1_{kj} [Zim94, p. 38]. Utilizing the partial derivatives provided by the backpropagation algorithm, the parameter optimization task is solved by specific learning rules [Zim94, NZ98, p. 40-49 and p. 395-397].

In case of a 3-layer neural network, the backpropagation algorithm consists of *two* steps [Zim94, p. 38]: First, we calculate the partial derivatives of Eq. 5.1 with respect to the weights w^1_{kj} which are located between the hidden and the output layer of the network. For this purpose, we are using the chain-rule.[2] Second, by the reapplication of the chain-rule, we compute the partial derivatives of Eq. 5.1 with respect to the weights w^0_{ji} which connect the inputs i to the hidden layer.

Approaching the first step of the backpropagation algorithm, the partial derivatives of the network error function (Eq. 5.1) with respect to the weights w^1_{kj} are computed as follows [Zim94, p. 38]:

$$\begin{aligned} \frac{\partial \mathrm{MSE}}{\partial w^1_{kj}} &= \frac{1}{T}\sum_{t=1}^{T}\left(\mathrm{out}^2_k - \mathrm{tar}_k\right)\frac{\partial \mathrm{out}^2_k}{\partial \mathrm{netin}^2_k}\frac{\partial \mathrm{netin}^2_k}{\partial w^1_{kj}} \\ &= \frac{1}{T}\sum_{t=1}^{T}\left(\mathrm{out}^2_k - \mathrm{tar}_k\right) f'(\mathrm{netin}^2_k)\mathrm{out}^1_j. \end{aligned} \tag{5.2}$$

Assuming, that the neurons j of the hidden layer are fully connected to the output neurons k, we have a set of $k \cdot j$ partial derivatives of the error function (Eq. 5.1). A simpler notation of the partial derivatives in Eq. 5.2 can be obtained by using the auxiliary term of Eq. 5.3 [Zim94, p. 38].

$$\delta^2_k = (\mathrm{out}^2_k - \mathrm{tar}_k) f'(\mathrm{netin}^2_k) \tag{5.3}$$

The term δ^2_k (Eq. 5.3) is the product of the individual network error $(\mathrm{out}^2_k - \mathrm{tar}_k)$ and the partial derivative of the activation function f with respect to a particular weight w^1_{kj}. In this connection, f depends directly on the input netin^2_k (see Fig. 5.1). The substitution of the auxiliary term δ^2_k (Eq. 5.3) in Eq. 5.2 yields to

$$\frac{\partial \mathrm{MSE}}{\partial w^1_{kj}} = \frac{1}{T}\sum_{t=1}^{T}\delta^2_k \mathrm{out}^1_j \tag{5.4}$$

[2]According to the chain-rule, the derivative of $y = f(g(x))$ is $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\cdot\frac{\partial u}{\partial x}$.

Proceeding this way, we obtain the cumulative gradient of the second network level, i. e. the partial derivatives of Eq. 5.1 with respect to the weights w^1_{kj}. According to Eq. 5.4, the partial derivatives are defined as the product of the output of the hidden neurons out^1_j and the auxiliary term δ^2_k (Eq. 5.3).

The calculation of the partial derivatives of network error function (Eq. 5.1) with respect to the weights w^0_{ji} (see Fig. 5.1) requires a repetitive execution of the chain-rule [Zim94, p. 38]. Applying the chain-rule twice, the partial derivatives of Eq. 5.1 with respect to the weights w^0_{ji} are

$$\begin{aligned}\frac{\partial \text{MSE}}{\partial w^0_{ji}} &= \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{n}\left(\text{out}^2_k - \text{tar}_k\right)\frac{\partial \text{out}^2_k}{\partial \text{netin}^2_k}\frac{\partial \text{netin}^2_k}{\partial \text{out}^1_j}\frac{\partial \text{out}^1_j}{\partial \text{netin}^1_j}\frac{\partial \text{netin}^1_j}{\partial w^0_{ji}} \\ &= \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{n}\left(\text{out}^2_k - \text{tar}_k\right) f'(\text{netin}^2_k) w^1_{kj} f'(\text{netin}^1_j)\text{out}^0_i .\end{aligned} \tag{5.5}$$

Substituting the auxiliary term δ^2_k of Eq. 5.3 in Eq. 5.5 leads to

$$\frac{\partial \text{MSE}}{\partial w^0_{ji}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{n}\delta^2_k w^1_{kj} f'(\text{netin}^1_j)\text{out}^0_i . \tag{5.6}$$

Since the neurons i of the input layer are fully connected to the neurons j of the hidden layer, we obtain a set of $i \cdot j$ partial derivatives. Again, we simplify the notation of the partial derivatives in Eq. 5.6 by the usage of an auxiliary term δ^1_j [Zim94, p. 38-9], which is defined as

$$\delta^1_j = f'(\text{netin}^1_j)\sum_{k=1}^{n} w^1_{kj}\delta^2_k . \tag{5.7}$$

The auxiliary term δ^1_j of Eq. 5.7 is composed of *two* multiplicatively connected blocks: The *first* block is the partial derivative of the hidden layer activation function f with respect to a particular weight w^0_{ji}. Since we are now considering the first level of the network, the activation function f depends on the input netin^1_j (see Fig. 5.1). The *second* block is the sum of the auxiliary terms δ^2_k (Eq. 5.3) weighted by w^1_{kj}. Substituting Eq. 5.7 in Eq. 5.6 yields to the cumulative gradient of the first network level [Zim94, p. 389]:

$$\frac{\partial \text{MSE}}{\partial w^0_{ji}} = \frac{1}{T}\sum_{t=1}^{T}\delta^1_j \text{out}^0_i \ . \tag{5.8}$$

Having explained, how the partial derivatives of the 3-layer neural network with respect to the weights w_{ji}^0 and w_{kj}^1 are calculated, we are now going to review the backpropagation algorithm by its application to the 3-layer neural network in case of a single training sample (see Fig. 5.2) [Zim94, p. 39-40].

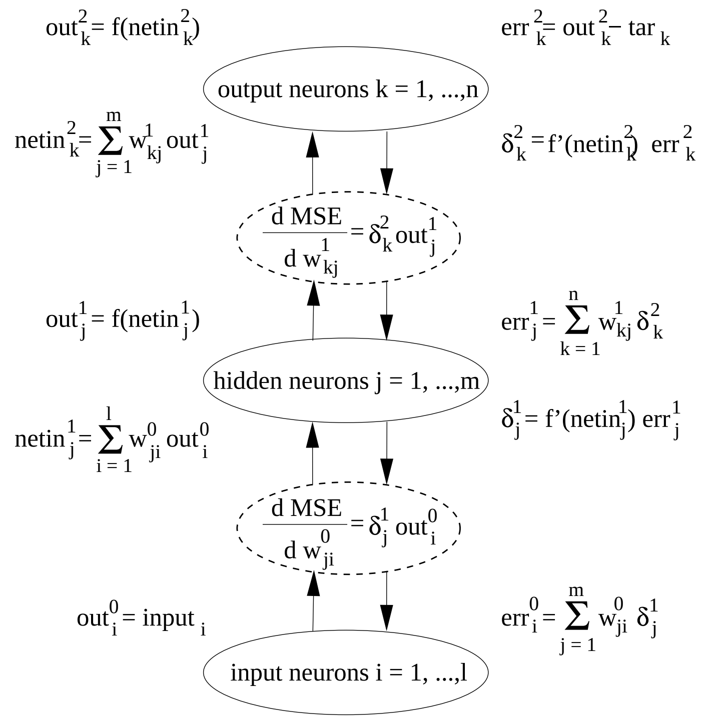

$out_k^2 = f(netin_k^2)$
$err_k^2 = out_k^2 - tar_k$
output neurons k = 1, ...,n
$netin_k^2 = \sum_{j=1}^{m} w_{kj}^1 out_j^1$
$\delta_k^2 = f'(netin_k^2)\ err_k^2$
$\frac{d\,MSE}{d\,w_{kj}^1} = \delta_k^2 out_j^1$
$out_j^1 = f(netin_j^1)$
$err_j^1 = \sum_{k=1}^{n} w_{kj}^1 \delta_k^2$
hidden neurons j = 1, ...,m
$netin_j^1 = \sum_{i=1}^{l} w_{ji}^0 out_i^0$
$\delta_j^1 = f'(netin_j^1)\ err_j^1$
$\frac{d\,MSE}{d\,w_{ji}^0} = \delta_j^1 out_i^0$
$out_i^0 = input_i$
$err_i^0 = \sum_{j=1}^{m} w_{ji}^0 \delta_j^1$
input neurons i = 1, ...,l

Figure 5.2 The complete error backpropagation algorithm in case of a single training sample [Zim94, p. 39].

As depicted in Fig. 5.2, the error flow, which is induced by the backpropagation algorithm, is contrary to the forward information path of the network [Zim94, p. 39-40]. This is clarified by the reverse indexing of the summations of the forward and backward path.

The cumulative gradients of Eq. 5.4 and Eq. 5.8 establish a crosslink between the forward and the error backpropagation path [Zim94, p. 39-40]: The outputs out_i^0 resp. out_j^1, which are calculated on the forward path, are multiplied with the auxiliary terms δ_j^2 resp. δ_j^1 of the backpropagation path in order to calculate the partial derivatives.

The auxiliary terms δ_j^2 and δ_j^1 (see Eq. 5.3 and 5.7) implicitly propagate the error information through the network. However, the *actual* error flow is induced by the propagation of err_k^2, err_j^1 and err_i^0 along the backward path of the network [Zim94, p. 38-40].

We explained the standard error backpropagation algorithm on the basis of a 3-layer feedforward neural network. The application of the algorithm to more sophisticated feedforward neural networks, e. g. the 11-layer architecture (chp. 3), is straightforward.

2. ROBUST ESTIMATION OF NEURAL NETWORK PARAMETERS

In Eq. 5.1, we introduced the mean square error function (MSE) as a well-known model selection criteria.[3] As an advantage, the MSE "yields relatively simple error derivatives and results in asymptotically best estimators under certain distribution assumptions, i. e. homoscedasticity" [NZ98, p. 387]. However, the application of the MSE is critical, if some of these assumptions are violated. In practice, this may lead to unreliable model forecasts [NZ98, p. 387].

In this context, outliers in the data set are a common problem in time series analysis [Wei90, p. 195], because they may interference the learning of the network parameters. This is especially true for economic applications. Outliers in financial time series are often caused by unexpected shocks, e. g. cropping up strikes, political or economic crises, or so-called "information shocks" [NZ98, p. 387]. Information shocks are caused by news which are not in line with the expectations of the market participants, e. g. lower than estimated earnings of a company [NZ98, p. 387]. These events may cause discontinuances in the trajectories of the considered targets [NZ98, p. 387]. A general overview of the handling of outliers is given in [Wei90, p. 195-205].

In order to handle outliers in the data set, we propose the application of a robust error function, which does not stress huge deviations between the model outputs and corresponding target values. Such a measure is given by the $\ln\cosh(.)$ error function of Eq. 5.9 [NZ98, p. 387-88].

$$E_t = \frac{1}{a} \ln\cosh\left(a\left(\text{out}_k^2 - \text{tar}_k\right)\right) \tag{5.9}$$

The error function E_t of Eq. 5.9 can be seen as a smoother version of the mean absolute error function $|\text{out}_k^2 - \text{tar}_k|$ [NZ98, p. 387]. The parameter a is chosen on the constraint to be greater than 1 ($a > 1$). A typical setting is $a \in [3, 4]$ [NZ98, p. 387].

[3]An overview of alternative model selection criteria is given in e. g. Wei (1990) [Wei90, p. 151-156] or Zapranis et al. (1999) [ZaRe99, p. 22-35].

In case of small deviations between the model output and the target, the ln cosh error function adapts the behavior of the mean square error function (Eq. 5.1). For huge deviations, the ln cosh error function acts like the mean absolute error function [NZ98, p. 387-88]. Fig. 5.3 depicts the ln cosh error function (left-hand-side) and its derivative (right-hand-side) [NZ98, p. 388].

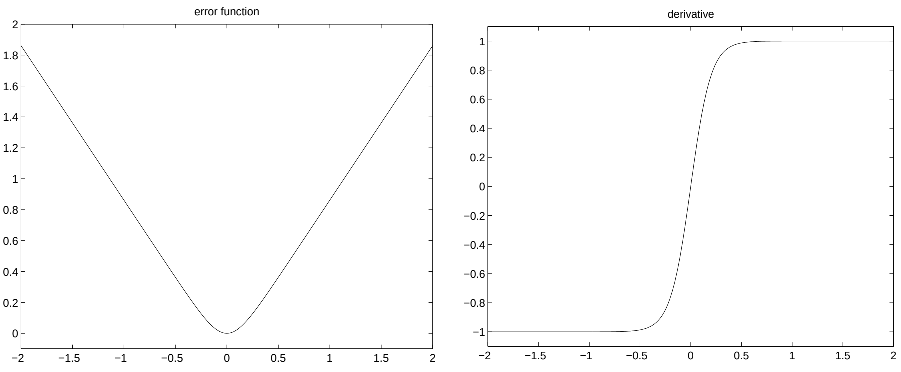


Figure 5.3 The ln cosh error function of Eq. 5.9, left, and its derivative, right [NZ98, p. 388]. Both functions are calculated for the case of $a = 5$.

Note, that the derivation of $\frac{1}{a}\ln\cosh(ax)$ is $\tanh(az)$, which is a continuous, smooth adaption of the derivative of the absolute value function $|x|$, i. e. the signum function $\operatorname{sign}(x)$ [NZ98, p. 388].

Besides the choice of a suitable error function, the specification of a neural network requires the selection of a non-linearity, which serves as an activation or so-called squashing function for the neurons. In principle, any non-linearity can be applied to the neurons as long as it is continuous and differentiable [ZaRe99, p. 24]. Most common activation functions are sigmoidal, logistic or thermodynamic-like functions [ZaRe99, p. 24].

We prefer to use hyperbolic tangent (Fig. 5.4, left) non-linearities as squashing functions [Zim94, Bis95, p. 40 and p. 127]. Already mentioned above, the hyperbolic tangent is a smooth approximation of the discontinuous signum function $\operatorname{sign}(x)$ (Fig. 5.4, right). Due to its linear range near the origin, small inputs pass the hyperbolic tangent essentially unchanged. In contrast, huge inputs are pushed into the saturation level of tanh(.) [NZ98, p. 375-76]. For example, in Fig. 5.4, left, the hyperbolic tangent reaches it saturation level for input values $|x| \geq 2$, while smaller values of x pass the non-linearity nearly unchanged. This implies, that the hyperbolic tangent acts as a limiter of outliers and

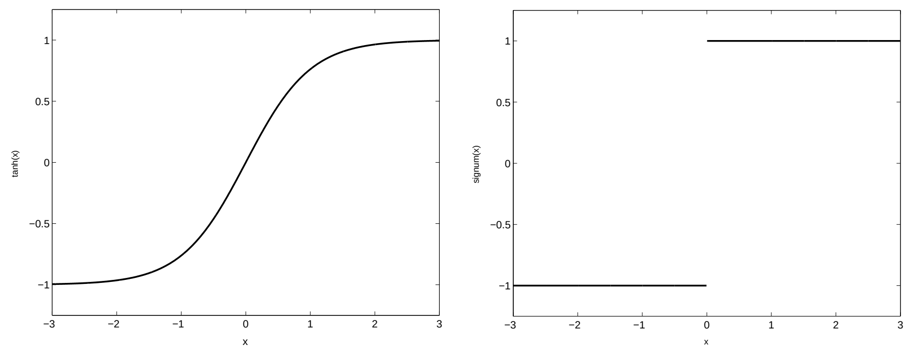


Figure 5.4 Hyperbolic tangent squashing function ($\tanh(x)$), left, which can be seen as a smooth approximation of the discontinuous signum function $\text{sign}(x)$, right [NZ98, p. 388]. Input values x are taken out of the interval $x \in \{-3, 3\}$.

supports a network internal preprocessing of the data [NZ98, p. 375-76]. Note, that the numerics works best if the variables included in the tanh squashing fluctuate around zero. – This fits best to the finite state space $(-1;1)^n$ created by this non-linearity [ZNG01a, p. 247].

The partial derivative of tanh(netin) with respect to a particular weight is

$$\begin{aligned} \frac{\partial \tanh(\text{netin})}{\partial \text{netin}} &= 1 - \tanh(\text{netin})^2 \\ &= 1 - (out)^2 \end{aligned} \tag{5.10}$$

During the backpropagation algorithm, the calculation of the partial derivatives of the hyperbolic tangent tanh(netin) is supported by former computations. For instance, one may re-utilize the value of tanh(netin) calculated on the forward path along backward path of the network [Zim94, p. 40]. Note, that netin reflects either netin_j^1 or netin_k^2 of Fig. 5.2.

As it can be seen from Eq. 5.10, large neural networks using tanh(.) activation function are likely to cause numerical difficulties during the backpropagation algorithm [MJ99, HBFS01, Hay94, ZN01, p. 134-6, p. 237-9, p. 773-4 and p. 325]. This is due to the fact, that the error flow often has to pass the derivative of the nonlinear squashing $\tanh(\cdot)$ several times top down the network architecture. For example, let us assume, that in case of a single training pattern the error err_k^2 and the net input netin_k^2 are large. Now, netin_k^2 pushes the activation function $\tanh(\cdot)$ towards its saturation level and the auxiliary term δ_k^2 (Eq. 5.3) becomes small due to Eq. 5.10. Since δ_k^2 is very small, the backward flowing error err_j^1 shrinks rapidly and thus, the gradients of Eq. 5.4 and

5.8 become meaningless as well. The outlined problem of long tanh(.) chains can be solved by a re-normalization of the gradients at each step of the backpropagation [NZ98, ZN01, p. 396 and p. 325]. Later on, we will introduce a proper learning rule called *vario-eta*, which includes such a re-normalization of the gradients [NZ98, p. 396].

3. SHARED WEIGHTS EXTENSION OF ERROR BACKPROPAGATION

Up to now, our considerations merely focused on feedforward neural networks. However, we believe that, time-delay recurrent networks are a more promising framework for the modeling of dynamical systems, because they e. g. allow the inclusion of memory effects [ZNG01a, p. 240 and 242]. Instead of solving the related system identification task by using a parameter optimization algorithm,[4] we refer to a so-called unfolding in time [RHW86, p. 354-57], which transfers the system identification into a spatial architecture. As we will show, this approach of training time-delay recurrent neural networks requires some enhancements of the standard error backpropagation algorithm. For this purpose, let us first revisit a simple model of a dynamical system and its specification by a time-delay recurrent neural network [ZN01, p. 321-327].

In Chapter 4, we described an open dynamic system in case of discrete time grids by a recurrent state transition $s_t = f(s_{t-1}, u_t)$ and an output equation $y_t = g(s_t)$ (see Eq. 4.1) [ZN01, p. 321]. The system identification task is to find two functions f and g such that the deviation between the system output y_t and corresponding target values y_t^d is minimal.

The outlined system identification task can be specified as a time-delay recurrent neural network, with $s_t = \tanh(As_{t-1} + Bu_t)$ and $y_t = Cs_t$ (see also Eq. 4.5). Doing so, the system identification task has turned into the parameter optimization problem. Here, we adjust the weight matrices A, B and C, such that the overall error function of the neural network is minimized [ZNG01a, p. 241].

Instead of solving the system identification task of the recurrent neural network by a parameter optimization algorithm [Pearl01, p. 182-96], we propose a (finite) unfolding in time solution by means of a spatial neural network architecture [ZN01, p. 323-25]. This procedure is based on the idea, that every time-delay recurrent neural network has an identically behaving representation in form of a feedforward neural network

[4]For details see e. g. Pearlmatter (2001) [Pearl01, p. 182-96] or Haykin (1994) [Hay94, p. 750-1].

[RHW86, p. 354-55]. The unfolding in time of a recurrent neural network into a corresponding feedforward architecture is illustrated in Fig. 5.5 [RHW86, p. 355].

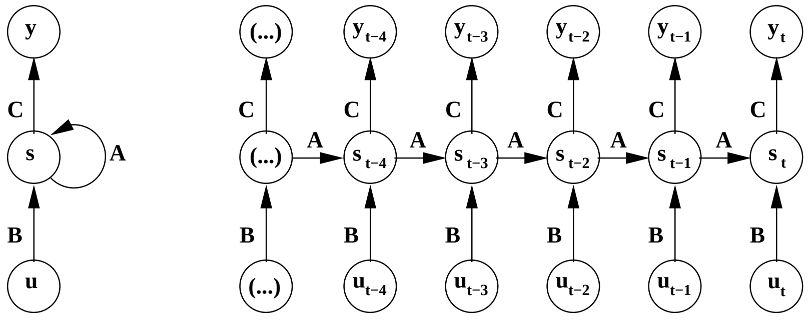


Figure 5.5 Time-delay recurrent neural network (see Eq. 4.5), left, corresponding unfolding in time neural network architecture, right [RHW86, p. 355].

If we ignore, that the internal state s_t of the recurrent neural network of Fig. 5.5, left, depends on the previous state transition s_{t-1}, i. e. assuming that $s_t = f(u_t)$, we are back in the framework of feedforward modeling [ZN01, p. 322]. The spatial unfolding in time architecture of Fig. 5.5, right, adapts this characteristic of the recurrent neural network: At each time step τ of the unfolding, we have the same non-recursive structure, i. e. a hidden state s_τ is only dependent on externals influences u_τ. In order to model the recursive structure, we establish connections between the hidden states $s_{t-4}, \ldots, s_t$ of the spatial unfolding network with respect to their chronological order. Now, the hidden state s_τ of the unfolding network depends on the previous hidden state $s_{\tau-1}$ *and* external influences u_τ. By this, we approximate the behavior of the recurrent neural network (Fig. 5.5, left) up to a sufficient level [RHW86, p. 354-56].

It is important to note, that the weight matrices A, B and C of the unfolding neural network of Fig. 5.5, right, are identical at each step of the unfolding in time. In other words, matrices A, B and C share the same memory for storing their weights. Hence, A, B and C are called *shared weight matrices*. The usage of shared weights ensures, that the unfolding network behaves as in the recurrent case [Hay94, RHW86, ZN01, p. 751-52, p. 355 and p. 323-4].

The unfolding in time neural network (Fig. 5.5, right) can be handled by a so-called *shared weights* extension of the standard error backpropagation algorithm.

Let us exemplify the integration of shared weights into the backpropagation algorithm by examining a 3-layer feedforward neural network (see Fig. 5.1). To adapt the idea of shared weights, we apply an additional constraint to the 3-layer network: We assume, that the network weights w_{kj} and w_{ji} are *identical.* This means that the weights w_{kj} connecting the hidden to the output layer are equal to the connections w_{ji} between the input and hidden layer. By this constraint, we also presume appropriate dimensions of the neural network layers. More formally, the shared weights constraint can be written as

$$w = w_{kj} = w_{ji}. \tag{5.11}$$

Given the constraint of Eq. 5.11, the partial derivative of the network overall error function (Eq. 5.1) with respect to a particular weight w can be obtained by applying the chain-rule and the product rule:[5]

$$\begin{aligned}
\frac{\partial \mathrm{MSE}}{\partial w} &= \frac{1}{T}\sum_{t=1}^{T} \left(\mathrm{out}_k^2 - \mathrm{tar}_k\right) \frac{\partial \mathrm{out}_k^2}{\partial \mathrm{netin}_k^2} \frac{\partial \mathrm{netin}_k^2}{\partial w_{kj}} \\
&\qquad + \left(\mathrm{out}_k^2 - \mathrm{tar}_k\right) \frac{\partial \mathrm{out}_k^2}{\partial \mathrm{netin}_k^2} \frac{\partial \mathrm{netin}_k^2}{\partial \mathrm{out}_j^1} \frac{\partial \mathrm{out}_j^1}{\partial \mathrm{netin}_j^1} \frac{\partial \mathrm{netin}_j^1}{\partial w_{ji}} \\
&= \frac{1}{T}\sum_{t=1}^{T} \left(\mathrm{out}_k^2 - \mathrm{tar}_k\right) f'(\mathrm{netin}_k^2)\mathrm{out}_j^1 \\
&\qquad + f'(\mathrm{netin}_j^1)\sum_{k=1}^{n} w_{kj} f'(\mathrm{netin}_k^2)\left(\mathrm{out}_k^2 - \mathrm{tar}_k\right)\mathrm{out}_i^0 \ .
\end{aligned} \tag{5.12}$$

Since we assume, that the network is fully connected, we have a set of $k \cdot j = j \cdot i$ partial derivatives. Likewise to the standard backpropagation algorithm, we simplify the notation of the partial derivatives (Eq. 5.12) by referring to the auxiliary terms δ_k^2 and δ_j^1 of Eq. 5.13 and Eq. 5.14:

$$\delta_k^2 = f'(\mathrm{netin}_k^2)(\mathrm{out}_k^2 - \mathrm{tar}_k) \ , \tag{5.13}$$

$$\delta_j^1 = f'(\mathrm{netin}_j^1)\sum_{k=1}^{n} w_{kj}\delta_k^2 \ . \tag{5.14}$$

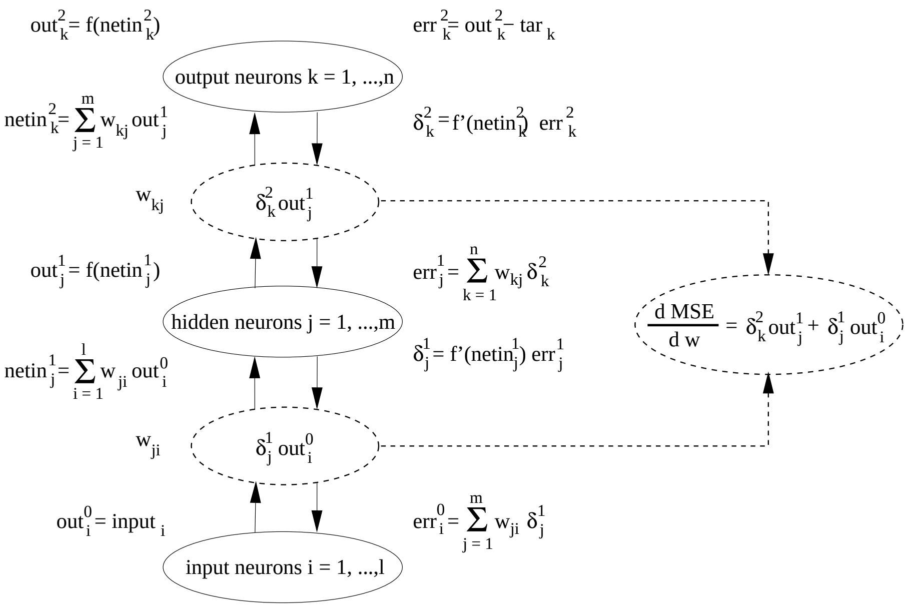
Substituting the auxiliary terms of Eq. 5.13 and Eq. 5.14 in Eq. 5.12 and re-arranging terms yields to

$$\frac{\partial \mathrm{MSE}}{\partial w} = \frac{1}{T}\sum_{t=1}^{T} \delta_k^2 \cdot \mathrm{out}_j^1 + \delta_j^1 \cdot \mathrm{out}_i^0 \ . \tag{5.15}$$

[5]According to the product-rule, the derivative of $y = f(x) \cdot g(x)$ is $\frac{\partial y}{\partial x} = f(x) \cdot \frac{\partial f}{\partial x} + g(x) \cdot \frac{\partial g}{\partial x}$.

As stated in Eq. 5.15, the overall partial derivative of the network error function with respect to a particular weight w consists of *two* blocks: The *first* block describes the local partial derivative $\delta_k^2 \cdot \mathrm{out}_j^1$ of the error function with respect to the weights w_{kj} located in the upper network level. The *second* block refers to weights w_{ji} of the lower network level. Here, the local partial derivative is computed by $\delta_j^1 \cdot \mathrm{out}_i^0$. Thus, the overall partial derivative of the error function with respect to a weight w is computed by summing the local partial derivatives [RHW86, p. 356].

According to the calculations of the partial derivatives (Eq.5.15), the standard backpropagation algorithm (Fig. 5.2) has to be modified. Fig. 5.6 depicts the shared weights extension of the backpropagation algorithm in case of the 3-layer neural network constrained by Eq. 5.11.[6]



$\mathrm{out}_k^2 = f(\mathrm{netin}_k^2)$
$\mathrm{err}_k^2 = \mathrm{out}_k^2 - \mathrm{tar}_k$
output neurons k = 1, ...,n
$\mathrm{netin}_k^2 = \sum_{j=1}^{m} w_{kj}\,\mathrm{out}_j^1$
$\delta_k^2 = f'(\mathrm{netin}_k^2)\ \mathrm{err}_k^2$
w_{kj}
$\delta_k^2\,\mathrm{out}_j^1$
$\mathrm{out}_j^1 = f(\mathrm{netin}_j^1)$
$\mathrm{err}_j^1 = \sum_{k=1}^{n} w_{kj}\,\delta_k^2$
$\frac{d\,\mathrm{MSE}}{d\,w} = \delta_k^2\,\mathrm{out}_j^1 + \delta_j^1\,\mathrm{out}_i^0$
hidden neurons j = 1, ...,m
$\mathrm{netin}_j^1 = \sum_{i=1}^{l} w_{ji}\,\mathrm{out}_i^0$
$\delta_j^1 = f'(\mathrm{netin}_j^1)\ \mathrm{err}_j^1$
w_{ji}
$\delta_j^1\,\mathrm{out}_i^0$
$\mathrm{out}_i^0 = \mathrm{input}_i$
$\mathrm{err}_i^0 = \sum_{j=1}^{m} w_{ji}\,\delta_j^1$
input neurons i = 1, ...,l

Figure 5.6 Shared weights extension of the standard backpropagation algorithm in case of a single training sample.

As shown in Fig. 5.6, the local partial derivatives of the weights w_{kj} and w_{ji} are computed by using information of the forward and the backward pass of the network. This corresponds to the standard backpropagation algorithm (Fig. 5.2). In contrast to the standard approach, the

[6]For simplicity, we omit bias connections to the hidden and the output layer. For the inclusion of bias weights see Bishop (1995) [Bis95, p. 142].

local partial derivatives are stored along the backward pass of the network for each weight separately. After the backward pass is finished, the resulting sum of all local partial derivatives determines the overall partial derivative of the error function with respect to the concerned weight w [RHW86, p. 356].

Having explained the shared weights extension of standard backpropagation on the basis of a simple feedforward neural network, we are now dealing with its application to an unfolding in time neural network. The unfolding in time architecture is similar to the recurrent neural network depicted in Fig. 5.5, right. For simplicity, the unfolding was truncated after *three* time steps t, $t-1$ and $t-2$. The shared weights of this network are (*i.*) $w^A = w^A_{ji} = w^A_{iu}$, (*ii.*) $w^B = w^B_{jh} = w^B_{ig} = w^B_{uw}$ and (*iii.*) $w^C = w^C_{lj} = w^C_{ki} = w^C_{su}$. Fig. 5.7 illustrates the resulting backpropagation algorithm in case of a single training pattern.

As shown in Fig. 5.7, the computation of the partial derivatives is analogous to the case of the shared weights 3-layer neural network (Fig. 5.6). Likewise to the description of Fig. 5.6, we simplified the notations by the auxiliary terms δ. The upper indices correspond to the time steps t, $t-1$ and $t-2$ of the finite unfolding.

The local partial derivatives are derived by a combination of the forward and the backward flow of the unfolding in time network (Fig. 5.7). The overall partial derivatives of the error function with respect to the shared weights w^A, w^B or w^C are obtained by summing the local partial derivatives for each shared weight separately [RHW86, p. 355-56]. The partial derivative of the network error function with respect to the shared weights w^B is composed of *three* local derivatives. These derivatives are collected at the different steps t, $t-1$ and $t-2$ of the unfolding. The same is valid for the shared connector w^C. Since we truncated the unfolding after *three* time steps, the shared connector w^A is only applied twice (i. e. w^A_{ji} and w^A_{iu}). Thus, the overall partial derivative of the error function with respect to the shared weights w^A is only composed of the local derivatives $\delta^t_j \cdot \mathrm{out}^{t-1}_i$ and $\delta^{t-1}_i \cdot \mathrm{out}^{t-2}_u$.

The application of the extended backpropagation algorithm to more sophisticated unfolding in time neural networks, e. g. error correction neural networks [ZNG01a, p. 247-49], is straightforward.

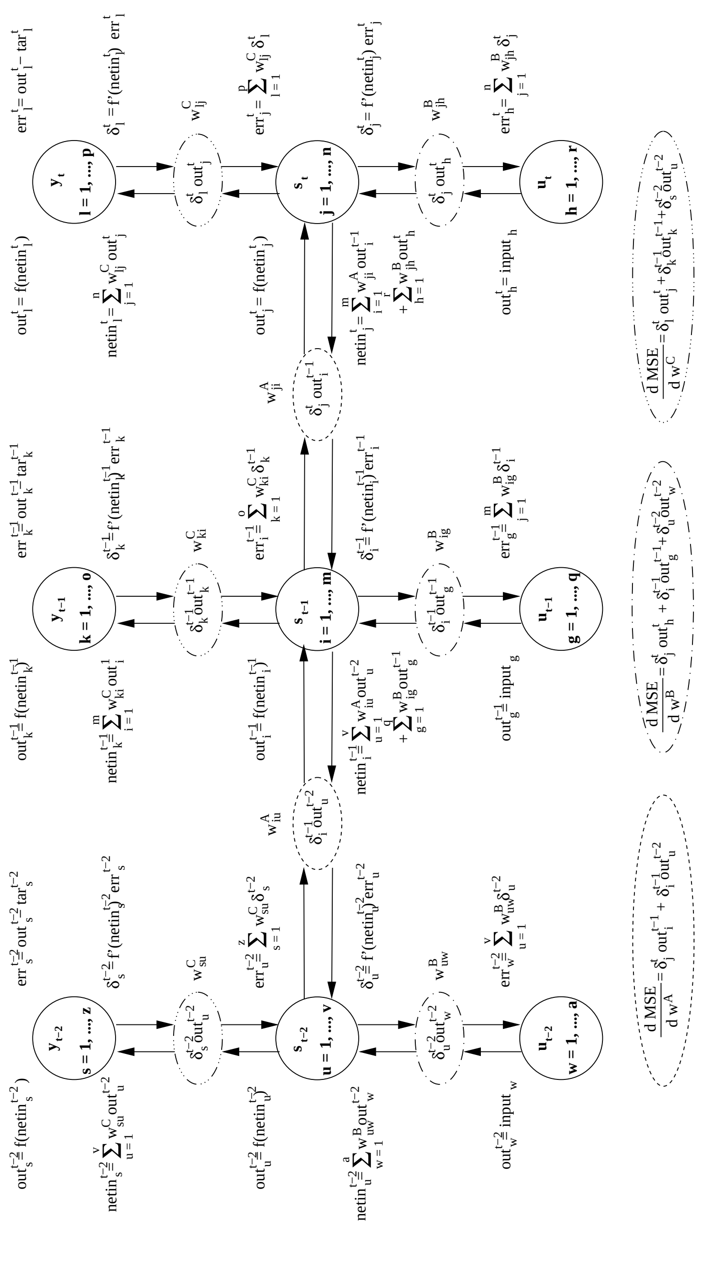


Figure 5.7 The shared weights extension of standard backpropagation in case of a finite unfolding in time neural network with shared weights w^A, w^B, w^C. (The resulting partial derivatives of the error function are depicted at the bottom of the network architecture. For simplicity, (shared) bias connections are omitted.)

4. ESTIMATING THE MAXIMAL INTER-TEMPORAL CONNECTIVITY

In the previous section, we solved the system identification task of a time-delay recurrent neural network by unfolding in time [Hay94, RHW86, ZN01, p. 751-52, p. 354-7 and p. 323-4]. The resulting spatial network architecture can be trained by an extended version of standard backpropagation (sec. 3.). In general, unfolding in time neural networks are approximated by a finite unfolding which truncates the unfolding after some time steps [ZN01, p. 323-5]. Such a preset truncation of the unfolding is depicted in Fig. 5.8, where the finite truncation length is set to *five* time steps t, ..., $t-4$.

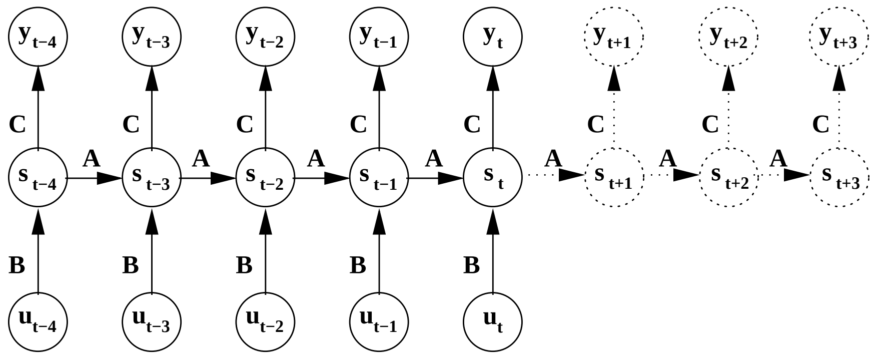


Figure 5.8 Finite unfolding in time truncates the unfolding of a recurrent neural network after a preset amount of time steps. The latter truncation length is determined by the maximal inter-temporal connectivity [ZN01, p. 324]. The dotted parts of the network architecture refer to the so-called overshooting technique, which iterates the autonomous part of the system (coded in matrices A and C) into future direction. Overshooting supports the estimation of the finite truncation length [ZN01, p. 326].

Considering *finite* unfolding in time, an important decision of the modeling is to determine the finite truncation length of the unfolding. In other words, we have to decide how many time steps $t-\tau$ are required to describe the dynamics y_t at the present time state [ZN01, p. 324].

We propose to use the following procedure [ZN01, p. 324]: We start with a preset truncation length of e. g. *five* time steps $t, \ldots, t-4$ (see Fig. 5.8). Now, the final truncation length can be derived by examining the individual model errors $y_{t-\tau}$ along the different time steps of the unfolding. Usually, the model errors are decreasing from left (y_{t-4}) to right (y_t), because more and more memory is accumulated in order to explain the development of the system. For example, the leftmost output y_{t-4} is only calculated on the basis of the external information

u_{t-4}, while y_{t-3} depends on the external influences u_{t-3} *and* the internal state s_{t-4}. Hence, we obtain a superposition of information along the unfolding time steps and the individual model errors decrease until a minimum is achieved. The saturation of the model error determines the maximal length of the unfolding in time. Only the inclusion of these time steps provides valuable information for the description of underlying dynamics at the present time state [ZN01, p. 324]. In the following, we refer to the optimal truncation length of the unfolding as the *maximal inter-temporal connectivity* (MIC).

In our example, we might discover, that the saturation of the error level takes places at y_{t-1}. Thus, we can omit the last time step $(t-4)$ of the unfolding. In other words, we achieve an appropriate description of the dynamics by accumulating a memory over the last *four* time steps $t-3, \cdots, t$. In contrast, if the individual model errors do not saturate within the considered unfolding time steps $(t, \ldots, t-4)$, we would restart with a broader expansion, e. g. up to $t-6$ [ZN01, p. 324].

In this connection, one may think of the development of so-called *early warning indicators.* By estimating the maximal inter-temporal connectivity, we try to discover long-term dependencies between the model targets y_τ and the external influences $u\tau$. More precisely, we are interested in finding the first causal connection of the external inputs and the model targets, which supports the description of the underlying dynamics at the present time state t. This idea corresponds to the concept of early warning indicators. Here, a serious change in one of external influences should be detected as early as possible in order to estimate the future development of the dynamical system [ZN01, p. 327-28]. Especially in economics, early warning indicators are vitally important for the analysis of financial markets or the evaluation of emerging countries [ZN01, p. 328]. This in mind, the estimation of the MIC can also be interpreted as a *first cause analysis.*

Interestingly, we can enforce the detection of long-term dependencies in the data by the application of *overshooting.* Overshooting is the extension of the autonomous recurrence in future direction $t+1, t+2, \cdots$ [ZN01, p. 326-27]. Due to the error signals of the additional model outputs, overshooting enforces the autonomous part of the dynamics and acts as a regularization for the learning [ZN01, p. 326-28]. An example of the overshooting concept is depicted in Fig. 5.8. The overshooting part of the network (Fig. 5.8, dotted unfolding time steps) extends the autonomous part of the dynamics three time steps in future direction $(t+1, \ldots, t+3)$. By the iteration of the shared weight matrices A and C we enhance the forecast horizon of the network and obtain a series of predictions, y_{t+1}, y_{t+2} and y_{t+3}.

One may argue, that an unfolding neural network (Fig. 5.8, without the dotted overshooting connections) already establishes concatenations between the external influences and the model targets. However, this is misleading, because learning by backpropagation typically focuses primarily on the most recent inputs in order to describe the dynamics. This is due to the fact, that the fastest adaptation takes place in the shortest path between input and output [ZN01, p. 327]. In other words, backpropagation learning has a tendency to emphasize the importance of the nearest inputs u located e. g. at the present time step t. Only if the learning is unable to extract additional valuable information out of the nearest inputs, more distant input signals are evaluated [ZN01, p. 326-27]. From this it follows, that the unfolding in time network (Fig. 5.8, without the dotted overshooting part) "tries to rely as much as possible on the part of the dynamic which is driven by the most recent inputs $u_t, \cdots, u_{t-\tau}$" [ZN01, p. 327]. In contrast, overshooting enforces the learning of the autonomous dynamics due to the additional sequence of future outputs (Fig. 5.8, dotted part). By this, we also support the detection of long-term dependencies between the model targets and the external influences.

It is important to note, that the application of overshooting, i. e. the maximum number of autonomous iterations, is constrained by the MIC [ZN01, p. 326-27]. In case, the sequence of future outputs $y_{t+\tau}$ does *not* exceed the estimated truncation length, we propose the following procedure: First, we train the model until convergence. Second, if we can observe over-fitting on the training set, we enhance the overshooting by an additional output $y_{t+\tau+1}$ and train the network again until convergence. Interestingly, we made the following observation: If the network is able to learn the additional forecast horizon $y_{t+\tau+1}$ as well, the error level of all the other outputs $y_t, \cdots, y_{t+\tau}$ is also decreased, because more useful information in form of error signals is backpropagated to the weights [ZN01, p. 326]. We stop this iterative process of extending the overshooting, when the error level saturates or even starts to increase [ZN01, p. 326]. In our experiments we found, that the MIC can be larger than the length of the overshooting [ZN01, p. 327].

Due to shared weights, the described extensions do not enlarge the number of network parameters. Thus, the application of an unfolding in time neural network in combination with overshooting is most suitable for problems with a small training data sets. Typically, this is true for macroeconomic analysis. Since overshooting networks have a small number of free parameters and an enhanced error signal flow, they seem to be very appropriate for such applications [ZN01, p. 326-28].

5. PATTERN-BY-PATTERN & VARIO-ETA LEARNING RULE

Backpropagation is only an efficient way to calculate the partial derivatives of the network error function with respect to the weights [Zim94, p. 38]. Now the question arises, how the information of the gradients can be used to adapt the weights such that the network is optimized, i. e. the overall network error becomes minimal. Typically, weight changes are performed in accordance with a particular learning rule, which utilizes the before-hand calculated partial derivatives [Zim94, p. 40].

Supervised learning is the task of minimizing the difference between the network output y and corresponding observations y^d across a given set of training patterns $t = 1, \dots, T$ [Zim94, Hay94, p. 36 and p. 63-4]. The comparison between the network output y and observed target values y^d is drawn on the basis of an error function E. For example, one may refer to the mean square error (Eq. 5.1) or to the robust cost function $\ln\cosh(.)$ (Eq. 5.9). Weight modifications are performed in order to minimize the assumed error function E. By the minimization of the error function E, we want to identify time invariant structures in varying time series and expect to achieve a good generalization performance of our model [Zim94, ZGT02, p. 36 and p. 409].

In general, the optimization of the neural network overall error function is a high-dimensional problem which comes along with a high degree of nonlinearity [Zim94, p. 41-42]. Under such conditions, learning is an iterative search through the solution space by adjusting the weights according to a certain learning rule [Zim94, p. 40]. During the learning procedure, weight changes from step τ to $\tau + 1$ can be performed as follows [NZ98, p. 395]:

$$w^{(\tau+1)} = w^{(\tau)} + \eta \cdot \Delta w^{(\tau)} . \tag{5.16}$$

As stated in Eq. 5.16, a weight change from iteration τ to $\tau + 1$ is determined by the variation of the particular weight $\Delta w^{(\tau)}$ and the increment of the learning η. The weight variation $\Delta w^{(\tau)}$ describes the search direction [Zim94, p. 40]. The parameter η is the length of a learning step along the search direction $\Delta w^{(\tau)}$ (often referred to as *learning-rate*) [Zim94, p. 40]. The learning increment η is usually held constant or follows a specific annealing procedure [CheM98, NZ98, p. 262-63 and p. 395-96]. It should be chosen on the constraint to be sufficiently small in order to guarantee, that the learning is able to converge in a local minimum [Zim94, p. 41].

The search direction $\Delta w^{(\tau)}$ is determined by a particular learning rule [NZ98, p. 395]. These learning algorithms are typically classified in two categories: (*i.*) first-order and (*ii.*) second-order optimization methods [ZaRe99, p. 27]. *First-order* algorithms utilize various kinds of *gradient descent*, which is probably the easiest learning rule [Zim94, ZaRe99, p. 41 and p. 27]. *Pattern-by-Pattern* as well as *vario-eta* learning belong to the class of first-order algorithms and thus, both are in some way related to gradient descent [Zim94, NZ98, p. 41 and p. 396]. *Second-order* algorithms examine the first *and* second partial derivatives in order to determine a minimum of the error function. Examples of second-order methods are *quasi-Newton* or *variable metric* algorithms [ZaRe99, p. 27]. A detailed overview of such algorithms can be found in Polak (1991) [Pol91]. However, due to methodical and computational reasons [NZ98, ZaRe99, p. 395-404 and p. 27], we propose the usage of *first-order* algorithms.

5.1 GRADIENT DESCENT

In gradient descent, the search direction $\Delta w^{(\tau)}$ is defined by the negative cumulated gradient g of the error function with respect to the weights. The partial derivatives of the error function E, which are needed to calculate the gradient g_t for a single training pattern $t = 1, \ldots, T$, are provided by the backpropagation algorithm. Typically, the gradients g are calculated over the complete training set [Zim94, p. 41]. In these so-called batch versions of gradient descent the search direction $\Delta w^{(\tau)}$ from step τ to $\tau + 1$ of the learning procedure can be written as

$$\Delta w^{(\tau)} = -g = -\frac{\partial \mathrm{E}}{\partial w} = -\frac{1}{T}\sum_{t=1}^{T}\frac{\partial \mathrm{E}_t}{\partial w}\,. \tag{5.17}$$

By Taylor expansion of the error function up to the second-order, it can be shown, that gradient descent leads to a decreasing value of the network error [Zim94, p. 41]. The gradient descent algorithm can be exercised as long as the calculated gradient g is not zero. In this case, the error of the network has converged to a (local) minimum. Since the gradients are calculated with respect to the complete training set, this procedure is often referred to as *cumulative gradient* approach [Zim94, p. 41].

One of the major disadvantages of *gradient descent* is that the learning is often caught in a sub-optimal local minimum, because the gradients navigate the learning to the nearest reachable minimum without noticing its possible inefficiency [Zim94, p. 41-2]. In addition, the learning-rate η has to be sufficiently small in order to ensure an improvement of the learning [Zim94, p. 41]. Other first-order approaches try to avoid these disadvantages by modifying the search direction $\Delta w^{(\tau)}$ [NZ98, p. 395-7].

5.2 PATTERN-BY-PATTERN LEARNING

Following the concept of gradient descent, the *pattern-by-pattern* learning rule adjusts the weights after the evaluation of *each* training sample [NZ98, p. 396]. The training samples are often chosen at random from the data set in order to support a broader exploration of the solution space [Zim94, NZ98, p. 42-3 and p. 396]. For a single training pattern t, the search direction $\Delta w^{(\tau)}$ of *pattern-by-pattern* learning is

$$\Delta w^{(\tau)} = -g_t = -\frac{\partial \mathrm{E}_t}{\partial w}\,. \tag{5.18}$$

Interestingly, learning pattern-by-pattern or with small batch sizes N, i. e. the gradient calculations are based on a subset N of all training samples, can be viewed as a stochastic search process [Zim94, NZ98, p. 42 and p. 397]. – To explain this, let us re-formulate the search direction $\Delta w^{(\tau)}$ by utilizing the cumulative gradient g:

$$\begin{aligned}
\Delta w^{(\tau)} &= -\frac{1}{N}\sum_{t=1}^{N} g_t = -\frac{1}{N}\sum_{t=1}^{N}(g_t) + g - g \\
&= -g - \frac{1}{N}\sum_{t=1}^{N}(g_t - g) \\
&= -\frac{\partial \mathrm{E}}{\partial w} - \left[\frac{1}{N}\sum_{t=1}^{N}\left(\frac{\partial \mathrm{E}_t}{\partial w} - \frac{\partial \mathrm{E}}{\partial w}\right)\right].
\end{aligned} \tag{5.19}$$

The adaptation of a stochastic search process in the solution space is due to inconsistencies of the training data. As stated in Eq. 5.19, the search direction $\Delta w^{(\tau)}$ is composed of two blocks: (*i.*) the cumulative gradient g, which induces a drift to a local minimum, and (*ii.*) the deviations of the gradients g_t (calculated with single training patterns t) from the cumulative gradient g. The deviations $(g_t - g)$ disturb the drift g. The noise term $(g_t - g)$ is *zero* on the average of the chosen batch, because the cumulative gradient g is the average of the single gradients g_t [Zim94, NZ98, p. 42 and p. 397-98].

If the learning is close to a local minimum, the cumulative gradient g disappears per definition. In contrast, the deviations $(g_t - g)$, which disturb the drift g, remain due to the inconsistencies of the training samples. As an advantage, the learning is able to escape from a sub-optimal local minimum [Zim94, p. 42]. Note, that the deviations $(g_t - g)$ become more equally distributed if the training samples are chosen randomly out of the training set [Zim94, NZ98, p. 42-43 and p. 396-97].

5.3 VARIO-ETA LEARNING

Vario-eta learning introduces a variable length to the learning rate η [Zim94, p. 48-49]. By this, we achieve an individual scaling of the weight changes [NZ98, p. 396]. The search direction $\Delta w^{(\tau)}$ of *vario-eta* learning is given by

$$\Delta w^{(\tau)} = -\frac{1}{\sqrt{\sum_{t=1}^{N}(g_t - g)^2}} \cdot g_t = -\frac{1}{\sqrt{\sum_{t=1}^{N}\left(\frac{\partial \mathrm{E}_t}{\partial w} - \frac{\partial \mathrm{E}}{\partial w}\right)^2}} \cdot \frac{\partial \mathrm{E}_t}{\partial w} \qquad (5.20)$$

Multiplying the search direction of $\Delta w^{(\tau)}$ of *vario-eta* learning by the learning-rate η, each weight w is changed by an individual increment $\eta \cdot \Delta w^{(\tau)}$ (see also Eq. 5.16). This is due to the fact, that the learning rate η is scaled by the standard deviation of the squared deviations $(g_t - g)$ [NZ98, Zim94, p. 396-97 and p. 48-49]. Our experiments indicate that *vario-eta* learning is most useful in combination with small batch sizes N of e. g. 10% of the training set [NZ98, p. 396].

The additional scaling of the learning-rate has a true benefit [NZ98, p. 396-7]: If the weight increments $\eta \cdot \Delta w^{(\tau)}$ calculated for single training samples are similar to each other, the denominator in Eq. 5.20 becomes small and thus, the rescaled learning-rate is enlarged. In this case, the gradients g_t of single training samples t are nearly equal to the cumulative gradient g. In other words, the variation of the gradients g_t around the cumulative gradient g is small. On the other hand, if the individual weight increments $\eta \cdot \Delta w^{(\tau)}$ are inhomogeneous (i. e. the gradients g_t fluctuate heavily around the cumulative gradient g), the denominator in Eq. 5.20 becomes large. Hence, the rescaled learning-rate is down-sized. In the latter case, the training data might be inconsistent or the search path leads through a narrow valley in the solution space [Zim94, p. 49].

Due to the rescaling, *vario-eta* learning behaves like a stochastic approximation of a *quasi-Newton* method [NZ98, p. 396]. To illustrate this similarity, let us consider a search path which leads through a narrow valley in the solution space [NZ98, p. 396-7]. Such a situation is shown in Fig. 5.9 in case of a two dimensional solution space.[7] As depicted in Fig. 5.9, gradient descent (Eq. 5.17) would follow a 'zigzagging' track through the narrow valley (Fig. 5.9, left). Hence, gradient descent converges very slowly to the minimum. In contrast, *vario-eta* learning damps the 'zigzagging' along w_2 and accelerates the drift along w_1 simultaneously (Fig. 5.9, right) [NZ98, p. 396-97]. "This behavior

[7]Fig. 5.9 is taken from Neuneier and Zimmermann (1998) [NZ98, p. 396].

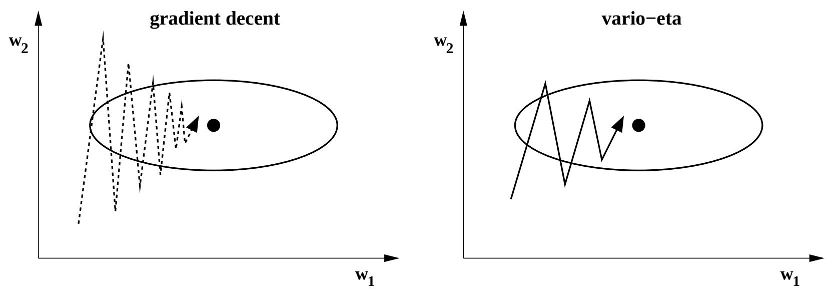


Figure 5.9 Approaching a narrow valley in a two dimensional solution space: Learning path of gradient descent, left, and of *vario-eta* learning, right.

is similar to the weight trajectories classical Newton methods show in such a situation" [NZ98, p. 396].

As a further advantage, *vario-eta* learning allows the training of large neural networks. As we have already mentioned (sec. 2.), backpropagating the error signals through long sequences of hidden layers equipped with hyperbolic tangent squashing leads to a leakage of error information [MJ99, p. 134-5]. However, the rescaling in Eq. 5.20 is able to refresh the learning information for each weight no matter where it is located in the network. By this, the continuous shrinking of the error information through long hidden layer transformations is prevented [NZ98, ZN01, p. 397 and p. 325].

Rescaling the gradients at each step of the backpropagation algorithm, *vario-eta* learning makes it also possible to discover long-term intertemporal dependencies in unfolding in time neural networks (Fig. 5.5, right), which often contain long chains of tanh(.) squashing functions [ZN01, p. 325 and p. 327-8]. Thus, *vario-eta* learning implicitly supports the estimation of the MIC. The empirical study (sec. 7.), illustrates this effect by a comparison of *vario-eta* and *pattern-by-pattern* learning.

6. EXPLORING INVARIANT STRUCTURES IN TIME

Besides the integration of prior knowledge or the implementation of first principles (e. g. variants-invariants separation, see chp. 4), learning from data, i. e. the optimization of the neural network, is also a major issue of the model building. In this section, we refocus on the learning of neural networks. Based on standard error backpropagation, we develop a new learning rule which evaluates only specific gradients during the training. We call this approach *partial learning.*

Training neural networks, we are interested in finding time invariant structures out of varying time series. These structures allow us to predict the future development of our dynamical system. Standard learning algorithms like error backpropagation [Wer94] in combination with a particular learning rule for changing the weights w generate a hypothesis of such a time invariant structure. The task of forecasting the future by an analysis of past data implies therefore an invariance hypothesis [NZ98, p. 413-17]: Fitting our model $y_{t+1} = f(x, w)$ to observed data x of the past, forecasting is only feasible, if we assume the persistence of the explored structures, i. e the structures have to be time invariant.

Unfortunately, a pure fitting of the data often leads to unsatisfactory forecasting results, because the underlying dynamics may drift over time. Furthermore, overfitting, i. e. learning by heart of false causalities, is a well-known problem [Zim94, p. 58-59]. Due to these difficulties, the learning might be misleading in the sense, that we get a local minimum of the error function covering the underlying structural instabilities. Thus, the time invariant structures which are created during the learning have to be challenged. The neural network topology represents only a hypothesis of the true underlying structure. In other words, we need to revise the invariance hypothesis constructed by the learning in an additional step in order to find structural instabilities. Only if we cannot falsify the generated hypothesis about the time invariant structures, the forecasts of our model are reliable [NZ98, p. 413-17].

Instead of referring to traditional techniques (e. g. early stopping [Zim94, p. 60-61]), we propose a learning procedure which consists of two parts [ZGT02, p. 409-11]: First, we optimize the beforehand designed neural network until the error on a preset training set converges, i. e. a minimal training error is achieved. We propose to training the neural network by standard error backpropagation in combination with *vario-eta* learning rule, which can be seen as a fast stochastic approximation to a Newton method. Second, we focus on the structures discovered by the learning and look for structural instabilities in order to extract an appropriate generalizing model. We suggest to use weight pruning techniques like e. g. stochastic pruning, early brain damage, optimal brain damage or instability pruning in order to revise the structural hypothesis of the learning [NZ98, ZGT02, p. 405-410, 413-17 and p. 409-10]. The two model building steps can be seen as the generation of a structural hypothesis followed by a falsification of the generated structure [ZGT02, p. 409-11]. We will explain these model building steps more detailed in this section.

6.1 STOCHASTIC PRUNING

Stochastic pruning can be seen as a *t-test* on the network weights [NZ98, p. 405-06]. The test value for a particular weight w is given by

$$\text{test}_w = \frac{|w+g|}{\sqrt{\frac{1}{T}\sum(g_t-g)^2}} \ , \quad \text{with} \quad g = \frac{1}{T}\sum_t g_t \ . \tag{5.21}$$

According to this test criterion, the size of a particular weight is divided by the standard deviation of its fluctuations over time. Weights with low test values are pruned out of the network. By this, we prune weights which have a questionable liability and thus, a stabilization of the learning against resampling of the training data is obtained. Since gradients of larger weights do not fluctuate enough to give useful test values, the stochastic pruning includes a bias to nonlinear models.

6.2 EARLY BRAIN DAMAGE (EBD)

EBD is based on the well-known optimal brain damage (OBD)[8] pruning method [TNZ97, NZ98, p. 670-72 and p. 406-7]. The test value of EBD for a particular weight w is "an approximation of the difference between the error function for $w = 0$ versus the value of the error function for the best situation [w_{min}] this weight can have" [NZ98, p. 406]:

$$\text{test}_w = E(0) - E(w_{\text{min}}) = -gw + \frac{1}{2}w'Hw + \frac{1}{2}gH^{-1}g' \ . \tag{5.22}$$

The above mentioned approximation can be derived from a Taylor expansion of the network error function [NZ98, p. 406]. The Hessian H in Eq. 5.22 is computed in the same way as in the original OBD pruning algorithm. For further details see Tresp, Neuneier and Zimmermann (1997) [TNZ97, p. 670-72]. Likewise to stochastic pruning, EBD favors weights with low fluctuations over time. As it can be seen from the test value, EBD implicitly tests the importance of a weight with regards to the network error function. Since the network error is also used by error backpropagation, "EBD cancels out only weights which are not disturbing the learning" [NZ98, p. 406-7].

6.3 INSTABILITY PRUNING

Instability pruning uses an important property of the minimal training error [ZGT02, p. 409-10]. If the learning has reached a local minimum,

[8]OBD was firstly introduced by Le Cun et al. (1990) [LDS90, p. 600-02]. For a comprehensive description of OBD see also Bishop (1995) [Bis95, p. 360].

the cumulative gradient g of each weight is approximately zero on the training set $t = 1, \dots, T$:

$$g = \frac{1}{T} \sum_{t}^{T} g_t = \sum_{t}^{T} \frac{1}{T} g_t \approx 0 \ . \tag{5.23}$$

If this criteria is still valid, in case we apply a time varying weighting to the gradients, the invariants hypothesis is *true* and our model is perfectly stable. The weighting is done by emphasizing the near to present time gradients on the training set:

$$\sum_{t}^{T} \alpha_t g_t = 0 \ , \quad \text{with} \quad \sum_{t}^{T} \alpha_t = 1 \tag{5.24}$$

In Eq. 5.24, we define a weighted average, which favors the most recent gradients on the training set by using a weighting factor $\alpha_t = \alpha t$. We propose to choose a normalized weighting factor $\alpha_t = 2t/(T(T+1))$.

Having trained the neural network until convergence, the test value of instability pruning (Eq. 5.25) is now computed for a single weight w by comparing the weighted and unweighted criteria of the gradients. Note, that we use the normalized weighting factor α_t to calculated the test value. The constant ϵ prevents numerical difficulties.

$$test_w = \frac{\epsilon + \left| \dfrac{1}{T} \displaystyle\sum_{t}^{T} \dfrac{2t}{T+1} g_t \right|}{\epsilon + \left| \dfrac{1}{T} \displaystyle\sum_{t}^{T} g_t \right|} \tag{5.25}$$

If we have a stationary distribution of the gradients g_t, the *weighted* cumulative gradient is similar to the cumulative gradient (see Fig. 5.10, left). In this case, the test value of instability pruning (Eq. 5.25) would be close to 1. On the other hand, drifting structures over time cause instable gradient distributions (see Fig. 5.10, right). In this case, the test value of Eq. 5.25 would be larger than 1. We use this measure as an indicator of instabilities in order to test the invariant hypothesis.

We suggest to apply the following training procedure [ZGT02, p. 410]: Train the network until the weights are near a local optimum and compute the test values of Eq. 5.25. Afterwards, take out some of the most unstable weights as measured by this test criterion and restart the learning of the network. This iterative procedure should be terminated, if the network error measured on a certain validation set tends to increase.

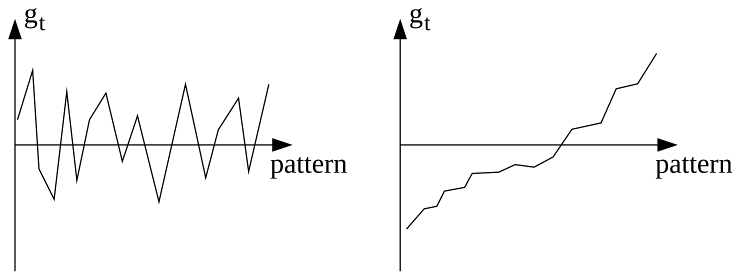


Figure 5.10 The gradients of valuable weights have a stationary distribution (left Figure). In contrast, gradients of instable weights are characterized by a non-stationary distribution (right Figure). Instable weights can be pruned out of the network by instability pruning. We test the stability of the local minima by applying a time varying weighting to the gradients (Eq. 5.25)

Have in mind, that this pruning technique does not use a validation set for the calculations of the test values. In principle, the test value of Eq. 5.25 can also be smaller than 1. In this case, the system is even more stable in the most recent time and thus, the associated weights should not be pruned out of the network.

Opposite to early brain damage (see Eq. 5.22) or optimal brain damage, instability pruning does not have a bias towards linear models. Another difference is, that instability pruning does not focus on the network error function. Hence, cancelling out weights by instability pruning may have a large impact on the overall network error. However, instability pruning supports time invariant structures. – This is a feature which is not touched by the cost function alone. In other words, instability pruning introduces a new feature in the model building process: stability over time.

6.4 PARTIAL LEARNING

As a further improvement of the network structure in the sense of sparse network architectures, we propose a new learning rule, which is called *partial learning* [ZGT02, p. 410-11]. To explain partial learning, let us briefly revisit standard error backpropagation. Standard error backpropagation is an efficient way to compute the partial derivatives of the network error function with respect to the weights. Weight modifications occur in accordance with a certain learning rule, e. g. *pattern-by-pattern* or *vario-eta* learning.

A drawback of using standard learning rules like e. g. *pattern-by-pattern* or *vario-eta* learning together with error backpropagation is that a *distributed* representation of the input to output relationship is generated. This means that *all* parts of the network contain some relevant information about the underlying dynamics. In other words, each weight of the network is partly responsible for the explanation of the overall network error. Being so, we get a non-local coding of our information in the network structure.

Unfortunately, due to the distributed structures, the falsification of the generated invariance hypothesis by pruning methods is complicated. In addition, structural redundancies in different parts of the network make the task of finding structural instabilities even more difficult.

As a remedy, we propose to support the learning of localized structures during the error backpropagation. We achieve the generation of localized structures by using *only* the $p\%$ largest gradients of each connector for the modification of the network weights. Proceeding this way, the learning effects only sub-structures of the network. With η as the learning rate, the weight adaption rule of *partial learning* can be formulated as

$$\begin{aligned} \hat{w} &= w + \eta f(p,g) g_t \ , \\ \text{with} \quad f(p,g) &= \left\{ \begin{array}{ll} 1 & \text{for the p\% largest gradients } |g_t| \\ & \text{in each connector} \\ 0 & \text{otherwise} \end{array} \right\} . \end{aligned} \tag{5.26}$$

According to Eq. 5.26, a weight w is updated to a new value $\hat{w}$, if its associated gradient g_t is one of the $p\%$ largest gradients of the concerned connector. Proceeding this way, we neglect the standard implication of error backpropagation, that the gradients which are created for a specific training pattern are able to evaluate *all* connections of the network. Partial learning focuses only on weights being in charge for a specific training pattern. As a result, the learning of single training patterns changes only parts of the network and thus, localized structures are preferred. Each weight is only partially effected by the (backpropagated) input to output error of the network. Since partial learning enforces the building of localized structures, the network acts like a "mixture of experts".

The reason for choosing only the $p\%$ largest gradients of each connector for updating the weights is that large gradients contain the most relevant information about the (time invariant) structure of the underlying problem. These gradients are also responsible for creating raw structures in the beginning of the learning. In addition, large gradients have the highest impact on the network in the sense of structural changes, because they are always associated with large values of the error function. Large backpropagated error signals are an indicator of important mismatches of the model, which have to be handled. As a side effect, using large gradients is important to generate highly non-linear structures, because these gradients significantly increase the weights such that the non-linear parts of the squashing functions (e. g. tanh(.)) are reached.

On the other hand, one may argue, that large gradients are often caused by outliers in the data set and are therefore non-constructive for learning of time invariant structures. On this problem, we propose to use a (network internal) outlier preprocessing together with a robust error function, e. g. $\ln\cosh(.)$ [NZ98, p. 375-6 and 387-8].

In contrast to large gradients, small gradients contain only insufficient structural information. During the learning, these gradients act more or less as noise on the weights. Being so, the learning is disturbed. Therefore, these gradients should be omitted during the learning. Likewise to small gradients, medium gradients provide no clear information about the structure. These gradients often occur near local minima. Since the structural information of medium gradients is not clear, we propose to suppress these gradients as well.

Our experiments indicate, that choosing even 90% of the small gradients for the learning of the weights, leads to a poor fitting on the training set and the efficiency of the learning is handicapped. In contrast, choosing only 10% of the largest gradients for the weight modifications, allows a perfect fitting of the training data. In addition, there is no loss on learning efficiency. We found, that partial learning should not be applied to output and bias connections of the neural network.

7. EMPIRICAL STUDY: GERMAN INTEREST RATE FORECASTING

This section examines a simple empirical example, in which we illustrate the estimation of the MIC. The problem is to forecast the weekly development of the 3-month German inter-banking offered interest rate.

For this purpose, we utilize a time-delay recurrent neural network (Eq. 4.5). The corresponding system identification task is solved by unfolding in time. We employed a spatial neural network architecture similar to Fig. 5.5, right. The truncation length of the *finite* unfolding in time is preset to *eight* past time steps ($t-8$), because we are interested in finding long-term dependencies in the data. The recurrent neural network contains an overshooting sequence of four future time steps [ZN01, p. 326-27]. Hence, we obtain the the forecasts y_{t+1}, y_{t+2}, y_{t+3} and y_{t+4} of the 3-month German interest rate. We trained the network until convergence by using (*i.*) *vario-eta* and (*ii.*) *pattern-by-pattern* learning. In order to study the behavior of *vario-eta* and *pattern-by-pattern* learning, the neural network was trained until convergence by using the algorithms separately from each other in two different passes. The network weights are randomly initialized before the application of each algorithm.

We work on the basis of weekly data in order to forecast the development of the 3-month German interest rate one week ahead. The data set ranges from March 1991 to Aug. 1997. The neural network parameters are estimated by using a training set from March 1991 to Dec. 1995. The generalization set covers the time period from Jan. 1996 to Aug. 1997. Besides the short-term German interest rate, the basic data used in this exercise are e. g. stock market indices (USA, Japan, Germany), foreign exchange (Yen, US-Dollar), commodity prices (Oz. Gold, CRB-Future Index) and other short- and long-term interest rates (USA, Japan, Germany). Except the 3-month German interest rate, these inputs are used as external influences u_τ in the neural network (Fig. 5.5).

Dealing with state space models, no complicated preprocessing of the raw input data is needed, because we can utilize memory effects [ZN01, p. 324-25]. Hence, the preprocessing of the raw input data is basically done by using an indicator for the momentum (relative change) of each time series. This preprocessing of the raw input data is also useful to eliminate exponential trends which, for example, may be caused by inflationary influences [NZ98, PDP00, p. 374-5 and p. 97].

Using neural networks with hyperbolic tangent tanh(.) as a squashing function, the numerics works best if the concerned variables fluctuate around zero. This fits best to the finite state space $(-1;1)^n$ created by the tanh(.) nonlinearity [ZNG01a, p. 247]. Thus, we included a *scaling* of the transformed input data. The resulting input signals have a mean value of zero and a statistical variance of one [NZ98, PDP00, p. 374-5 and p. 73].

In order to determine the truncation length of the finite unfolding in time, we start off with a preset length of *eight* time steps $(t-8)$. By examining the individual model errors at the different time steps of the unfolding, we can obtain the *MIC*: The saturation of the model error determines the maximum amount of unfolding time steps, which are required to describe the dynamics at the present time state. Fig. 5.11 depicts the development of the model errors at the different time steps of the unfolding in case of *pattern-by-pattern* learning, left, and *vario-eta* learning, right.

Both learning algorithms show a comprehensible formation of the individual model errors at the different time steps of the unfolding (Fig. 5.11): The model errors decrease from the leftmost output y_{t-8}

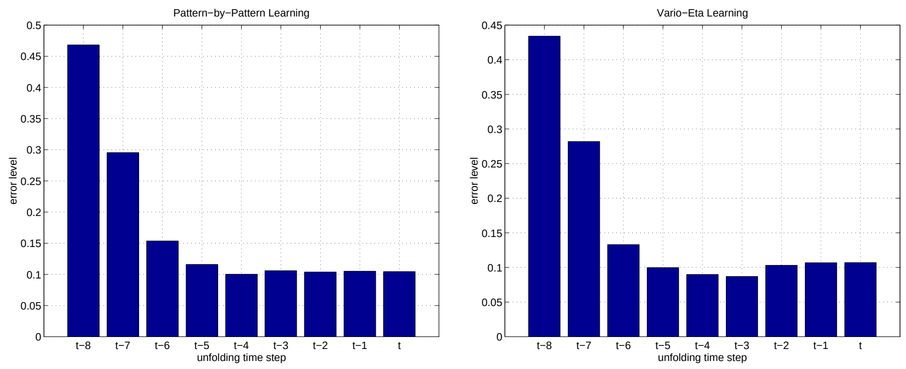


Figure 5.11 Estimating the MIC: Development of the individual error levels y_{t-8}, $\ldots$, y_t using *pattern-by-pattern* learning, left, and *vario-eta* learning, right.

to the right, because we obtain a superposition of more and more information. Due to this accumulation of information, we are in a position to explain the subsequent outputs y_{t-7}, y_{t-6}, $\ldots$ more accurately.

In case of *pattern-by-pattern* learning, we observe a saturation of the error level at time step $t-4$ (Fig. 5.11, left). In accordance with *pattern-by-pattern* learning, we would determine *five* time steps as the maximum amount of unfolding time steps, which support the description of the dynamics at the present time state. Thus, only the time steps t to $t-4$ of the unfolding are required to describe the dynamics at the present time state and we can omit time steps $t-8$ to $t-5$.

Training the unfolding in time network by *vario-eta* instead of *pattern-by-pattern* learning, the saturation of the error level takes place at time step $t-3$. Consequentially, the derived truncation length of the unfolding is *six* time steps, i. e. the MIC is by one time step larger. This indicates, that we achieve an appropriate description of the underlying dynamics at t by embracing the time steps $t-5$ to t in the unfolding (Fig. 5.11, right).

This phenomenon is due to the fact, that *vario-eta* learning includes a rescaling of the gradients. The rescaling avoids the shrinking of the error signals through long chains of hidden layers equipped with hyperbolic tangent squashing [NZ98, ZN01, p. 397 and p. 325]. Hence, we are able to discover additional long term inter-temporal structures in the data, which might support our task of forecasting the short-term German interest rate.

The benefit of an accurate estimation of the finite truncation length become more obvious by comparing the learning algorithms on the basis of a common model evaluation criteria. For this purpose, we utilize the accumulated return of investment, which results from a simple trading strategy referring to the respective model forecasts [Ref95, NZ98, p. 70-1 and p. 419]. Fig. 5.12 depicts the accumulated model return across the generalization set using *vario-eta* resp. *pattern-by-pattern* learning for the training of the neural network.

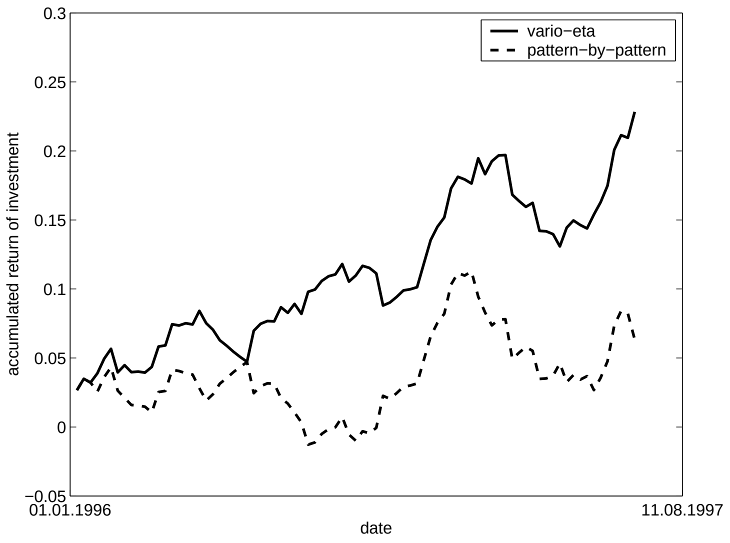


Figure 5.12 Comparison of resulting accumulated model returns using (*i.*) *vario-eta* and (*ii.*) *pattern-by-pattern* learning for the network training. The accumulated return is measured on the generalization set.

As shown in Fig. 5.12, the time-delay recurrent neural network trained by *vario-eta* learning is able to achieve a higher accumulated return of investment on the generalization set than the network trained by *pattern-by-pattern* learning. In contrast to *pattern-by-pattern* learning, *vario-eta* enables us to exploit long-term dependencies in the data in an extended way, such that the model performance is improved. Interestingly, we found, that the additional knowledge of long-term dependencies in the data is especially useful during side-wards moving markets, e. g. in the middle of the generalization set (July 1996 – Dec. 1996). Here, the network trained by *vario-eta* is in a better position to explain the underlying dynamics of the short-term German interest rate and thus, is able to achieve a higher profit.

8. PRELIMINARY CONCLUSION

We considered the standard backpropagation algorithm for the training of feedforward neural networks [Wer74, Wer94]. Error backpropagation is an efficient method to compute the partial derivatives of the network error function with respect to the weights. The shared weights extension of standard backpropagation enables us to train time-delay recurrent neural networks by transferring the system identification task into a spatial architecture [RHW86, Hay94, p. 354-57 and p. 751-52].

Especially in financial applications, the observations are often covered with noise and contain outliers. In these cases, the learning of the network parameters might be interferenced and thus, the model forecasts become unreliable. On this problem, we suggest the usage of the robust error function $\ln\cosh(.)$, which alleviates the effects of outliers in the data set. In addition, the handling of outliers is supported by the application of hyperbolic tangent non-linearities as squashing functions. Small values pass nearly unchanged through this non-linearity, while outliers are damped by the saturation level [NZ98, p. 375-6].

Two learning rules, namely *pattern-by-pattern* and *vario-eta* learning are introduced, which utilize the partial derivatives calculated by the error backpropagation algorithm. In accordance with these learning algorithms, weight changes are performed in order to minimize the network error function on a set of training data. *Pattern-by-pattern* learning is closed to the concept of *gradient descent*, while *vario-eta* behaves like a stochastic approximation of a *quasi-Newton* method [NZ98, p. 396-7].

The training of neural networks is especially supported by partial learning, which generates localized time invariant structures. By this, we also support the optimization of the network by weight pruning methods like e. g. EBD, OBD or instability pruning.

Modeling dynamical systems by recurrent neural networks using unfolding in time, the *finite* truncation length of the unfolding has to be determined in order to solve the system identification task conveniently. By examining the individual model errors $y_{t-\tau}, \ldots, y_t$ at the different time steps of the unfolding, the finite truncation length can be derived from the saturation level of the individual error signals. In this context, *vario-eta* learning supports the estimation of the MIC due to re-normalization of the gradients at each step of the backpropagation. Thus, we are able to discover a wide spectrum of long term inter-temporal dependencies in the data. *Pattern-by-pattern* does *not* include such a rescaling of the gradients. Consequently, as we have shown in our empirical study of the 3-month German inter-banking offered rate, it is not recommendable to use *pattern-by-pattern* in such applications.

Chapter 6

MULTI-AGENT MARKET MODELING: A GUIDE TO LITERATURE

In this chapter we give an overview of recent developments in the newly research area of multi-agent modeling. At a glance, multi-agent models can be seen as a new approach to build a more realistic theory of financial markets [Far99a, LeB99, LeB00, p. 9992, p. 8 and p. 679-80]. Instead of assuming market efficiency (i. e. market prices instantaneously incorporate *all* available information) [Fam70, Fam91] or rational expectations of the market participants [Mut61, Luc72] as it is done in standard equilibrium theory of financial markets, multi-agent models typically incorporate theories of human behavior from psychology or sociology [Far99a, Far98, p. 9992 and p. 5]. By this, e. g. 'market psychology' or 'investors sentiment' as some kind of irrational behavior is introduced into the market model, which is more likely to reflect the conditions of real-world financial markets [LAHP97, Hom01, p. p. 1-2 and p. 149-50]. In a multi-agent approach, the market is typically modeled from the bottom up, i. e. market price are explained by the interaction of a large number of agents [LeB01a, LeB00, p. 254 and p. 679]. Compared to the descriptive character of traditional econometric modeling (e. g. regression analysis or GARCH models) multi-agent models can be seen as an explanatory approach to financial markets: "Prices and returns originate from the buying and selling decisions of economic agents and, thus have identifiable behavioral roots which should be accessible to economic reasoning and analysis" [LCM01, p. 328].

The organization of this chapter is as follows: First, we introduce multi-agent models as an alternative way of explaining market prices. More precisely, we point out that there are evidences that the traditional economic theory of efficient markets [Fam70, Fam91] and its related assumptions are not fulfilled in real-world financial markets [Mut61, Luc72]. This is underlined by several empirical findings which (from the standpoint of the efficient market hypothesis or standard equilibrium theory of financial markets) may be characterized as market anomalies [Shl00, Shi97]. In this connection, we argue that multi-agent models may help to gain a deeper understanding of real-world financial markets, because they "provide a simple theoretical framework in which to investigate the impact of psychological behavior on price dynamics" [Far98, p. 5]. For instance, the interaction of artificial agents may generate a market dynamics which shares important characteristics of real-world financial data like fat-tailed return distributions (see Fig. 6.2), volatility clustering (see Fig. 6.3), low autocorrelation in the time series of returns and persistence (long-term autocorrelation) in market volatility (see Fig. 6.4) as well as complexity (i. e. evidence of non-linearity without the presence of a low-dimensional chaotic attractor) [Lux98, LM99, p. 155-62 and p. 498].

Intending to model a financial market from the bottom up with a large number of interacting agents brings up several detailed questions concerning the design of the multi-agent model. Central design issues are (*i.*) the microeconomic structure of the agents and (*ii.*) the macroeconomic structure of the market. An overview of these design issues is given in Fig. 6.1 [LeB01a, WHD02, BMT96, p. 255-9, p. 88-93 and p. 175].

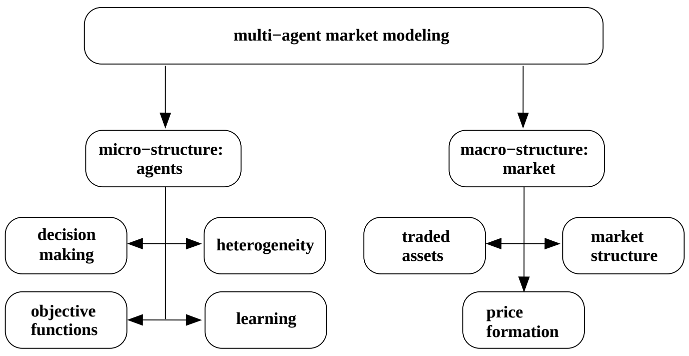


Figure 6.1 Design issues of multi-agent models [LeB01a, WHD02, BMT96, p. 255-9, p. 88-93 and p. 175]

On the microeconomic level (Fig. 6.1), one has to consider the modeling of the agents. Related design issues involve e. g. the agents' decision making, the objective function, heterogeneous decision behavior or learning [WHD02, LeB01a, p. 90-2 and p. 255-6]. Macroeconomic design issues address the general setup of the market and the interactions among the traders. In particular, macroeconomic design issues deal with the traded assets, the structure of the market and the organization of the trading as well as the price formation mechanism [WHD02, LeB01a, p. 88-90, 92-3 and p. 256-59]. Sections 2 and 3 try to answer the micro- and macroeconomic design questions by giving an outline of what is known about these topics in recent multi-agent literature.[1] In this connection, we also compare our multi-agent approaches to the surveyed models and show how the key design issues are addressed in our models.

1. MULTI-AGENT MODELS AS A NEW THEORY OF FINANCIAL MARKETS

In this section, we will point out that multi-agent models can be seen as an alternative view on financial markets that supplements the theory of informationally efficient markets [Far99a, LAHP97, p. 9991-2 and p. 1-2]. First, we explain the efficient market hypothesis [Fam70, Fam91] and discuss its underlying assumptions [Fam70, p. 383]. Thereafter, we point to several market anomalies (e. g. overreacting investors, price bubbles or profitability of technical trading) that are not in line with informationally efficient markets. As we will show, these market anomalies can be explained by agent-based models of financial markets [LeB99, LeB01a, p. 8 and p. 259-60].

The Efficient Market Hypothesis (EMH) and underlying Assumptions

Standard economic theory of financial markets is strongly related to the concept of market efficiency. The term market efficiency refers to "the speed with which information is impounded into security prices" [EG95, p. 407] and "under what conditions an investor can earn returns on this security" [EG95, p. 407]. Statements concerning market efficiency are typically summarized by the so-called efficient market hypothesis (EMH) [Fam70, Fam91, p. 383 and p. 1576-8]. The EMH is divided into *three* sub-categories specifying the manner in which markets process information [EG95, p. 406-7]:

1. *Tests of return predictability (weak form of the EMH):* The weak form of the EMH implies that current market prices fully reflect all information about historical price patterns [EG95, p. 4063]. Consequently, market prices cannot be predicted by the analysis of past returns and (technical) trading strategies which are based upon historical asset prices do not allow to gain excess returns [SvR96, p. 23]. Since only all information about historical price patterns is incorporated into actual market prices, it is possible to gain excess returns by evaluating (*i.*) other publicly available

[1]For surveys of recent developments in multi-agent literature, the reader is also referred to LeBaron (2001) [LeB01a], Wan et al. (2002) [WHD02] and Levy et al. (2000) [LLS00].

information like fundamental data or (*ii.*) private information (so-called insider knowledge).

2. *Event studies (semi-strong form of the EMH):* This sub-category of the EMH assumes that market prices fully reflect *all* publicly available information [EG95, SvR96, p. 406 and p. 23]. Public information includes all market data and all other publicly available information about the specific asset or the economy. If the semi-strong form of the EMH is valid, not only information about historical price patterns but also other publicly available information like fundamental data are contained in market prices. Only trading strategies which rely on private information are able to gain excess profits.

3. *Tests for private information (strong form of the EMH):* The strong form of the EMH states that actual market prices fully reflect *all* types of information, i. e. public as well as private information are completely incorporated into current market prices [EG95, p. 406-7]. Thus, it is *not* possible to gain excess returns even if private information is used [SvR96, p. 23].

The sub-categories of the EMH are originally proposed by E. Fama (1970) [Fam70, p. 383] and slightly modified in Fama (1991) [Fam91, p. 1576-8]. Only if *all* sub-categories of the EMH are *not* valid, it is possible to gain excess returns from the analysis of historical price data [EG95, p. 408-9].

Let us turn to the major assumptions and implications of the EMH.[2] The EMH implies that the actual market price *immediately* changes when new information are available [SvR96, p. 22]. Being so, asset returns are unpredictable and there is no opportunity to make systematic profits [Far99a, EG95, p. 9991 and p. 408-9]. The main assumptions underlying the EMH are those of perfect capital markets [SvR96, p. 22]. Among others, the EMH presumes e. g. (*i.*) complete information, (*ii.*) perfect rationality and (*iii.*) common expectations [PAHL94, Hom01, SvR96, p. 264-6, p. 149-50 and p. 22].

Complete information means that *all* publicly available information is accessible to all market participants simultaneously and exempt from charges [PAHL94, p. 264-5]. The agents have fully knowledge of all relevant information. Perfect rationality implies that no matter how complex the decision situation is, agents always deduce the optimal decision

[2]In the following the term EMH refers to the strong form of the EMH.

[PAHL94, p. 264-5]. Common expectations means that the agents presume that others derive their decisions on a perfect rational basis using the same information and knowledge of the market [PAHL94, p. 264-5]. From this it follows, that the agents evaluate the price impact of newly-available information in the same manner. In other words, the agents have common expectations about the actual and forthcoming market prices [PAHL94, p. 264-5].

Presuming rational agents, market efficiency can be derived by an arbitrage argument [Hom01, p. 149-50]: Market inefficiencies can be exploited by the rational agents through arbitrage processes in order to gain risk-less profits. The EMH therefore implies that "asset prices and returns [can be viewed] as the outcome of a competitive market consisting of rational traders, who are trying to maximize their expected returns" [Hom01, p. 150]. The value of any risky asset is completely determined by the fundamental price, i. e. the price is equal to the discounted stream of all future earnings [Hom01, p. 149-50]. From this it follows, that if the EMH is valid, market price changes are completely drawn at random for unexpected news about changing economic fundamentals [Far99a, p. 9991].

Critical View on the EMH and Evidences of Market Anomalies

In real-world financial markets the underlying assumptions of the EMH, e. g. perfect rationality, common expectations or complete information, are typically not fulfilled, because "rationality is difficult to define, human behavior is often unpredictable, information can be difficult to interpret, technology and institutions change constantly, and there are significant 'frictional' costs to gathering and processing information, and transacting" [Far99a, p. 9992]. For a critical evaluation of the assumptions underlying the EMH see also Palmer et al. (1994) [PAHL94, p. 265-6]. From this it follows, that the EMH is somehow questionable [Far99a, PAHL98, Shi97, p. 9992, p. 27, p. 1305-7].

Various studies – especially in the new research area of behavioral finance[3] – have shown market phenomena (or anomalies) that are not in line with the EMH. For instance, Grossman and Stiglitz "showed that when the efficient market hypothesis is true and information is costly, competitive markets break down" [GroSt80, p. 404]. This is referred to as the information paradox [GroSt80].

[3]Behavioral finance tries to explain observed anomalies of real-world financial markets by behavioral principles which come from the fields of psychology or sociology [Shl00, p. 1305].

DeBondt and Thaler (1985) [DeBT95] challenge the EMH by detecting long-term anomalies in stock returns. A performance ranking based on historical returns indicated, that stocks showing a lower-than-average performance in the past have a superior average performance over the next few years. In contrast, stocks with a higher-than-average performance tend to come off badly in the future. The authors claim that the cause of the observed long-term reversals in stock returns is an over-reaction of the investors. According to this theory, investors over-react to news about the companies. This behavior causes arbitrary over- and undervaluations of the stocks. In subsequent time periods, the investors notice the misvaluation and the stock prices revert to a correct level. This process can be utilized by other investors as basis for a profitable trading strategy.

In a related study, Cutler et al. (1989) [CPS89] found that investors underreact to financial news. This means, that good financial news, such as higher than expected earnings, are not completely incorporated into the stock price. Corrections of the underreaction are performed in subsequent time periods. Similar to over-reactions, investors can also profit from market anomalies which are caused by underreactions [CPS89].

Further evidences of market anomalies (i. e. signals of inefficient markets) and related studies in behavioral finance are summarized in Shleifer (2000) [Shl00] and Shiller (1997) [Shi97].

It is important to note, that neither the outlined market anomalies nor the violation of the underlying assumptions lead automatically to the rejection of the EMH [SvR96, Far99a, p. 23-4 and p. 9992]. There are lots of empirical studies concerning the validity of the efficient market hypothesis. An overview of such analysis is given in e. g. Fama (1991) [Fam91] or Elton and Gruber (1995) [EG95, p. 411-39]. The findings of these studies are controversial, because the proof of the validity of one of the sub-categories of the efficient market hypothesis is exceedingly difficult (for details see Fama (1991) [Fam91, p. 1575-6]). Test results for a particular market and time horizon can therefore only be taken as an *evidence* of the validity or invalidity of the EMH [SvR96, p. 24]. In the face of these difficulties, there are only specific empirical evidences, that the weak form of the EMH and specific forms of the semi-strong EMH (e. g. concerning dividend announcements) are valid, while the strong form of the EMH is likely to be rejected [SvR96, p. 24].

Multi-Agent Models as an Alternative View on Financial Markets

However, the empirical findings suggest, that there is a need for *other* explanatory models of financial markets *supplementing* the traditional economic concepts of equilibrium theory and efficient markets [BMT96, Far99a, LAHP97, LeB99, p. 269-70, p. 9991-2, p. 1-2 and p. 8].

Now, the question arises how we can approach real-world financial markets in a more realistic manner. In fact, there is growing interest in building a supplementary theory of financial markets which provides explanations of the observed market anomalies on a more realistic basis [Far99a, p. 9992]. Besides recent developments in behavioral finance (see e. g. Shleifer (2000) [Shl00] or Shiller (1997) [Shi97]) or in the field of evolutionary game theory [Fri91], multi-agent modeling of financial markets is probably one of the most important trends in this research area [Far98, Far99a, p. 5 and p. 9992].

While models in behavioral finance typically "focus on the manner in which human psychology influences the economic decision making process as an explanation of apparent departures from rationality" [Far99a, p. 9992], evolutionary game theory "studies the evolution and steady-state equilibria of populations of competing strategies in highly idealized settings" [Far99a, p. 9992]. Even more important than these aspects, multi-agent models give us a rise to the dynamics of real-world financial markets from the perspective of interacting individuals (agents) [LeB99, p. 8].

In an agent-based model the market is approached from the bottom up [LeB00, p. 679-80]. Typically, one starts off with modeling the behavior and decision making of the agents at the microeconomic market level. The resulting decisions (market orders) of the agents are collected at the macroeconomic side of the market. On this basis, a specific price formation mechanism determines the market clearing price. For instance, a so-called market impact function can be applied to compute a price change as a response to the observed market excess [Far98, p. 7-9]. Remarkably, multi-agent models incorporate agents who are heterogeneous in their decision behavior. In addition, the behavior of the agents is typically *not* completely rational [LeB01a, p. 255-6]. This is the view that traders and practitioners have of financial markets: In contrast to the theory of efficient markets, these people "believe that technical trading is profitable, and that 'market psychology' and herd effects unrelated to market news can cause bubbles and crashes" [Art95, p. 21].

We may conclude, that multi-agent models are able "to capture complex learning behavior and dynamics in financial markets using more realistic markets, strategies, and information structures" [Far99a, p. 9992]. Thus, they give an explanation of many facets of financial markets which are not within the scope of the efficient market theory [Far99a, p. 9992]. Or as LeBaron (1999) put it [LeB99, p. 8]:

> "Agent approaches to financial markets offer a radical new way to think about financial markets [...]. [... At the moment,] financial theory is at a state where it is asking for new theories to help explain the large swings and capital flows in modern markets. Traditional theories have a difficult time with this behavior [...]. In the future, agent based methods may be used to help in the design of actual trading mechanisms leading to more stable global foundations for goods and financial flows."

In the following, we point to some examples of agent-based financial markets, that help to explain complex market phenomena like price bubbles and crashes. Furthermore, we present multi-agent models that focus on the so-called stylized facts of financial markets (e. g. volatility clustering or fat-tailed return distributions [Lux98, LM99, p. 155-62 and p. 498]) or provide explanations for other market anomalies (e. g. profitability of technical trading [JPB00, p. 472-78]). These examples underline, that multi-agent models supplement standard economic theory and provide deeper insights in the dynamics of financial markets.

Agent-based Explanations for Price Bubbles and Market Crashes

Multi-agent based studies of price bubbles and market crashes are provided in e. g. Steiglitz et al. (1997) [Ste97], Levy et al. (1994) [LLS94], Kaizoji (2000) [Kai00] or Bornholdt (2001) [Bor01].

The model of Steiglitz et al. (1997) [Ste97] describes an artificial economy with two commodities (gold and food). The economy is populated with three types of agents: (*i.*) regular agents, who have zero intelligence and just produce and consume, (*ii.*) fundamentalists, who act upon their forecasts of fundamental values, and (*iii.*) chartists, who rely on estimated market trends [Ste97, p. 2]. Changing the fundamental value exogenously, the authors show that the interaction of the agents causes inflationary and deflationary price bubbles [Ste97, p. 2-3].

Levy et al. (1994) [LLS94] model heterogeneous agents, who differ in their memory spans and are influenced by unknown individual psychological factors. The individual psychological factors are introduced by a random variable ϵ with standard deviation σ_ϵ [LLS94, p. 105]. The agents try to maximize their utility by considering two investment opportunities, a risky stock paying a dividend D and a risk-less bond. The authors found, that if the investors are homogeneous, i. e. ϵ is small, market crashes are likely to appear [LLS94, p. 106-9]. Furthermore, the dividend to price ratio is suitable indicator of the market behavior, i. e. "when the dividend yield is relatively low, it is a sign of a bear market and a crash is to be expected" [LLS94, p. 109]. If the agents are more heterogeneous in their decision making, market cycles become milder, market crashes are much smaller and also the probability of a market crash is much lower [LLS94, p. 109-10]. Remarkably, the interaction of the agents fits real-world behavior and supports common views in financial media [LLS94, p. 109-10].

In the approach of Kaizoji (2000) [Kai00], stock market bubbles and crashes are due to the speculative activity of interacting agents. The multi-agent approach is motivated by the Ising model which is well-known in statistical mechanics [Kai00, p. 497-8]. In the model, single agents are influenced by the market expectations of other individuals [Kai00, p. 495-7]. Market crashes and bubbles are described and explained on the basis of phase transitions [Kai00, p. 498-501]: In case the interaction among the agents reaches some critical value, "a second-order phase transition and critical behavior can be observed, and a bull market phase and a bear market phase appear" [Kai00, p. 493]. If the bull market phase persists, one can observe speculative bubbles. The phase transition from a bull to a bear market can be seen as a stock market crash. Interestingly, the model parameters are estimated from real-world financial data of the Japanese stock market. Analyzing the price bubble and the market crash of the Japanese crisis (1987 – 1992), the good fitting of the model indicates, that the Japanese crisis may have its "origin in the collective crowd behavior of many interacting agents" [Kai00, p. 493].

Bornholdt (2001) [Bor01] also uses a simple Ising spin model to analyze the expectation bubbles and crashes which emerge from the dynamics of interacting agents. The investment attitude of a single agent is represented by a single spin with orientation -1 or $+1$. In case the agent is a buyer (seller), the orientation of the spin is $+1$ (-1) [Bor01, p. 669]. Now, the market dynamics can be understood in terms of ferromagnetic couplings which connect each spin (agent) to its local neighbors. An additional coupling introduces a relationship of each spin (agent) to the global magnetization (superposition of agents' decisions). This coupling may be anti-ferromagnetic. This means, that a minority of agents is in opposition to the majority opinion of the market [Bor01, p. 670-1]. As a result of the coupling conditions, "a metastable dynamics with intermittency and phases of chaotic dynamics" is created [Bor01, p. 667]. The multi-agent model reproduces well-known stylized facts of financial markets, namely power-law distributed returns and volatility clustering [Bor01, p. 672-4].

Agent-based Explanations of Stylized Facts

Another remarkable property of agent-based models is that the artificial time series share important characteristics of real-world financial data [LeB01a, p. 259-60]. These so-called stylized facts of financial markets result (endogenously) from the interaction of heterogeneous agents within the multi-agent framework [Lux98, LCM01, p. 146-7 and p. 328]. Examples of stylized facts are

- unit root properties of market prices (non-stationarity [PDP00, p. 93-7 and p. 352]),
- fat-tailed return distributions (leptokurtotic return distributions [PDP00, p. 142-4], see Fig. 6.2),
- volatility clustering (quiet and turbulent periods of volatility tend to cluster together [PDP00, p. 129 and p. 310], see Fig. 6.3),
- low autocorrelation in time series of returns together with long-term high autocorrelation in series of return volatility (linear dependencies within the analyzed time series [PDP00, p. 97-102], see Fig. 6.4), and
- complexity (evidence of non-linearity without the presence of a low-dimensional chaotic attractor [Lux98, p. 155-62]).

Multi-agent models that exhibit stylized facts of financial markets are e. g. Lux (1998) [Lux98], Lux and Marchesi (1999) [LM99], Farmer (1998) [Far98], Bornholdt (2001) [Bor01], Caldarelli et al. (1997) [CMZ97] and Youssefmir et al. (1997) [YH95].

The financial market model of Lux (1998) [Lux98][4] incorporates two groups of traders: 'noise traders' and 'fundamentalists'. The decision making of the noise traders is based on historical price patterns *and* the majority opinion of the market (behavior of other agents). The latter leads to so-called herding or bandwagon effects [Lux98, p. 148-51]. The population of noise traders is divided into optimists or pessimists. Optimists believe in upwarding market trends, while pessimists anticipate declining market prices [Lux98, p. 148]. Fundamentalists expect that the market price has an inherent tendency to follow the fundamental value of the asset. If the actual market price is above (below) the estimated fundamental value, a fundamentalist sells (buys) the asset [LM99, p. 498]. As a remarkable property, the agents are allowed to switch between the different groups, i. e. a pessimistic noise trader may become a

[4]For an extended version, see Lux and Marchesi (1999) [LM99]

fundamentalist [Lux98, p. 148-51]. Each time period, the agents submit their buying and selling orders to the market. Adjustments of the market price are performed by a market maker in order to absorb imbalances of demand and supply [Lux98, p. 151].

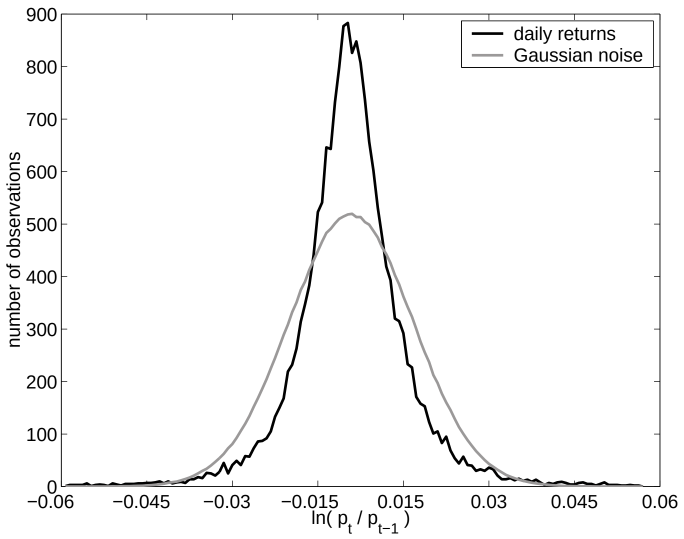


Figure 6.2 Distribution of the daily returns of the Dow Jones stock market index (Jan. 1931 – Mar. 2002). Compared to the distribution of a Gaussian white noise process, the return distribution has fat-tails [PDP00, p. 142-4].

The agent-based model of Lux (1998) [Lux98] generates all of the mentioned stylized facts of financial markets endogenously through the interaction of the agents. For instance, Lux and Marchesi (1999) argue that the source of volatility clustering and leptokurtotic return distributions is the switching of agents between the population of chartists and fundamentalists [LM99, p. 499-500]. The key feature for the stability of the market dynamics is the development of the noise trader population. Market turbulences emerge, when the portion of noise traders is close to a critical value [LM99, p. 500]. However, the market dynamics has a globally stable equilibrium in which market prices track fundamental values [Lux98, p. 154-5].

Similar to the design of Lux (1998) [Lux98], Farmer (1998) [Far98] investigates a financial market in which the price dynamics is caused by the interaction of 'value investors' and 'trend followers'. Value investors compare the current market price to the fundamental value of the asset, while trend followers rely on technical trading rules. The behavior of each agent is described by a single trading rule [Far98, p. 16-23 and p.

26-9]. A market maker collects the orders of the agents and balances demand and supply by adjusting the market price level [Far98, p. 7-9]. The interaction of value investing and trend following agents "gives rise to commonly observed market phenomena such as fat tails in the distribution of log-returns, correlated volume and volatility and temporal oscillations in the difference between prices and values" [Far98, p. 15].

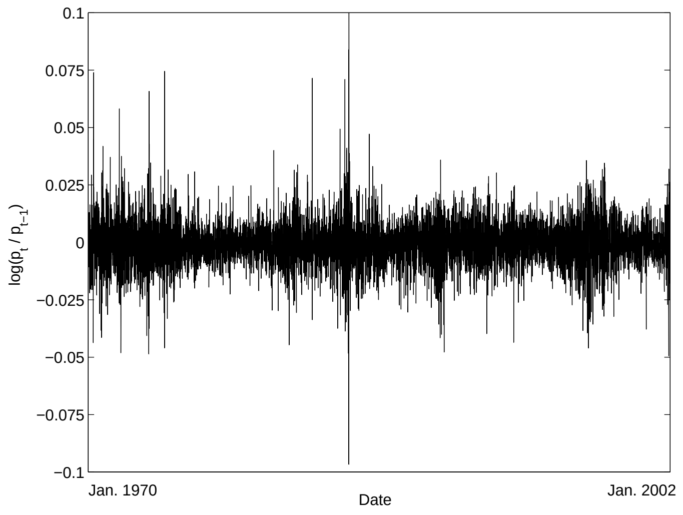


Figure 6.3 Volatility clustering in the time series of daily returns of the Dow Jones stock market index (Jan. 1970 – Jan. 2002). As shown in the plot of the daily returns, quiet and turbulent periods of volatility tend to cluster together [Lux98, p. 160].

Furthermore, Caldarelli et al. (1997) [CMZ97] study the interaction among agents in an artificial stock market which is *not* influenced by external factors. The agents rely on trading rules which are solely based on historical price patterns. Price adjustments are performed as a response of the observed market excess. As a major outcome of the model, the generated market price time series shows a very rich and complex statistics which is comparable to the behavior of real-world financial data. For instance, the returns have scaling properties similar to real-world stock or foreign exchange markets [CMZ97, p. 482-3]. The tails of the return distribution have a power law character. From time to time, market crashes emerge without any advance notice. They are generated by the collective trading activity of the agents. According to the authors, the results suggest "that the statistics we observe in real markets is mainly due to the interaction among speculators trading on technical grounds, regardless of economic fundamentals" [CMZ97, p. 484].

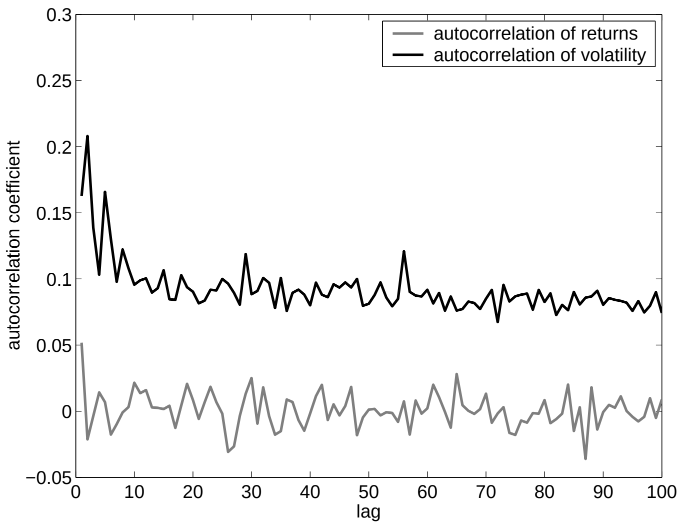


Figure 6.4 Autocorrelation coefficients of raw and squared daily returns of the Dow Jones stock market index (Jan. 1931 – Mar. 2002). The autocorrelation coefficients of both series are computed up to a lag of $\tau = 100$ [PDP00, p. 97-98]. Both autocorrelation functions exhibit a typical behavior [Far98, p. 30-2]: While there is low autocorrelation in the time series of returns, the time series of volatility (squared returns) shows persistence, i. e. high levels of autocorrelation over long-term time scales.

The model of Youssefmir and Huberman (1997) [YH95] incorporates agents who continuously change their behavior in order to gain access to limited resources in an unpredictable environment. The behavior of the agents is driven by utility maximization. The agents "decide on the basis of having bounded rationality, which implies imperfect information and delayed knowledge about the present state of the system" [YH95, p. 20]. The authors found that there are fluctuations around a stable market equilibrium which result from changes in the behavior of the agents. Every now and then, one can observe sudden bursts of activity in the market, i. e. volatility clustering (see Fig. 6.3) [YH95, p. 10-13]. The authors conclude, that the behavior and interaction of the agents "may explain the clustered volatility observed in some economic time series, such as stock returns and exchange rates" [YH95, p. 21].

Agent-based Explanations of other Market Anomalies

Finally, let us point to some other market phenomena that seem to contradict the strong form of the efficient market hypothesis. Market anomalies like the profitability of technical trading or investment strategies that produce excess returns are difficult to explain by traditional economic theories [LeB01a, p. 259-60]. As we will show, multi-agent models try to explain the observed market anomalies from the viewpoint of interacting agents [WHD02, p. 88]. Examples of such agent-based models are De la Maza and Yuret (1995) [MY95], Bak, Paczuski, and Shubik (1996) [BPS96], Joshi, Parker and Bedau (2000) [JPB00] and Chen et al. (2001) [ChYe01].

De la Maza and Yuret (1995) [MY95] present a stock market simulation on the basis of genetic algorithms. The agents refer to trading strategies, which are modified and improved on the basis of genetic algorithms [MY95, p. 292-3]. Each agent computes a fair price for the stock on the basis of historical price patterns and submits his price expectation to the market [MY95, p. 292]. The market equilibrium price is the median of the fair prices. Agents who estimated a fair price above (below) the equilibrium price buy (sell) one share at the equilibrium price [MY95, p. 292]. In their experiments the authors discovered that "under certain conditions, some market participants can make consistent profits over an extended period of time" [MY95, p. 290]. This result "might explain the success of some real-world money managers" [MY95, p. 290].

Bak, Paczuski, and Shubik (1996) [BPS96] construct an artificial stock market, which incorporates agents with imitating behavior. There are two types of agents, 'noise traders' and 'rational agents' [BPS96, p. 3 and p. 7-10]. Noise traders consider historical price patterns and may also imitate the behavior of other agents. Rational traders optimize their utility functions on the basis of a fundamental analysis of the traded stock. The interaction among the agents causes market price series that have the same statistical properties (e. g. volatility clustering or fat-tailed return distributions) as empirical observations [BPS96, p. 15-28]. Another major result is, that "when the relative number of rational traders is small, 'bubbles' often occur, where the market price moves outside the range justified by fundamental market analysis" [BPS96, p. 1]. In other words, the imitative behavior of the agents can be seen as an explanation for temporary deviations between market prices and the underlying fundamental value.

An explanation for the profitability of technical trading rules is provided by the agent-based stock market model of Joshi, Parker and Bedau (2000) [JPB00]. Based on the Santa Fe artificial stock market [PAHL94] the authors show that "widespread technical trading can arise due to a multi-person prisoners' dilemma in which the inclusion of technical trading rules to a single agent's repertoire of rules is a dominant strategy" [JPB00, p. 465].

Chen et al. (2001) [ChYe01] study an artificial stock market resting on an evolving agent population. The underlying learning mechanism of the agents is a so-called business school, which is based on genetic programming. As a major result of the multi-agent model, the authors find that the artificial price series follows a random walk process. Insofar, the strong form of the efficient market hypothesis is supported [Far99a, p. 9991]. Interestingly, the random price fluctuations are generated by the interaction of agents who do *not* believe in market efficiency. However, the experimental results of the authors also indicate, that some agents are able to outperform the market on a short-term time scale. This finding can be seen as an evidence of short-term market anomalies.

As it can be seen from these examples, agent-based models provide an experimental framework for studying various market anomalies and stylized facts of real-world financial markets. Multi-agent models provide new explanations of observed market anomalies, which are *not* covered by the theory of efficient markets or the related theory of rational expectations [WHD02, Far99a, p. 124 and p. 9992]. From this point of view, agent-based markets supplement standard economic theory and provide deeper insights in the dynamics of real-world financial markets [Far99a]. Or as LeBaron (2001) [LeB01a, p. 259] put it:

> "There appears to be a growing set of features that traditional financial models find difficult to generate, but agent-based models readily provide. Among these are fat tailed return distributions [(see Fig. 6.2)], persistent volatility [(see Fig. 6.4)], and widely fluctuating trading volume."

Up to now, multi-agent models only provide *qualitative* results, i. e. the interaction of the agents gives 'only' an explanation of complex economic phenomena. Remarkably, *quantitative* results, i. e. the prediction of real-world market prices, are *not* supplied by the mentioned multi-agent models [Far98, LeB01a, p. 31 and p. 259-60]. This lack of multi-agent modeling is addressed in chapters 7 and 8. Here, we introduce multi-agent models which enable us to forecast price shifts of real-world financial markets.

2. MICROECONOMIC DESIGN ISSUES OF MULTI-AGENT MODELS: AGENTS

Probably the most important design issue of a multi-agent approach is the modeling of the agents. The spectrum of agent modeling varies from simple 'zero-intelligence' traders as in Gode and Sunder (1993) [GS93], who submit random buying or selling orders to the market, to highly sophisticated agents as in the Santa Fe artificial stock market [LAP99, LAHP97], who adapt to each other's behavior and explore new trading strategies using genetic algorithms. In the following, we approach the modeling of the agents by clarifying *four* design issues (Fig. 6.5).

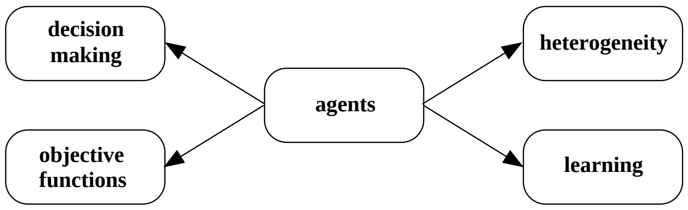


Figure 6.5 Major design issues of microeconomic agent modeling (see also Wan et al. (2002) [WHD02, p. 90-3] and LeBaron (2001) [LeB01a, p. 255-6])

The first design issue addresses the decision making processes of the agents (Fig. 6.5). In general, one may distinguish between 'rule-based' and 'forecasting' agents [BMT96, WHD02, p. 174-5 and p. 90-2]. The decision making of rule-based agents is guided by a specific hard-wired or dynamical trading strategy (rule). Forecasting agents deduce their decisions from a specific prediction model. As additional topics of the agents' decision making, we discuss intertemporal decision making schemes [LeB00, p. 697] and contagious decision making (imitative behavior) [Lux95, p. 883-9].

The second design issue deals with objective functions which guide the decision making of the agents [LeB01a, p. 255-6]. The underlying assumption is that the decision making of rational agents is always goal-oriented. This means, that the agents always perform their actions in accordance with a specific objective function, e. g. profit or utility maximization.

The third issue discusses the topic of heterogeneity [BMT96, p. 178], which is strongly related to the decision making of the agents. In a first shot, heterogeneous decision making is essential for the modeling of the market dynamics. If the agents behave too homogeneous, the market often tends to be highly volatile. On the other hand, too heterogeneous agents may cause a stationary market, which likewise does not reflect real-world behavior [WHD02, BMT96, p. 124 and p. 177].

The fourth design issue considers the way agents learn and evolve their trading strategies over time [WHD02, BMT96, p. 91 and p. 175]. For instance, one may start off with naive agents that rely on simple trading rules. During the market process, the agents learn and may develop advanced trading strategies [LeB01a, p. 255-6]. As we will point out, learning is not necessarily associated with evolutionary algorithms, since there are other ways to improve the decision making behavior of the agents.

2.1 DECISION MAKING

First of all, one has to make a basic assumption concerning the general structure of the agents: The decision making schemes of the agents may either have the same structure or may differ somewhat within the total population. We speak of homogeneous and heterogeneous decision making schemes [BMT96, p. 178].

The multi-agent model of Gode and Sunder (1993) [GS93, p. 121-3] is probably one of the most simple examples of *homogeneous* decision making processes. The decision making of the 'zero-intelligence' traders is completely random. Each agent draws his buying and selling decisions from a random distribution over a predefined range. Another example of agents with homogeneous decision making schemes can be found in Levy et al. (1994) [LLS94]. Here, all agents are endowed with the same structural type of utility function and rely on identical decision making schemes. This design is similar to Lettau (1997), who assumes that all agents have constant absolute risk aversion preferences [Let97]. A more sophisticated example of homogeneous decision making processes is given in the Santa Fe artificial stock market. Here, each agent is basically composed of a set of forecasting rules [PAHL94].

The most common way of modeling *heterogeneous* decision making schemes is to introduce specific types of agents within the population. Each type is characterized by a different decision making process or trading strategy. Examples of heterogeneous decision making schemes can be found in Bak et al. (1996) [BPS96], Chiarella et al. (2001) [ChHe01], Steiglitz (1997) [Ste97], Farmer (1998) [Far98], Youssefmir et al. (1997) [YH95] or Lux (1998) [Lux98].

Most of these papers introduce two types of agents: chartists and fundamentalists. The decision making of these agent types is different. While chartists only concern historical price patterns and related technical indicators, fundamentalists compare the actual market price to an expectation of the underlying fundamental value in order to come to a decision.

Remarkably, the size of the sub-populations can either be fixed or may change over time. Fixed sub-populations are used in Farmer (1998) [Far98, p. 15], whereas Kaizoji (2000) [Kai00, p. 495-7] or Lux (1998) [Lux98, p. 148-54] allow the agents to switch between the different trading strategies. Note, that different structural designs of the agents' decision making schemes are also an important source of heterogeneity (see sub-section 3).

Let us now turn to the concrete design of the agents' decision making schemes. The principal task of an economic agent is to process information and convert this into buying and selling decisions [FJ99, LeB01a, p. 2, 24 and p. 255-6]. A simple decision making process typically incorporates three stages: (*i.*) the processing of information, (*ii.*) the formulation of a superposition of information and (*iii.*) the generation of a trading decision on the latter basis [Zim89, Zim94, p. 496-7 and p. 3-4]. Obviously, there are many ways to process market information and to form a trading decision. Following the suggestions of LeBaron (2001) [LeB01a, p. 255-6] and Beltratti, Margarita and Terna (1996) [BMT96, p. 174-5], we propose to classify the decision making processes of the agents as depicted in Fig. 6.6. The classification is based on the structural representation of the agents' decision making processes.

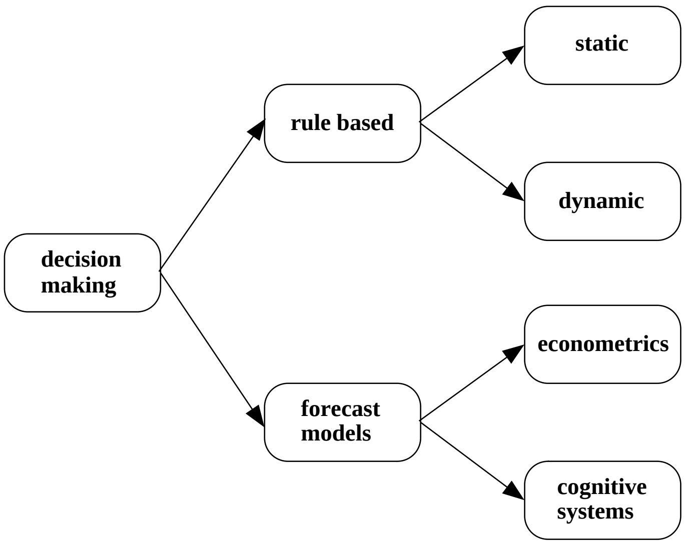


Figure 6.6 Classification of agents' decision making processes. [LeB01a, BMT96, p. 255-6 and p. 174-5].

As shown in Fig. 6.6, one can classify the decision making processes of the agents into two major groups. First, the decision making process can be represented a single trading rule or, alternatively, by a system of trading rules. Here, we speak of 'rule-based' agents. Second, the decision making of the agents may incorporate a forecast model. The forecasts are utilized during the decision making in order to derive a trading decision. We refer to these agents as 'forecasting' agents. In other words, so-called 'rule-based' agents "decide the actions directly from their information set" [BMT96, p. 174], while 'forecasting' agents "use their information set to forecast a variable which is then processed by exogenous rules" [BMT96, p. 174].

2.1.1 RULE-BASED AGENTS

In the simplest case, the behavior of a single agent is described by a single trading rule, which is often deduced from real-world trading strategies [LeB01a, p. 255]. The buying and selling decisions of a rule-based agent are directly derived from the trading rule. In other words, incoming information are directly fed into the trading rule in order to compute a trading decision [BMT96, p. 174]. Or as Farmer and Joshi (1999) [FJ99, p. 2] put it: "Insofar as individual traders use deterministic decision rules, they act as signal filters and transducers, converting random information shocks into temporal patterns in prices."

One approach to classify the trading rules is by analyzing the sort of input data [Far98, p. 15]. Typically, one distinguishes between 'technical' and 'fundamental' input information. Strategies that rely on historical price patterns are called technical trading rules. A common example of a technical trading rule is a trend following strategy, which tries to identify market trends [FJ99, p. 8]. Remarkably, the long and short positions of a trend following agent show a positive correlation with the most recent market price changes.

In contrast to technical rules, fundamental (or so-called 'value') strategies "are based on external information leading to a subjective assessment of the long term fundamental value" [FJ99, p. 8]. The estimation of the fundamental value is typically based on fundamental data like earnings or dividend prospects. The underlying assumption of a fundamental trading rule is that the market price has a specific tendency to revert to the fundamental value. Value investors compare the fundamental value to the actual market price. If the fundamental value is lower (higher) than the market price, value investors take a long (short) position. For simplicity, some multi-agent models (e. g. Farmer (1998) [Far98], Bak et al. (1996) [BPS96] or Lux (1998) [Lux98]) assume that the fundamental value is exogenously given. Changes in the fundamental value are introduced by a random noise term. An example of a more sophisticated value investing agent with entry and exit thresholds is depicted in Fig. 6.7 [FJ99, p. 16].

Technical rules can be seen as signal filters, because the information contained in historical price patterns is used to anticipate future prices. Fundamental rules can be seen as signal transducers, since external input data (other than historical price patterns) is transferred into future prices [FJ99, p. 8].

Within the class of rule-based agents (Fig. 6.6), one can distinguish between static and dynamic rule sets. Agents relying on static rules cannot adapt their trading strategies to changing market situations, i. e. the

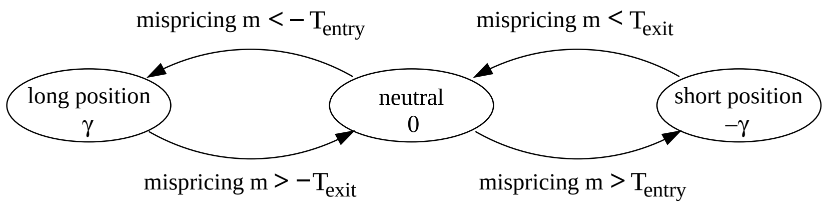


Figure 6.7 A value investor with entry and exit thresholds T_{entry} and T_{exit} enters a fixed long position γ if the deviation m between the actual market price p and the fundamental value v is smaller than the negative entry threshold $-T_{entry}$, i. e. $m = p - v < -T_{entry}$. In this situation, the asset is highly undervalued. The agent maintains the long position γ until the market price has nearly converged to the fundamental value, i. e. the mispricing becomes larger than the negative exit threshold $-T_{exit}$. Correspondingly, the agent takes a short position $-\gamma$ if the mispricing $m = p - v > 0$ exceeds the entry threshold T_{entry}. The short position $-\gamma$ is covered when the overvaluation of the stock vanishes, i. e. in case the mispricing drops below the exit threshold T_{exit}. If the mispricing is small, the agent is market neutral. Note, that the entry threshold T_{entry} is by definition larger than exit threshold T_{exit} ($T_{entry} > T_{exit}$).

rules are fixed over time [LeB01a, p. 255]. Even slight modifications of these 'hard-wired' trading rules are *not* possible.

On the other hand, agents incorporating dynamic rules are able to modify their rules in order adapt themselves to changing market conditions [LeB01a, p. 255]. In this case, the dynamic rule set is typically evolved by a form of evolutionary learning [LeB01a, p. 255-6]. Details on learning and adaptive behavior are outlined below (see subsection 2.4). Furthermore, we speak of dynamic rules, if the agents are able to switch between different 'hard-wired' trading strategies. This can be seen as an imitation process [Lux95, Lux98, p. 883-6 and p. 148-54]. For instance, strategy switches may depend on the relative attractiveness (pay-off differential measured by excess profits) of the different strategies [Lux98, p. 151-54].

An example of rule-based agents incorporating static rules is the model of Gode and Sunder (1993) [GS93, p. 121-3]. Agents who are endowed with a set of dynamic trading rules can be found in the Santa Fe artificial stock market [PAHL94, PAHL98]. Let us briefly point to these examples of rule-based agents.

Probably the simplest version of rule-based agents is given by Gode and Sunder (1993) [GS93]. The trading rule of their so-called 'zero-intelligence' traders is as follows [GS93, LeB00, WHD02, p. 121-3, p. 683-4 and p. 115-7]: Each time period, the agents submit bids or offers for an asset to the market. Bids as well as offers are completely drawn at random over a predefined range. Consequently, the agents neither evaluate any kind of information nor have an objective function which guides their decision making. The trading rules are 'hard-wired'. In a second series of experiments, the authors employed an additional budget constraint to their agents. Since each agent is restricted to the budget constraint, a buyer (seller) will not submit a bid (offer) to the market which is lower (higher) than a specific predefined redemption value of the asset.

The agents in the Santa Fe artificial stock market refer to a set of dynamic trading rules [PAHL98, p. 29-30]. Each trading rule fits to a specific market situation, i. e. the trading rules which are activated in a trend market are different from those designed for a steady market. The agents try to identify the actual state of the market by analyzing a set of fundamental and technical indicators. The market evaluation leads to the activation of a specific trading rule, which matches the identified market situation. The activated trading rule is directly used to derive the trading decision of the agent [PAHL98, p. 29-30]. Since the activation of a particular trading rule depends on the condition of the market, these agents are called 'condition action' agents [PAHL94, WHD02, p. 269-71 and p. 106-7]. In an extended version of the Santa Fe artificial stock market the rules are used to *predict* the future return and dividend of the traded asset. The resulting forecasts are evaluated by a risk aversion analysis. These agents are referred to as 'condition forecasting' agents [PAHL98, p. 29]. The trading rule sets are evolved on the basis of genetic algorithms. The performance of the rules is used as a fitness criterion. – Superior trading rules have a higher weighting (fitness) than unprofitable rules. Inefficient trading rules are replaced by new rules, which are explored by the genetic algorithm using mutation and crossover operators [PAHL98, p. 30]. Initially, the agents are equipped with little rationality and specialized knowledge about the market. During the market process, the agents learn and thus, become reasonable experts in their domains [PAHL94, p. 265-6].

A benefit of simple rule-based agents is that "this method can lead to very tractable precise results, which gives insights about the interactions between trading rules" [LeB01a, p. 255]. Probably the most important disadvantage of rule-based agent modeling is, that if an important trading strategy is left out, the interaction of the present strategies may not reflect the dynamics of real-world financial markets [LeB01a, p. 255]. Even if the agents are allowed to explore new rules or to adapt each others behavior, it is not ensured that missing strategies are discovered by the agents themselves. Another disadvantage of rule-based agents is that the decision making is typically not guided by an objective function [LeB01a, p. 255]. Hence, one cannot measure the performance of the agents in terms of their own subjective information processing. Furthermore, without an objective function, the agents may not consider tradeoffs between different investment alternatives or different investment goals. Such tradeoffs are typically incorporated into the agents' decision making by utility concepts or related objective functions [WHD02, p. 90].

2.1.2 FORECASTING AGENTS

Another way of modeling the agents' decision making is to incorporate a specific forecast model into the decision scheme [BMT96, p. 174]. The resulting decision scheme typically consists of two steps [BMT96, p. 174]: First, the agent predicts the future development of the market price. Afterwards, the forecast is processed by an additional decision rule or objective function. In this case, we speak of a 'forecasting' agent (see Fig. 6.6).

The major difference between rule-based and forecasting agents is that the former deduce their trading activities *directly* from the input information, while the latter use the input signals to construct a forecast model. The resulting forecast is then evaluated by an additional decision rule or objective function. In other words, forecasting agents deduce their trading decisions *indirectly* from the input data by the construction of a forecast, which is afterwards evaluated by an additional decision rule [BMT96, p. 174].

Within the category of forecasting agents, one may distinguish between agents referring to econometric forecast techniques and agents who are modeled by cognitive system based approaches (see Fig. 6.6). As a major difference between these sub-categories, cognitive system based agents are modeled by semantic specifications instead of being limited to the assumption of ad-hoc functional relationships [EM01, p. 759-61]. In other words, the cognitive process of an agent is either explicitly modeled or implicitly represented by an econometric forecast model and a related decision rule.

First let us turn to agents who rely on econometric forecast techniques. To model the forecasts of these agents, various techniques from the field of econometric time series processing can be applied [LeB01a, p. 256]. For instance, one may refer to simple or multiple regression analysis, nonlinear regression, neural networks or (G)ARCH processes. Comprehensive summaries of econometric forecast methods are given in e. g. Poddig et al. (2000) [PDP00] or Poddig (1999) [Pod99]. Examples of forecasting agents can be found in Beltratti et al. (1996) [BMT96], Yang (1999) [Yan99] and LeBaron (2001b) [LeB01b].

In the model of Beltratti et al. (1996) [BMT96], the agents use feedforward neural networks to predict upcoming stock price movements.[5] Basically, the authors introduce two kinds of market participants [BMT96, p. 224-5]: smart and naive agents. Smart agents predict stock price movements on the basis of 3-layer feedforward neural networks with four input signals. Inputs to the 3-layer networks are the most recent market prices π_{t-1}, π_{t-2} and the former transaction prices $P_{ij,t-1}$, $P_{ij,t-2}$. In contrast, the forecast models of naive agents are simplified: Naive agents only rely on the most recent market price π_{t-1} in order to forecast the future development of the stock price. The underlying neural network architecture consists only of a single input neuron containing the most recent market price and one output neuron computing the price forecast. The feedforward neural network architectures of smart and naive agents are depicted in Fig. 6.8 [BMT96, p. 224-5].

A multi-agent model based on recurrent neural networks is presented in Yang (1999) [Yan99]. The model of Yang (1999) includes three types of agents, value traders, momentum traders, and noise traders [Yan99, p. 10-3]. The agents place there funds either in a risky stock paying a stochastic dividend or in a riskless bond. Value investors believe, that the actual stock price reflects the discounted stream of all future dividends. Momentum traders and noise traders are technicians who only consider historical price patterns [Yan99, p. 12-3]. While momentum and noise traders are modeled as rule-based agents who refer to technical trading rules, value traders form their expectations on the basis of Elman's recurrent neural networks [Yan99, p. 10-2]. These recurrent neural networks incorporate so-called context units, which feed the information of previous activation values back into to the network (see chp. 2). More precisely, value investors use recurrent neural networks to predict the dividend growth of the risky asset. Afterwards, the market price of the risky asset is estimated on the basis of the dividend growth by using Gordon's constant dividend growth model [EG95, Yan99, p. 452 and p. 12].

Another neural network based approach can be found in LeBaron (2001b) [LeB01b]. In this artificial stock market, the agents invest their funds either in a riskless bond or in a risky asset paying a stochastic dividend. As a specialty, the agents have different decision making horizons: Some agents are long-term investors, while others rely on short-term planning horizons. According to their planning horizon, the agents

[5]In their book, Beltratti et al. (1996) [BMT96] discuss one-agent, one-population and multi-population models. Here, we refer to the basic multi-population stock market model [BMT96, p. 223-35].

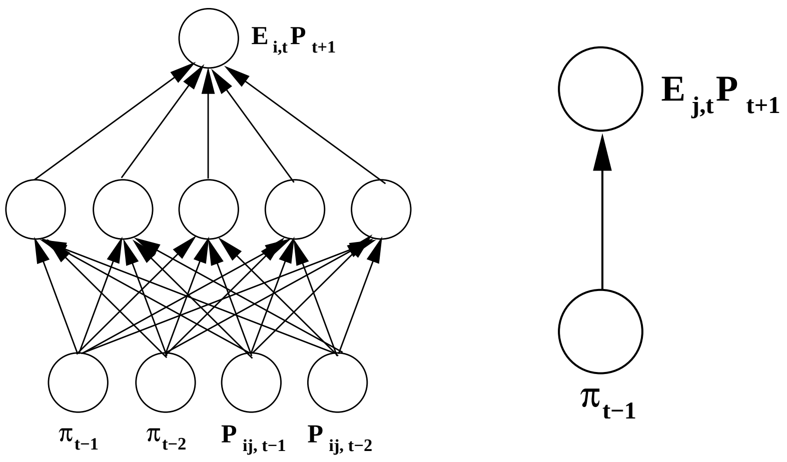


Figure 6.8 Neural network of smart agents, left, and network architecture used by naive agents, right [BMT96, p. 224-5]. Each smart agent i uses a 3-layer feedforward neural network to generate his expectation of the future market price $E_{i,t}P_{t+1}$. The network consists of four input, five hidden and one output neuron. Inputs to the network are the two most recent market prices π_{t-1}, π_{t-2} and the two previous transaction prices $P_{ij,t-1}$, $P_{ij,t-2}$ of the agent. The hidden neurons are equipped with logistic activation functions (see chp. 2, Eq. 2.5). The neural network of a naive agent j consists only of one input neuron containing the most recent market price π_{t-1} and an output neuron, which computes his price expectation $E_{j,t}P_{t+1}$. The output neuron is endowed with a logistic activation function. Both networks are trained by standard error backpropagation (see chp. 5).

choose from a broad spectrum of forecast models that are fitted to historical data. Forecast models and agents are therefore separated. The set of forecast models can be seen as a collective pool that is shared across all agents. Each forecast model is composed of a simple feedforward neural network incorporating one hidden neuron and a limited number of input signals. The input signals consist of technical and fundamental indicators. The neural networks are evolved by using a genetic algorithm.

As it can be seen from these examples, the decision making schemes of econometric agents do not incorporate semantic specifications of the underlying cognitive processes in terms of e. g. perception, internal processing and action. In other words, the econometric forecast models of the agents merely assume ad-hoc functional relationships between the input signals and the target values. Or as Edmonds and Moss (2001) [EM01, p. 760-1] put it:

"It is still common in multi-agent modeling to take an existing algorithm (designed with another purpose in mind) and use it for the cognition of the agent, regardless of whether this can be justified in terms of what is known about the behavior of the modeled entities. [... Simply] 'plugging in' existing techniques will not help in *understanding* what we model. [...] Rather, if we construct models whose structure *including that of cognition* corresponds to a description of the [agent's behavior and cognition ...]. Thus the algorithm that is to drive an agent's behavior must be, at least, *chosen* and *adapted* to correspond to what is known concerning the modelled entity's behaviour and cognition."

Cognitive system based agent models are an approach of capturing the semantic specifications of the agents' decision schemes in a structural framework [ZNG01b, p. 767-770]. More precisely, the decision making of an agent is modeled by a basic cognitive system. The cognitive system incorporates three properties [Cra43, RSMH86, RN95, GS83, p. 57, p. 40-4 and p. 13-4]: perception, internal processing and action. These properties constitute necessary conditions for a cognitive system and may include various other features [Cra43, RSMH86, p. 57 and p. 40-4]. As a structural representation of such a cognitive system, one may refer to time-delay recurrent neural networks [ZNG01b, p. 769-770]. The cognitive process generates not only expectations of the market price, but also concrete trading decisions. Since a cognitive agent has a specific objective function (e. g. utility maximization), the resulting actions are always goal-oriented [RN95, ZNG01b, p. 31-3, p. 41-5, p. 471 and p. 769]. As it can be seen from this outline, the buying and selling decisions of a cognitive agent are *not* deduced from a specific econometric forecast model which only presumes an ad-hoc functional relationship between external influences and the market development. Rather, the decisions of the agent are formulated in terms of the underlying cognitive system [EM01, p. 759-1]. This is truly a more realistic approach of modeling the agents' behavior. More details on cognitive agents can be found in chapter 8.

2.1.3 INTERTEMPORAL DECISION MAKING SCHEMES

The decision making horizon is a design issue which is related to rule-based and forecasting agents. Typically, the agents have a myopic view, i. e. their planning, objective function and decision making involves only a single time period. This means, that the agents do *not* consider intertemporal plans or multi-period preferences [LeB01a, LeB00, p. 256 and p. 697]. An additional point is that the majority of multi-agent models incorporates only agents who have the same (short-term) planning, forecasting and decision making horizon [LeB01a, p. 256]. Examples of 'myopic' agents can be found in Farmer (1998) [Far98], Beltratti et al. (1996) [BMT96] or Lux (1995) [Lux95]. However, to approach a market in a more realistic manner, the inclusion of long *and* short term investors is desirable [LeB01b, p. 225-7].

An example of multi-period decision schemes is given in LeBaron (2001b) [LeB01b]. In this model, the agents either place their funds in a stock paying a stochastic dividend or in a risk free asset. As a specialty of the artificial market, the agents maximize an infinite-horizon utility function. The agents have access to a collective pool of forecast models. The forecast models consist of simple feedforward neural networks, which have been fitted to historical data. The agents chose among the different forecast models by taking into account their individual planning horizons. Long-term investors refer to relative performance measures looking back into the distant past, while short-term investors evaluate the most recent performance of the forecast models. As the population of agents evolves, unsuccessful agents are removed from the market and replaced by new individuals. In this birth-and-die process, the planning horizon used by successful agents is overemphasized in the generation of new agents.

A further example of multi-period decision schemes is provided by Lettau and Uhlig (1999) [Let99]. The authors refer to a classifier system in order to solve the intertemporal consumption plans of the agents.

Nevertheless, the assumption of myopic agents who rely on a short-term decision making schemes is somewhat reasonable, because it is "not clear how well actual people stay with long-term plans in financial markets as opposed to simple adapting to current data and following simple rules of thumb" [LeB01a, p. 256].

2.1.4 CONTAGIOUS DECISION MAKING

Finally, let us comment on possible relationships between the decision making processes of single agents. On the one hand, one may design the decision schemes of the agents to be independent from each other. This means, that the decision making of agent a does *not* have a direct impact on the decision making of agent b. Examples of independent decision making can be found in Farmer (1998) [Far98], Levy et al. (1994) [LLS94], Gode and Sunder (1993) [GS93] or Yang (1999) [Yan99].

On the other hand, the decision making of agent a may influence the behavior of agent b. In other words, "a speculator will be more willing to buy (sell) if he sees most other traders buying (selling)" [Lux95, p. 883]. The underlying assumption of this behavior is, that the speculator believes that the other market participants have a better information basis about the future market development. We refer to this process as contagious decision making or herding behavior [Lux98, Lux95, p. 148-54 and p. 883]. Contagious decision making causes self-reinforcing market price fluctuations [Lux95, p. 883].

As a representative example of contagious decision making, let us consider the multi-agent model of Lux (1998) [Lux98]. The artificial stock market is populated with 'noise traders' (chartists) and fundamentalists. The noise traders are either optimists or pessimists [Lux98, p. 148-9]. The agents are allowed to switch between the groups of chartists and fundamentalists. Switches between the groups of noise traders and fundamentalists occur with a probability which depends on the relative attractiveness (pay-off differential measured by excess profits) of the strategies [Lux98, p. 151-54]. In addition, the model permits movements within the group of noise traders, i. e. noise traders may change from optimistic to pessimistic behavior and vice versa. Movements within the group of noise traders occur with a specific transition probability that depends on the majority opinion of noise traders and the observed market trend [Lux98, p. 151-54]. Since strategy switching is incorporated into the model on a probabilistic basis, "we always have a small probability of transitions against the trend" [Lux98, p. 151]. It is worth noticing, that the artificial market exhibits *all* interesting qualitative features of real-world financial time series [Lux98, p. 155-62]. The key feature for the stability of the market dynamics is the development of the noise trader population. Turbulent market situations emerge when the portion of noise traders is close to a critical value. However, the market dynamics has a globally stable equilibrium in which market prices track fundamental values [Lux98, p. 154-56].

Further examples of contagious decision making are given in Beltratti et al. (1996) [BMT96], Sornette et al. (1998) [SJ98], Kaizoji (2000) [Kai00] and Bak et al. (1996) [BPS96].

2.2 OBJECTIVE FUNCTIONS

An issue closely connected with the modeling of the agents' decision making processes is the design of an appropriate objective function.

The relationship between the decision making process and the objective function is especially obvious in case of forecasting agents (Fig. 6.6). Already mentioned, the decision scheme of a forecasting agent incorporates two stages. First, the agent forms an expectation about the future market development. Afterwards, the prediction is evaluated by a specific objective function or decision rule [BMT96, p. 174]. The objective function is therefore an integral part of the agent's decision making scheme. Since the decisions are always deduced from the objective function, the trading activities of a forecasting agent are always goal oriented.

In the following we give an overview of objective functions frequently used in multi-agent modeling. As a starting point for this overview, we provide a classification of objective functions which is based on their structural representation in the decision making schemes of the agents. The classification of the agents' objective functions is depicted in Fig. 6.9.

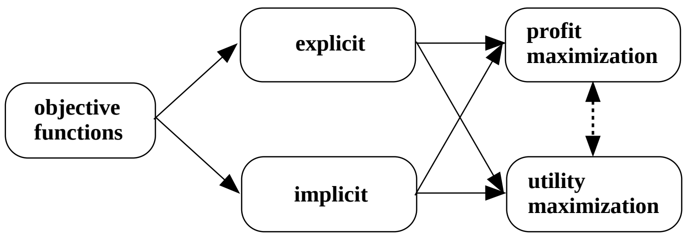


Figure 6.9 Classification of agents' objective functions

As shown in Fig. 6.9, one may distinguish between two major categories of objective functions. The first category refers to objective functions which are explicitly incorporated into the decision making of an agent. This means, that the agent has a well defined objective function which is directly included in his decision making process. Explicitly formulated objective functions can mainly be found in the decision making schemes of forecasting agents. Examples of explicit objective functions are given in Zimmermann et al. (2001d) [ZNG01d], Chakrabarti (2000) [Chak00] and Yang (1999) [Yan99].

The second category contains decision making schemes that only implicitly reflect the objectives of the agents. In other words, the objectives of an agent are only indirectly incorporated into his decision making scheme. Such representations are mainly associated with rule-based agents, who adapt real-world trading strategies. For instance, the chartists in the model of Farmer (1998) [Far98, p. 26-9] incorporate a single technical rule that aims to identify market trends. Although these agents have no explicitly formulated objective function, the underlying assumption of the trading strategy is to participate from market trends in order to maximize the trading profit.

Explicit as well as implicit objective functions can be subdivided into profit and utility maximization tasks. As indicated by the dashed arrow in Fig. 6.9, these subgroups are closely related: Typically, multi-agent models incorporate utility functions that only depend on the wealth of the agents (see e. g. Levy et al. (1994) [LLS94, p. 104-6] or Yang (1999)

[Yan99, p. 7-8]). The wealth of an agent is defined as the total value of his investment portfolio. This implies, that an increased wealth spends a higher level of utility. Since profit maximization is one way to increase the agents' wealth, there are no conflicts between these objectives within the latter framework. However, it should be noted that "utility maximization alone is not synonymous with wealth maximization" [LeB01a, p. 258]. For instance, risk averse agents may own well diversified investment portfolios, "but in the long run, these may not make them leaders in terms of wealth maximization" [LeB01a, p. 258].

In the following we discuss the categories of explicit and implicit objective functions in greater detail and give examples for each group.

2.2.1 EXPLICIT OBJECTIVE FUNCTIONS

Agents who are endowed with an explicitly formulated objective function mainly belong to the group of forecasting agents (Fig. 6.6). The objective function is an inherent part of the agents' decision scheme: First, the agents predict the future development of the market. Thereafter, the forecasts are evaluated by the agents' objective functions [BMT96, p. 174]. Within the category of explicit objective functions one may distinguish between profit and utility maximization tasks (Fig. 6.9).

Expected profit maximization. Profit maximizing agents trade on market price fluctuations such that their expected gain is maximized. An example of agents maximizing their expected profits is given in Zimmermann et al. (2001d) [ZNG01d, p. 737]. Each agent is endowed with a so-called '*prof-max*' objective function [Ref95, p. 69-70]. Given a set of expected price shifts $\hat{\pi}_{t+1} = E(\ln(p_{t+1}/p_t))$ with $t = 1, \ldots, T$ the *prof-max* objective function of an agent i can be formulated as

$$\text{prof-max} = \frac{1}{T}\sum_{t=1}^{T} \hat{\pi}_{t+1} \cdot \alpha_t^i \rightarrow \max_{\alpha_t^i} \,, \tag{6.1}$$

where α_t^i is the agent's trading decision at time period t. The *prof-max* objective function guides the decision making such that the agent gains from expected market price fluctuations $\hat{\pi}_{t+1}$. For instance, if the agent expects declining market prices ($\hat{\pi}_{t+1} < 0$), he can only make a profit by taking a short position in the asset ($\alpha_t^i < 0$). On the other hand, the agent enters a long position ($\alpha_t^i > 0$), if he expects increasing prices ($\hat{\pi}_{t+1} > 0$). This behavior leads to a maximization of the expected profits. Remarkably, the *prof-max* objective function does not only focus on the correctness of underlying forecasts, but also on the expected profit of each trading decision. Note, that the *prof-max* objective function is often optimized as an error minimization task by turning the sign of Eq. 6.1 [ZNG01d, p. 738].

Another example of profit maximizing agents who are endowed with explicit objective functions can be found in Chakrabarti (2000) [Chak00]. The author models the intra-day trading of agents in an inter-bank foreign exchange market. The risk averse agents try to maximize their expected end-of-day profits [Chak00, p. 35-6]. The objective functions of the agents take into account the costs of bearing risk during the day as well as the risk exposure caused by own overnight foreign exchange inventories [Chak00, p. 35]. The market as well as the inventory risk are incorporated into the agents' objective functions in form of penalty terms [Chak00, p. 35-6].

An interesting approach to profit maximizing agents is present by Kaizoji (2000) [Kai00]. The agents try to predict the behavior of the majority [Kai00, p. 495-7]. The underlying assumption of this approach is, that price changes result from the majority opinion of all trades in the market. For instance, prices will decline (increase), if the number of sellers (buyers) is higher than the number of buyers (sellers). From this it follows, that an agent can make profits by anticipating the behavior of the crowd. Relying on the latter assumption, the agents try to minimize a so-called 'disagreement function', which takes into account the strength of the interferences among the agents [Kai00, p. 496].

Utility maximization. Let us now turn to explicitly formulated utility function concepts. In the majority of agent-based models, a utility function measures the agents' benefit resulting from a particular combination of assets (e. g. a risk-less bond and a risky stock) [WHD02, p. 90]. Increases of the portfolio wealth are associated with higher levels of utility. The utility function may also represent other preferences of the agents. Examples of such preferences are the agents' investment attitudes or risk aversion characteristics [LeB01a, p. 255].

Probably one of the most remarkable utility concepts in multi-agent modeling is the so-called constant absolute risk aversion (CARA) utility framework. The CARA utility functions are used in various multi-agent models, e. g. Yang (1999) [Yan99], Lettau (1997) [Let97], the Santa Fe artificial stock market [PAHL94, PAHL98] or Chen et al. (2001) [ChYe01]. Modifications or enhancements of the CARA utility concept are presented in e. g. Hommes (2001) [Hom01], Chiarella et al. (2001) [ChHe01] or Brock et al. (1997) [BH97].

In the CARA framework [LeB00, LAHP97, PAHL98, Yan99, JPB00, p. 690-1, p. 8-9, p. 29-30, p. 7-10 and p. 467-8], the utility function U_i

of a single agent i can be written as

$$U_i(W_{i,t}) = -\exp(-\lambda W_{i,t}) \ , \tag{6.2}$$

where $W_{i,t}$ is the wealth of agent i at time period t. The parameter λ determines the degree of the agent's relative risk aversion. Suppose, that the agent has two investment opportunities, a risky stock and a riskless bond. The agent can split his funds among these investment alternatives. Thus, the wealth of the agent at time period t can be written as

$$W_{i,t} = M_{i,t} + P_t h_{i,t} \ , \tag{6.3}$$

where $M_{i,t}$ are the funds agent i has placed into the riskless asset, $h_{i,t}$ are the number of risky shares agent i holds and P_t is the price of the stock. In the next time period $t+1$, the risky stock issues a stochastic dividend d_{t+1} and the risk free bond pays an interest r. The future wealth $W_{i,t+1}$ of agent i is therefore given by

$$W_{i,t+1} = (1+r)M_{i,t} + (P_{t+1} + d_{t+1})h_{i,t} \ . \tag{6.4}$$

On the basis of Eq. 6.4, the objective of the agent is to *maximize* the expected utility $E(U_i(W_{i,t+1}))$:

$$E(U_i(W_{i,t+1})) = E(-\exp(-\lambda W_{i,t+1})|u_{i,t}) \ . \tag{6.5}$$

Given a set of external influences $u_{i,t}$, the agent's conditional utility expectation $E(U_i(W_{i,t+1}))$ depends on his future wealth $W_{i,t+1}$. Under the additional assumption, that the agent's price and dividend expectations $E_i(P_{t+1} + d_{t+1})$ are Gaussian distributed, one may calculate the agent's demand of risky shares $h_{i,t}$, that spends the highest expected utility. The agent's demand $h_{i,t}$ for the risky stock is

$$h_{i,t} = \frac{E_i(P_{t+1} + d_{t+1}) - P_t(1+r)}{\lambda \sigma^2_{i,P_{t+1}+d_{t+1}}} \ , \tag{6.6}$$

where $\sigma^2_{i,P_{t+1}+d_{t+1}}$ is the variance of the Gaussian distributed price and dividend expectations $E_i(P_{t+1} + d_{t+1})$. According to Eq. 6.6, the agent "hold[s] positions that are linear in the difference between the expected returns of the stock and the rate of interest" [Yan99, p. 8].

One of the most important design issues of the CARA utility function concept is the construction of the price and dividend expectations $E_i(P_{t+1} + d_{t+1})$, because these forecasts are required to build optimal portfolios. For instance, the value traders in the model of Yang (1999) [Yan99, p. 10-12] use recurrent neural networks for this task, whereas the 'conditional forecasting' agents in the Santa Fe artificial stock market choose from a set of forecasting rules that is evolved by genetic algorithms [PAHL94, p. 29-30].

Other examples of explicit utility functions are given in LeBaron (2001b) [LeB01b], Chiarella et al. (2001) [ChHe01], Levy et al. (1994) [LLS94], Sornette and Johansen (1998) [SJ98] or Arifovic (1996) [Ari96].

The agents in the model of LeBaron (2001b) [LeB01b] have constant relative risk aversion (CRRA) preferences with a common discount factor. The CRRA preferences are formulated in a logarithmic form. The agents either place their funds in a risky asset or a riskless bond. With regards to the CRRA preferences, the agents maximize the expected return of their investment portfolios. The portfolio decisions of the agents are based upon feedforward neural networks. Heterogeneous agents with CRRA utility functions are also used by Chiarella et al. (2001) [ChHe01] and Levy et al. (1994) [LLS94].

In the model of Sornette and Johansen (1998) [SJ98] the utility function of the agents determines their willingness and timing for investing in the market. All traders are equipped with the same utility function concept. Heterogeneity is introduced by assuming, that the agents need different time horizons to come to a trading decision and to enter the market.

Arifovic (1996) [Ari96] introduces agents who maximize their utility over a two-period horizon by determining their holdings of two currencies. The currency holdings are the only way to save money from period t to $t + 1$. The foreign exchange inventories are entirely used to buy a consumption good. The utility of the agents depends on their overall consumption in both time periods.

2.2.2 IMPLICIT OBJECTIVE FUNCTIONS

Agents incorporating an implicit objective function mainly belong to the group of rule-based agents (see Fig. 6.6 and 6.9). The decision making schemes of these agents are typically represented by at least one trading rule, which is often adapted from real-world trading strategies [LeB01a, BMT96, p. 255 and p. 174].

For instance, fundamentalists compare the actual market price to a specific estimation of the asset's fundamental value. If the market price is below the fundamental value, fundamental traders buy the asset, since they expect that the market price will revert to the fundamental value. Examples of fundamental traders are given in Farmer (1998) [Far98, p. 16-26] or Lux and Marchesi (1999) [LM99, p. 498]. Another example of rule-based agents are technical traders (chartists) who rely on historical price patterns. Chartists try to gain from market price trends, which can be identified by technical trading rules [Far98, p. 26-9].

As it can be seen from these examples, the agents "do not operate with any well-defined objective function" [LeB01a, p. 255]. Nevertheless, the goal of rule-based trading strategies is to maximize the expected profits of the agents.

Agents who only operate with an implicitly formulated objective function can be found in the artificial markets of e. g. Farmer (1998) [Far98], Lux (1995) [Lux95], Steiglitz et al. (1997) [Ste97], De la Maza and Yuret (1995) [MY95], Beltratti, Margarita and Terna (1996) [BMT96] or Gode and Sunder (1993) [GS93].

Already mentioned above, the models of Farmer (1998) [Far98] and Lux (1995) [Lux95] incorporate populations of fundamental and technical traders. Following their trading strategies, these agents implicitly maximize their expected profits. This is also true for the value investors and trend traders in the market model of Steiglitz et al. (1997) [Ste97].

In the model of De la Maza and Yuret (1995) [MY95], agents explore trading rules on the basis of genetic algorithms. During the evolutionary learning process, the trading rules are judged by a fitness criterion which measures the historical performance (realized profits) of each strategy [MY95, p. 293 and p. 301]. Since profitable trading rules are preferred during the learning, the behavior of the agents is mainly driven by a profit maximization calculus.

Remarkably, the forecasting agents in the stock market model of Beltratti, Margarita and Terna (1996) [BMT96] are also endowed with an implicitly formulated objective function. In the model, the agents meet randomly at the market place. Let us consider two agents, i and j [BMT96, p. 179-81]. Both agents have different expectations about the stock price. The mean value of the agents' price expectations is the price at which the transaction takes places. If the price expectation of agent i is lower (higher) than that of agent j, agent j will buy (sell) one share from (to) agent i. In other words, a transaction takes place because the agents i and j have different expectations about the stock price [BMT96, p. 180]. This price formation mechanism can be seen as "the focal point of a non-cooperative bargaining game between two players who have to split a pie" [BMT96, p. 180]. By averaging the price expectations, the expected gain is equally splitted among the two agents [BMT96, p. 180]. The underlying objective function of the agents is therefore an expected profit maximization task. Note, that the price expectations of the agents are based on feedforward neural networks with different complexity [BMT96, p. 223-8].

The behavior of the zero-intelligence traders of Gode and Sunder (1993) [GS93, p. 121-3] is basically random and consequently, the agents do *not* have an explicitly formulated objective function. In a second series of experiments, the authors employed an additional budget constraint to their agents. If the budget constraint is satisfied, a "buyer would not bid more than what the asset is worth in redemption value, and a seller will not offer below costs" [LeB00, p. 684]. The additional restriction can be seen as an implicit profit maximization task: Buyers will not accept exaggerated prices and on the other hand, agents will not sell at a loss [WHD02, p. 115-6].

2.3 HETEROGENEITY

The design of the agents' decision making schemes also incorporates the topic of heterogeneity. Heterogeneity means that the behavior and decision making of the agents differs in a given market situation [BMT96, WHD02, LeB00, p. 178, p. 89-92 and p. 680]. For instance, in an up-warding trend market, some agents, who believe that the trend continues, may decide to buy assets, while others, who expect a trend reversal, sell. Generally, heterogeneous decision making is required to adapt the underlying market dynamics. If the agents' decisions are too homogeneous, the market tends to be highly volatile. In contrast, if the agents are too heterogeneous, one may only observe small price movements. Both cases do not reflect real-world behavior [WHD02, BMT96, ZNG01d, p. 124, p. 177 and p. 741].

As shown in Fig. 6.10, heterogeneous decision making can be introduced to the agents' behavior in *four* different ways [BMT96, WHD02, p. 178 and p. 89-92]: (*i.*) by a varying information basis, (*ii.*) by different parameter settings, (*iii.*) by miscellaneous agent types, or (*iv.*) by the learning and evolution of the agents. In the following, we explain these ways of incorporating heterogeneity into the agents' decision making in more detail.

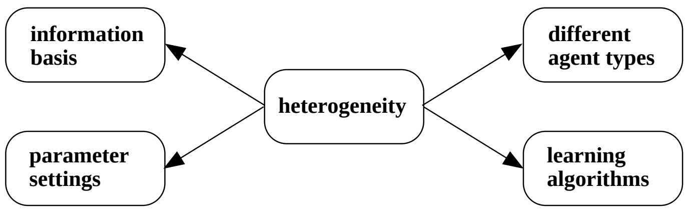


Figure 6.10 Sources of heterogeneous decision making behavior

2.3.1 INFORMATION BASIS

Presuming, that the underlying decision making processes of the agents are homogeneous [BMT96, p. 178], the first source of heterogeneity is a varying information basis (Fig. 6.10). The underlying assumption is that the agents rely on different types of information [WHD02, p. 90-1]. For example, some agents may stress fundamental indicators, whereas other agents evaluate historical price patterns to come to a trading decision. We refer to this source of heterogeneity as a process of information filtering [Zim94, ZNG01d, p. 3-4 and p. 737].

A related way of modeling heterogeneity would be to spread the information asymmetrically among the agents [WHD02, p. 90-1]. In this case, some agents may only have access to endogenous market information (e. g. previous market prices or trading volume), while others are limited to exogenous data (e. g. dividends or interest rates).

Another possibility to characterize the agents' information sets is to distinguish between private (insider) and public information. An example of such an asymmetric information flow is given in Chan et al. (1999) [CLeLP99]. Among a variety of other experiments, the authors study the behavior of insiders in a financial market with a single asset paying a specific dividend at the end of each trading period. In contrast to partially informed traders, insiders know the upcoming dividend in advance [CLeLP99, p. 13]. According to the results of Chan et al. (1999) [CLeLP99, p. 17-8], insiders have a higher wealth than traders who only rely on public information. This wealth difference "represents the value of the insider information and may be an estimate of the price traders would be willing to pay if information signals were sold" [CLeLP99, p. 18].

2.3.2 PARAMETER SETTINGS

In the presence of homogeneous decision making processes and common information, heterogeneity may also result from different parameter settings of the agents' decision making schemes. Heterogeneity is introduced by variations of the agents' personal characteristics such as risk aversion coefficients, psychological factors or planning horizons [BMT96, WHD02, p. 174 and p. 89-91].

For instance, the agents in the model of Levy et al. (1994) [LLS94] are homogeneous in the sense that *all* traders have the same type of utility function. The authors introduce heterogeneity by assuming, that each agent is individually influenced by unknown psychological factors. These effects are incorporated into the decision making processes by adding a random variable to the agents' capital allocation decisions [LLS94, p.

105-6]. – "This randomness is the source of heterogeneity"[LLS94, p. 106].

In the artificial market of LeBaron (2001b) [LeB01b], the decision making of each agent is based on an infinite-horizon time-separable utility function. Heterogeneity among the agents is introduced by allowing each agent to chose an individual planning horizon. This means, that the decision making of some agents is based on long-term planning horizons, while others have a short-term calculus [LeB01b].

All agents in the model of Sornette (1998) [SJ98] are endowed with the same type of utility function, which "determines their willingness and timing for buying the stock" [SJ98, p. 585]. The agents are "heterogeneous in the sense that the time they need for their analysis is different for each of them and hence each trader has a characteristic time to form his decision and enter the market" [SJ98, p. 585].

Among other experiments, Farmer (1998) [Far98, p. 20-3] studies the dynamics of financial markets only consisting of so-called value investors (fundamentalists). These rule-based agents use entry and exit thresholds to determine their long and short positions (see Fig. 6.7). The decision making processes (trading strategies) are homogeneous. Heterogeneity among the agents results from different entry and exit thresholds, which are drawn from a uniform random distribution [Far98, p. 21-2].

2.3.3 AGENT TYPES

Heterogeneous decision making can also be achieved by the interaction of different agent types (Fig. 6.10) [WHD02, p. 91-2].

Probably the most common types of agents are chartists and fundamentalists. The decision making of chartists is based on the analysis of historical market prices patterns, while fundamentalists derive their trading decisions by comparing the asset's fundamental value to the current market price [LM99, FJ99, p. 498 and p. 8-18]. Examples of chartists and fundamentalists can be found in Bak et al. (1996) [BPS96], Farmer (1998) [Far98], Lux (1998) [Lux95] or Castiglione (2000) [Cast00].

A more sophisticated approach is presented in Chan et al. (1999) [CLeLP99]. The authors introduce *three* agent types [CLeLP99, p. 9-12]: (*i.*) momentum traders, (*ii.*) nearest-neighbor traders and (*iii.*) Bayesian traders.

Momentum traders are similar to chartists. These agents simply expected that "tomorrow's return is today's return" [CLeLP99, p. 10]. The trading strategy of the momentum traders is fixed, i. e. these agents do not learn. According to Chan et al. (1999), the momentum traders "reinforce and magnify the ups and downs of price movements, introducing extra volatility and irrational valuations of the security which make information aggregation and dissemination more difficult" [CLeLP99].

The technical trading strategy of the nearest-neighbor traders is more complicated [CLeLP99, p. 12]: Nearest-neighbor traders form a sequence of n-tuples x_t^i $(t = k, k+1, \ldots, T)$ from recent market prices p_t, i. e. $x_t^i = (p_{t-n+1}, p_{t-n+2}, \ldots, p_t)$. Each price tuple is associated with an end-of-period rational expectations equilibrium (REE) price D_i, i. e. we get the pairs (x_k^i, D_k), (x_{k+1}^i, D_{k+1}), $\ldots$, (x_T^i, D_T). To forecast the market development, the r nearest neighbors (Euclidean distance) of the most recent market price tuple x_t^i are identified. The market forecast is the average of the r REE prices associated with the neighboring price tuples.

Bayesian traders can be seen as the fundamentalists of the market. These agents form an expectation of a so-called 'base price' (fundamental value) on a probabilistic basis. If the estimated base price is higher (lower) than the actual market price, Bayesian traders buy (sell) the asset [CLeLP99, p. 10-2]. During the trading hours, "Bayesian traders continuously observe market activities, update their beliefs, and adjust their positions accordingly" [CLeLP99, p. 10]. If the market price is almost equal to the fundamental base price, Bayesian traders close their positions.

Another example of different agent populations is given in Beltratti, Margarita and Terna (1996) [BMT96, p. 223-6]. The authors introduce so-called smart and naive agents. The decision making schemes of both agent types is based on feedforward neural networks. Heterogeneity is introduced by the design of the network architectures (see Fig. 6.8). While naive agents have limited skills, smart agents have superior abilities to discover nonlinear relationships in the data (i. e. more hidden neurons) [BMT96, p. 223]. Additional sources of heterogeneity are different learning abilities and a varying information basis [BMT96, p. 224-5].

2.3.4 LEARNING ALGORITHMS

Finally, heterogeneous behavior may originate from the learning and evolution of the agents (Fig. 6.10) [BMT96, LeB01a, p. 178 and p. 255-6].

Multi-agent models that refer to this source of heterogeneity typically incorporate agents who are endowed with a set of (dynamic) trading rules. The rule sets are evolved by genetic algorithms. In such a model, "agents are very homogeneous at the start in their abilities and strategy structures" [LeB01a, p. 255]. During the market process, the agents learn and develop more sophisticated trading strategies. In other words, "differences in behavior and strategy [...] evolve endogenously as the market runs, [and thus] agent heterogeneity becomes a changing feature of the market" [LeB01a, p. 255]. Examples including genetic algorithms as a source of heterogeneity are given in De la Maza and Yuret (1995) [MY95], Arifovic (1996) [Ari96], LeBaron (2001b) [LeB01b] or the Santa Fe artificial stock market [PAHL94, PAHL98].

Besides genetic algorithms, one may also refer to gradient-based learning techniques in order to generate heterogeneous decision behavior. First, the agents may simply differ in their underlying learning techniques. – Different ways of learning should lead to heterogeneous agents. Even if the applied learning mechanism is the same for all agents, heterogeneous decision behavior may arise [BMT96, p. 178 and p. 225]: In this case, the agents are endowed with different learning parameters. An example of such an approach can be found in the model of Beltratti et al. (1996) [BMT96, p. 225]. The stock market is populated with smart and naive agents. Both agent types learn with standard error backpropagation (see chp. 5). However, the learning behavior of smart and naive agents is different: Smart agents can learn faster than naive ones, because they use more effective parameters for the learning [BMT96, p. 225].

2.4 LEARNING AND EVOLUTION

Finally, let us consider how the agents evolve their trading strategies, adapt to changing market conditions and learn, how to improve their behavior. Already mentioned, learning is also an important source of heterogeneous decision behavior (see Fig. 6.10). Generally spoken, 'learning' or 'adaptive behavior' means, that the agents are able to modify parts of their decision making schemes in a specific manner. By this, the agents adapt their behavior to changing market conditions [WHD02, p. 91-2].

The scope of learning and adaptive behavior is multifaceted: For example, rule-based agents (see Fig. 6.6) may either develop completely new trading strategies or change a few parameters of already existing ones. Another possibility is that rule-based agents switch between different trading strategies as the market evolves. This corresponds to contagious decision making or so-called herding behavior [Lux98, Lux95, p. 148-54 and p. 883]. Moreover, forecasting agents (see Fig. 6.6) learn by fitting the free parameters of their forecast models to historical data. During the learning, the agents generate a structural hypothesis of the underlying market dynamics, i. e. they try to identify invariant structures out of varying time series. The generated structures allow the agents to predict the future development of the market price [NZ98, p. 413-17].

As it can be seen from this outline, the agents may refer to a broad spectrum of learning techniques, which ranges from e. g. gradient based optimization methods to evolutionary algorithms. Fig. 6.11 depicts an approach to classify learning techniques that are frequently used in multi-agent modeling.

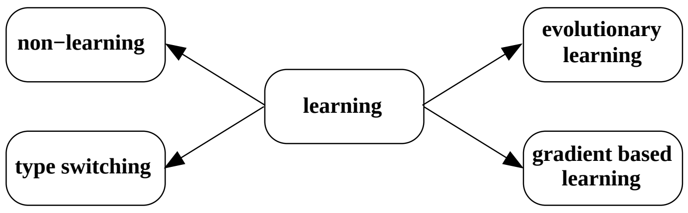


Figure 6.11 Classification of learning techniques frequently used in multi-agent modeling

Learning in agent-based financial markets can be subdivided into *four* categories: (*i.*) non-learning agents, (*ii.*) type switching agents, (*iii.*) agents evolved by evolutionary algorithms and (*iv.*) agents relying on gradient based optimization techniques. In the following we give a more detailed description of these categories.

2.4.1 NON-LEARNING AGENTS

In the majority of cases, non-learning agents are only endowed with a single trading rule (see Fig. 6.6). These rules are 'hard-wired'. This means, that the agents cannot adapt the underlying parameters of the trading strategies to changing market conditions. We speak of static trading rules.

A reason for such a design is, that the related multi-agent models only try to capture the market dynamics which results from the interaction of heterogeneous agents using real-world trading strategies, rather than focusing on the evolvement of the agents' behavior [Far98, p. 15].

An example of non-learning agents using static trading rules is given in Farmer (1998) [Far98, p. 15-33]. Among other experiments, the author studies the interaction of value investing and trend following agents who are endowed with market entry and exit thresholds (see Fig. 6.7). The thresholds of each agent are initially drawn from a uniform distribution with predefined limits. Both trading strategies are 'hard-wired': Neither the thresholds, nor any other specification of the trading rules (e. g. the amount of the long and short positions) can be modified.

Another example of non-learning agents can be found in the model of Gode and Sunder (1993) [GS93, p. 121-3]. The zero-intelligence traders deduce their bids and offers at random over a predefined range. In a second series of experiments, the agents are endowed with an additional budget constraint, which restricts their bids and offers. However, zero-intelligence traders do not learn, i. e. the parameters of their trading strategies remain constant over time [WHD02, p. 115].

2.4.2 TYPE SWITCHING AGENTS

The category of type switching agents builds upon the behavior of non-learning agents: In some multi-agent models, the agents are enabled to switch between different 'hard-wired' trading strategies [Lux95, p. 882-3]. For instance, in a specific market situation an agent might discover that a technical trading strategy yields a higher profit than a fundamental one. In this case, the agent will change his behavior accordingly, i. e. he switches from the technical to the fundamental trading strategy. Agents switching between several (static) trading strategies belong to the class of dynamic rule-based agents (see Fig. 6.6). According to Huberman (1996) [Hub96, p. 9] a strategy switching mechanism can be seen as a form of distributed learning:

> "For example, strategy optimization might involve agents observing the behavior and success of a few other agents and then picking a good strategy based on that information; or perhaps, agents keep records of how well strategies have performed in the past and make decisions based on that information. In either case, and indeed quite generally, the net effect of the strategy optimization procedure will be to change the usage of a specific strategy according to its overall performance. Note that we have thus explicitly assumed that agents do not have knowledge about which of the possible strategies is best. Instead, they make their strategy choices as the system continues to evolve. The strategy switching mechanism can thus be thought of as a way of learning in a distributed manner, with no single agent having full knowledge of the underlying model or state of the entire system."

The multi-agent models of Lux (1998) [Lux98, p. 148-54] and Lux and Marchesi (1999) [LM99, p. 498] are two examples of type switching agents: Chartists and fundamentalists meet randomly at the market place and compare the expected profitability of their trading strategies. A fundamentalist will become a chartist, if technical trading seems to be more profitable. In contrast, a technical trader switches to the group of fundamentalists, if value oriented strategies seem to be more promising. Switches between the different trading strategies are formulated on a probabilistic basis. The switching behavior of the agents implies a shift towards a specific strategy which is currently outperforming all others. Therefore, strategy switches have strong impacts on the dynamics and stability of the market. The authors report that if the portion of the noise traders in the market exceeds a specific critical value, the fundamental market equilibrium becomes unstable [Lux98, p. 156-59].

2.4.3 EVOLUTIONARY LEARNING

Rather than studying the interaction of several 'hard-wired' trading strategies, evolutionary market models mainly focus on the adaptive behavior of the agents. At the beginning of a market simulation, evolutionary agents are typically endowed with homogeneous abilities and trading strategies. As the market process evolves, the agents endogenously adapt their trading strategies and may also explore completely new trading rules [LeB01a, p. 255]. In other words, "agent heterogeneity becomes a changing feature of the market [which] can often be important in studying traditional finance questions related to market liquidity, or how hard it is to find someone to trade with" [LeB01a, p. 255]. In the majority of cases, these multi-agent models are build upon biologically inspired evolutionary learning schemes like e. g. genetic algorithms.[6]

The design of evolutionary multi-agent models can be quite general. This means, that one only has to make a few assumptions about statistical building blocks of common trading strategies, the agents initial structure, their information basis and their interaction, whereas the specification and development of the trading strategies are tasks of the evolutionary learning. In addition, "one of the practical appeals of [evolutionary learning] is that it can be used to take parameters for which one's prior beliefs are weak and put them under evolutionary control. In other words, give up on trying to set these to some arbitrary value, and let evolution decide their levels" [LeB01a, p. 258].

However, evolutionary learning comes along with a few disadvantages: First, evolutionary learning has explanatory drawbacks. This means, that new trading strategies "[evolve] out of a pure unformed soup of genetic material" [LeB01a, p. 255], rather than being explained or deduced from economic reasoning. In other words, "learning and evolution takes place in a space that is somewhat disconnected from the real one" [LeB00, p. 682].

Second, evolutionary learning increases the computational complexity of the multi-agent model [LeB01a, p. 255]. For instance, the modeling of

[6]Genetic algorithms are a well-known optimization method, which is based upon the biological principle of natural evolution. The theory of genetic algorithms was originally developed by John Holland in the 1970's [Hol92]. A basic introduction to genetic algorithms can be found in Beltratti et al. (1996) [BMT96, p. 22-42].

each agent may require a separate evolutionary algorithm. In this connection, one should also notice that "these artificial intelligence tools are still poorly understood operating alone, and in groups even less known" [LeB01a, p. 255].

Evolutionary learning schemes may also generate trading rules which are over-specified or too complex [LeB01a, p. 255-6]. A crucial issue is that it is probably difficult to provide economic interpretations for the generated rules. In addition, complex rules most likely do not reflect real-world behavior. Finally it should also be noted, that the parameterization of an evolutionary learning mechanism is also a critical question during the modeling [LeB00, p. 682-3].

Examples of agent-based financial markets incorporating evolutionary learning algorithms are given in Lettau (1997) [Let97], Arifovic (1996) [Ari96], LeBaron (2001b) [LeB01b], De la Maza and Yuret (1995) [MY95] and the Santa Fe artificial stock market [PAHL94, PAHL98].

Let us describe some of these examples in greater detail: The agents in the model of Lettau (1997) [Let97] use genetic algorithms to learn investment rules determining the optimal portfolio weighting of a risky asset. The genetic algorithms incorporate so-called mutation and crossover operators. Mutation leads only to a slight modification of an existing investment rule, whereas crossover combines two existing rules into a completely new one [BMT96, p. 24-26]. In order to determine the fitness (performance) of the investment rules, the author measures the expected utility as a function of the agents' wealth.

In Arifovic (1996) [Ari96] the implemented genetic algorithm encodes the agents' choice variables. The decision making of the agents is therefore directly encoded in the genetic algorithm. The agents are evolved by standard genetic operators like mutation and crossover. Furthermore, a so-called election operator is invented, which prevents poor offspring from entering the genetic population [LeB00, p. 686]. As a fitness measure, Arifovic (1996) [Ari96] refers to the realized utility levels of the agents.

In the artificial market of LeBaron (2001b) [LeB01b], the agents use forecasting rules (models) in the form of feedforward neural networks. The forecasting rules are evolved by a genetic algorithm with mutation, crossover and 'new weight' operators. The 'new weight' operator changes a single weight in a selected neural network to a random value. As a fitness measure, the author considers the 'popularity' of the forecasting rules [LeB01b, p. 235]: "A rule can be a parent for the next generation if at least one agent has used it over the last 10 periods. Rules that haven't been used for 10 periods are marked for replacement". The agents evaluate new forecasting rules by computing their historical performance.

The goal of the evolutionary learning scheme is "to produce new and interesting strategies that must then survive the competition with the other rules in terms of forecasting" [LeB01b, p. 235].

2.4.4 GRADIENT-BASED LEARNING

The fourth category includes agents who learn with gradient based optimization techniques (see Fig. 6.11). Gradient based learning can mainly be found in the group of forecasting agents (see Fig. 6.6). Here, learning is associated with the optimization of the underlying forecast models on the basis of historical data.

Learning from data means, that the agents generate a hypothesis of the true time-invariant structures of the market dynamics. The underlying assumption is that the explored structures enable the agents to predict the future development of the market (see chp. 5).

A typical example of a gradient based learning algorithm is standard error backpropagation, which is used for the training of neural networks. In combination with a particular learning rule for changing the network weights w, learning by error backpropagation generates a structural hypothesis about the market dynamics, which in turn is the basis of forecasting (see chp. 5).

The learning techniques of forecasting agents are implicitly determined by their underlying forecast models. For instance, an agent who generates his forecasts on the basis of a linear regression model will refer to ordinary least square estimates of the model parameters [PDP00, p. 211-30]. In contrast, an agent who is endowed with a neural network learns with error backpropagation. The learning algorithms of forecasting agents are therefore multifaceted and depend on the underlying forecasting techniques [LeB01a, p. 256].

Examples of forecasting agents who are endowed with gradient based learning techniques can be found in Yang (1999) [Yan99], Beltratti, Margarita and Terna (1996) [BMT96], Zimmermann et al. (2001d) [ZNG01d] and Zimmermann et al. (2001b) [ZNG01b].

The model of Yang (1999) [Yan99] incorporates three types of agents, value, momentum and noise traders. Value traders utilize Elman's recurrent neural networks [CK01, p. 21-2] to predict the fundamental value of the asset. The networks are trained by a modified version of standard error backpropagation [Yan99, p. 10-2]. Momentum traders rely on moving average strategies. The parameters of the moving average processes are estimated by ordinary least squares [Yan99, p. 12-3]. Noise traders do not learn and post random market orders [Yan99, p. 13].

The smart and naive agents in the model of Beltratti, Margarita and Terna form their expectations on the basis of feedforward neural networks with different complexity [BMT96, p. 224-5]. Both agent types learn with standard error backpropagation (see chp. 5). Remarkably, the agents have different learning abilities, which result from varying learning parameter sets [BMT96, p. 225].

The agents in the models of Zimmermann et al. (2001d) [ZNG01d] and Zimmermann et al. (2001b) [ZNG01b] also use error backpropagation to optimize their decision behavior. In Zimmermann et al. (2001d) [ZNG01d] the decision making scheme of each agent is modeled with a single neuron. The optimization of the weights is done by standard error backpropagation using *pattern-by-pattern* learning rule. In Zimmermann et al. (2001b) [ZNG01b] the behavior of each agent is modeled with a basic cognitive system [RN95, Bar97, Cra43, p. 31-9 and p. 393-419]. The cognitive systems of the agents are structurally represented by time-delay recurrent neural networks. The networks are trained by a shared weights extension of standard backpropagation using *vario-eta* learning rule (see chp. 5). Note, that both models are presented in chapters 7 and 8 of this thesis.

3. MACROECONOMIC DESIGN ISSUES OF MULTI-AGENT MODELS: MARKETS

In the previous section, we focused on microeconomic design issues of agent-based financial markets. Now, we turn to the macroeconomic side of the market. Major design issues which are related to the macroeconomic market level are depicted in Fig. 6.12 [WHD02, LeB01a, p. 92-3 and p. 256-9].

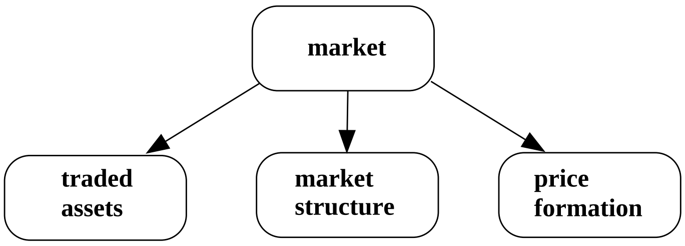


Figure 6.12 Major design issues of macroeconomic market modeling

As shown in Fig. 6.12, macroeconomic design questions can be subdivided into *three* categories: (*i.*) the assets traded on the market, (*ii.*) the structure and organization of the market and (*iii.*) the market price formation mechanism [WHD02, p. 88-93].

The *first* category addresses the traded assets. In the simplest case, a multi-agent model only operates with two assets. For instance, as in Levy (1994) [LLS94] or Lettau (1997) [Let97], the agents may place their funds either in a risky stock or a riskless bond. On the other hand, one may allow the agents to chose from a broad spectrum of investment opportunities. This extension leads to an integrated approach of market modeling in which several markets are treated simultaneously. In this case, an agent holds a well diversified portfolio of assets which may be composed of different stocks, bonds or currencies. An example of multiple assets can be found in the multi-agent model of Zimmermann et al. (2001d) [ZNG01d]. In addition, we point to common types of assets traded in artificial markets and discuss their properties.

The *second* category deals with the structure and organization of the market. First, we focus on the arrangement of the agents in the market by analyzing their communication and interaction structure. In many market models there are no direct relationships between the agents. This means, that there is no direct communication between the agents and even more important, the decisions of a single agent have no direct impact on the behavior of others. The only interaction among the agents is introduced by the market price formation mechanism. – We speak of a *global* communication and interaction structure. In contrast, the model may incorporate a neighborhood structure, which establishes local couplings among the agents. For instance, the agents may share information with their neighbors. More over, the neighborhood may have a direct impact on the agents' decision making behavior. This is referred to as a *local* communication and interaction structure.

Besides the arrangement of the agents, we deal with the issue of trading synchroneity, which means that trading is only permitted at predefined time points (e. g. at the end of each time period). In the majority of agent-based models "trading is synchronized [..., but] in real world there are no fixed periods in which all trades must take place" [LeB01a, p. 259]. We will explain synchronous and asynchronous trading and show, how this topic is addressed in multi-agent literature. Furthermore, we will discuss additional design issues of the market organization like transaction costs or the calibration of the model parameters.

The *third* category describes market price formation mechanisms. As we will show, this design issue is typically approached in one of *three* different ways [WHD02, LeB01a, p. 92-3 and p. 256-7]: Probably the most common way of modeling the price formation mechanism is to assume that the price change is a reaction to the observed market excess (i. e. demand - supply) [Far98, p. 7-11]. Another possibility is "to make several market structure assumptions so that a kind of temporary equilibrium price can be found" [LeB01a, p. 256]. In this case, the market design allows the definition of a demand function, which summarizes the individual demands for the traded asset. Under the assumption that the market supply is constant, a market clearing price can be determined either analytically or computationally [LeB01a, p. 256]. Finally, one may think of real-world trading mechanisms [EG95, p. 25-32]. In other words, "this means actually building a trading mechanism that replicates those used in actual markets, including possible limit orders and order crossing rules" [LeB01a, p. 256-7].

3.1 TRADED ASSETS

The design issue of assets traded in artificial markets can be subdivided into *three* aspects (Fig. 6.11): (*i.*) the number of assets incorporated into the modeling, (*ii.*) the types of the traded assets and (*iii.*) their related properties [WHD02, LeB01a, p. 88-90, p. 92-3 and p. 257]. In the following, we discuss these aspects in greater detail.

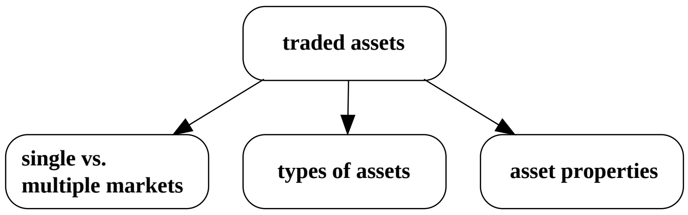


Figure 6.13 Design issues of traded assets

3.1.1 SINGLE VS. MULTIPLE MARKETS

Typically, agent-based models only focus on a *single* financial market in which a *single* asset (e. g. stock or foreign exchange) is traded. Although the agents are only engaged in a single market, they face two investment opportunities: the asset traded in the market and a specific investment alternative. By holding a stake in the traded asset, the agents typically bear a specific risk, which is either due to the market price fluctuations or may result from the uncertainty of the asset payments. As an alternative, the agents have the opportunity to invest their funds into a risk free asset (e. g. bonds or cash) [WHD02, p. 89-90]. This is referred to as a 'single risky asset setup'.

In a such a setup, the agents determine the fraction of their funds which they desire to place in the risky asset. The remaining fraction of their capital is invested into the riskless security. In other words, each agent holds a portfolio of two financial securities, the risky asset and the risk free investment opportunity. The related portfolio optimization task is solved by the agents' decision making process.

In the vast majority of cases, multi-agent models incorporate a single risky asset setup [WHD02, p. 89-90]. A collection of these models is shown in Tab. 6.1.

Table 6.1 Multi-agent models operating with a single risky asset setup

Single risky asset setups		
Multi-agent model	*Type of market*	*Assets*
Levy et al. (1994) [LLS94]	stock market	stock & risk free bond
Lettau (1997) [Let97]	stock market	stock & risk free bond
LeBaron et al. (1997) [LAHP97]	stock market	stock & risk free bond
Joshi et al. (2000) [JPB00]	stock market	stock & risk free bond
Farmer (1998) [Far98]	stock market	stock & cash
Beltratti et al. (1996) [BMT96]	stock market	stock & cash
Lux (1998) [Lux98]	stock market	stock & cash
Lux and Marchesi (1999) [LM99]	stock market	stock & cash
Chan et al. (1999) [CLeLP99]	stock market	stock & cash
Arifovic (1996) [Ari96]	FX-market	two currencies
Zimmermann et al. (2000b) [ZNG00b]	FX-market	two currencies
Zimmermann et al. (2001b) [ZNG01b]	FX-market	two currencies
Chakrabarti (2000) [Chak00]	FX-market	two currencies
Brock and Hommes (1997) [BH97]	security market	security & risk free asset
LeBaron (2001b) [LeB01b]	security market	security & risk free asset

Tab. 6.1 contains a long list of multi-agent models using a single risky asset setup. Most of these models are concerned with stock markets. Here, the agents can either invest their money into a risky stock paying a stochastic dividend or a risk free bond paying a specific interest. As a slight modification, the agents may hold their money in cash earning zero interest. In multi-agent models of foreign exchange markets, the agents allocate their funds between *two* different currencies. The first currency is referred to as the base currency and the second as the counter or quote currency.

A major drawback of single risky asset setups is that the agents have only two investment opportunities [WHD02, p. 89]. This design "is clearly a limitation in thinking about the real world" [LeB01a, p. 257]. Furthermore, "issues such as trading volume, diversification and derivative trading cannot be studied in these single risky-asset setups" [LeB01a, p. 257].

An obvious enhancement of a single risky asset setup is the inclusion of additional securities, which are traded in different markets [WHD02, p. 89-90]. In other words, the agents have at least two risky investment opportunities as alternatives to the risk free asset. The multi-agent model is therefore extended to an integrated approach which treats several financial markets simultaneously. – We speak of a 'multiple risky asset setup'.

An example of a multiple risky asset setup is given in Zimmermann et al. (2001d) [ZNG01d, p. 736-41].[7] The authors introduce a multi-agent approach of multiple FX-market modeling. The agents allocate their funds among N different currencies (e. g. Yen, US-Dollar (USD) and German Mark (DEM)). One of the concerned currencies is treated as a base currency (e. g. DEM). The agents trade in the $N-1$ foreign exchange markets of the counter currencies (e. g. Yen / DEM and USD / DEM FX-markets). The decision making of the agents is based on feedforward neural networks. A single neuron is interpreted as a simple model of economic decision making [ZNG01d, p. 736-7]. The agents are endowed with a separate decision model (neuron) for each FX-market, i. e. a single agent is formed by a group of $N-1$ neurons. Remarkably, the multi-agent model allows the fitting of real-world foreign exchange data [ZNG01d, p. 740-3].

The extension of a single to a multiple risky asset setup truly provides a better adaptation of real-world behavior [WHD02, p. 90]. For instance, a multiple risky asset setup does not ignore inter-market relationships through which all financial markets are now linked together. In addition, such a framework allows to study topics like derivative trading or diversification. Although "adding more securities is certainly desirable, [...] the complexity and computational burden this would bring to the entire market building process is a daunting task" [LeB01a, p. 257].

3.1.2 TYPES OF ASSETS

The types of assets used in multi-agent modeling correspond to those of real-world financial markets. Hence, this section gives an overview of real-world financial securities from the view point of agent-based markets.

[7]The multi-agent model of Zimmermann et al. (2001d) [ZNG01d, p. 736-41] is described in chapter 7.

Looking at real-world financial markets, one can identify a large number of different financial securities. We may restrict the set of financial securities to those that are traded in organized markets [EG95, p. 32-40], because the agents should be able to trade the assets in a centralized location. Furthermore, we may neglect so-called *indirect* investment opportunities,[8] because these types of financial securities are uncommon for agent-based models.

A classification of direct investment opportunities which are traded in organized markets is depicted in Fig. 6.14 [EG95, p. 12].

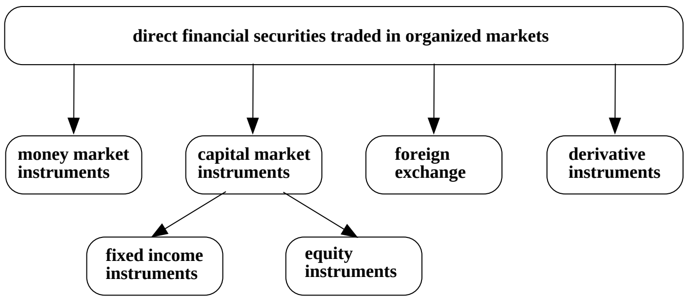


Figure 6.14 Classification of direct investments traded in organized markets.

Let us briefly turn to the different types of direct financial securities which are shown in Fig. 6.14. Referring to the time horizon of the investment, one may distinguish between money and capital market instruments [EG95, p. 12]. Money market instruments have a life of less than one year, while capital market instruments generally have maturities of more than one year [EG95, p. 12]. "Money market securities are short-term debt instruments sold by governments, financial institutions, and corporations" [EG95, p. 12]. Examples of money market securities are treasury bills (T-bills), commercial papers, Eurodollars and repurchase agreements [EG95, p. 13].

Capital market instruments may be subdivided into two categories, debt and equity instruments [EG95, p. 12 and p. 14]. The first category is referred to as the Fixed Income Market. Most of the financial securities in the Fixed Income Market are "traditional bonds and promise to pay specific amounts at specific times" [EG95, p. 14]. Examples of Fixed Income securities are Treasury Notes and Bonds, Federal Agency

[8]Indirect financial instruments are composed of *direct* investments such as stocks or bonds [EG95, p. 12 and p. 18-9]. In other words, an indirect investment can be seen as a (diversified) portfolio of *direct* investments. A mutual fund is a common example of an *indirect* financial instrument.

Securities and corporate bonds [EG95, p. 15-6]. The second category is the Equity Market [EG95, p. 17-8]. Here common stock (equity) is traded. "Common stock represents an ownership claim on the earnings and assets of a corporation" [EG95, p. 17]. Note, that common stock is usually considered as a long-term security, because it does not have a maturity date [EG95, p. 14].

Besides money and capital market instruments, one may also consider currencies as a form of financial securities. Currencies are exchanged for one another on a so-called foreign exchange market (FX-market). In other words, a FX-market is a centralized location, at which a country's currency is expressed in another country's currency. For instance, on the DEM / USD FX-market, German Mark is priced in US-Dollar. In this case, US-Dollar is referred to as the base currency, whereas German Mark is called counter currency. A foreign exchange market represents the interaction of demand and supply, which arises from foreign exchange transactions. The price at which two currencies are traded for another is called the foreign exchange rate (FX-rate). Further details on foreign exchange markets can be found in Mehta (1995) [Meh95, p. 178-86].

Derivative instruments are financial securities whose price is derived from the price of an underlying (primary) financial asset or portfolio of assets [EG95, p. 12 and p. 18]. Since the value of such an instrument is contingent on the value of the underlying asset, derivative instruments are also known as contingent claims. Well-known examples of derivative instruments are futures and options. Further details on the different types of financial securities can be found in Elton and Gruber (1995) [EG95, p. 11-24].

In principle, agent-based models may incorporate all types of financial instruments depicted in Fig. 6.14. – The financial securities shown in Fig. 6.14 are traded in organized markets. Hence, the price of each security results from the interaction of market participants (agents).

However, the majority of agent-based models is limited to financial securities with quite simple characteristics [WHD02, p. 89]. For instance, as outlined in Tab. 6.1, most of the agent-based models deal with simple types of risky stocks and risk free bonds rather than considering e. g. derivative instruments. Or as LeBaron (2001) [LeB01a, p. 257] put it: "The agent based modeling world often is quite simple in the types of traded securities because the complexity introduced through the heterogeneous agents usually pushes the researchers to streamline the assets available for trading." The latter characteristic becomes even more evident, when we describe the properties of the assets traded in artificial markets. This is done in the following section.

3.1.3 ASSET PROPERTIES

Looking at agent-based financial markets, one typically observes that the traded risky asset is often associated with a certain kind of fundamental value. The characteristics of the fundamental value (e. g. dividend, interest or earnings) can be adapted from the real world [LeB01a, p. 257]. The risk free asset is usually available in infinite supply and pays a specific interest r in each time period.

Let us exemplify the properties of the assets traded in agent-based financial markets: In the model of LeBaron (2001b) [LeB01b] the agents allocate their funds among a risk free asset and a risky security. The risk free asset is available in infinite supply and pays a fixed interest r. The risky security is available in a constant supply of 1 share for the whole population of agents. This means, that the share holdings s_i of *all* agents $i = 1, \ldots, N$ sum to 1 at all times. In each time period, the risky security issues a random dividend to the agents.

In the single risky asset setup of Castiglione (2000) [Cast00], the agents can either invest into the traded asset or hold their money in cash. Neither the traded asset nor the cash position pay an interest. The traded asset is connected to a fundamental value. The fundamental value is perceived by value investing agents and is used to incorporate exogenous information into the market. Hence, the fundamental value is "modeled as an exogenous stochastic process with the relative changes drawn from a Gaussian with zero mean and standard deviation σ" [Cast00, p. 867].

This design is similar to Farmer (1998) [Far98]. The agents place their funds either in a risky asset or hold a cash position [Far98, p. 7]. The risky asset does not issue a dividend but is associated with a fundamental value, which is perceived by the value investors in the market. For simplicity, the fluctuations of the fundamental value are computed by a random walk process and given to the value investors in form of an external input signal [Far98, p. 16].

Agent-based models that refer to the CARA utility function concept typically incorporate two securities, a risk free bond and a risky stock. The properties of these securities are as follows [PAHL98, p. 29-30]: The risk free bond pays an interest rate r, while the stock issues a Gaussian distributed dividend d at each time period. The CARA concept is frequently used in multi-agent modeling. Examples are the Santa Fe artificial stock market [PAHL94, PAHL98], Joshi et al. (2000) [JPB00], Yang (1999) [Yan99] or Chen et al. (2001) [ChYe01].

In the single risky asset setups of Lux (1998) [Lux98] and Lux and Marchesi (1999) [LM99] the agents may choose between a risky stock and cash. The cash position pays zero interest. The risky stock is linked to a fundamental value, which is defined as the discounted sum of expected future earnings [LM99, p. 498]. For simplicity, Lux (1998) [Lux98, p. 148] assumes that the fundamental value is constant over time and known to the agents with certainty. In a modified version of the model, changes in the fundamental value are introduced by a Gaussian random variable ϵ with a mean of zero and a time variance of σ_ϵ^2 [LM99, p. 499]. The fluctuations of the fundamental value "constitute the external driving force which affects the market through the operations of the fundamentalist traders" [LM99, p. 499].

The agents in the models of Zimmermann et al. (2000b) [ZNG00b] and Zimmermann et al. (2001b) [ZNG01b] are engaged in a foreign exchange market. The agents choose between two different currencies. Each agent is endowed with two accounts reflecting his holdings in the base and counter currency. The currency positions of the agents pay zero interest. The properties of the FX-market are quite simple: The FX-rate is free floating. This means, that the FX-market is not regularized by a specific FX-market regime (e. g. system of fixed exchange rates). In other words, the FX-rate is solely determined on the basis of the agents' foreign exchange transactions.

Finally, let us consider the multi-agent model of Chan et al. (1999) [CLeLP99]. Similar to other single risky asset setups, the agents either invest into a risky stock or hold their money in cash. The cash position pays zero interest. Remarkably, the risky stock issues a dividend stream D which is different for each investor. According to the authors, this property of the risky asset "is a convenient device for capturing the fact that [an agent] may value a payoff in a particular state of nature more highly than [others]" [CLeLP99, p. 8].

As it can be seen from these examples, multi-agent models refer to stylized properties of real-world financial securities. "Obviously, this yields greater tractability at this very early stage in this field, but extensions will have to be made in the future" [LeB01a, p. 257]. For instance, instead of modeling the fundamental value or incoming market news in form of external stochastic processes, one may think of the inclusion of real-world financial data (e. g. dividend announcements or earnings perspectives). This may not only lead to a better adaptation of real-world behavior, but also to new insights into the functioning of real-world financial markets.

3.2 MARKET STRUCTURE AND ORGANIZATION

This subsection deals with the structure and organization of artificial markets. As shown in Fig. 6.15, the subject can be divided into *three* categories: (*i.*) the arrangement of the agents in the market, (*ii.*) the synchroneity of the trading and (*iii.*) additional market settings like transaction costs or the calibration of the model parameters [Kir96, LeB00, LeB01a, p. 1-4 and p. 695-9 and p. 258-9].

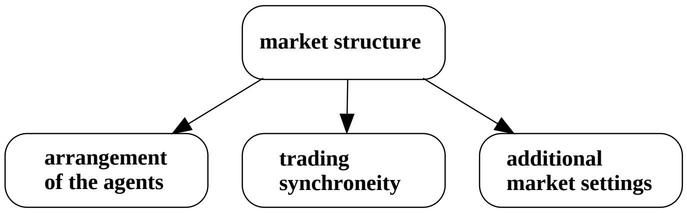


Figure 6.15 Design issues related to the structure and organization of the market

3.2.1 ARRANGEMENT OF THE AGENTS

In a multi-agent model the market dynamics results from the interaction of individuals (agents). The arrangement of the agents in the market, i. e. their communication and interaction structure, is therefore an important design issue for the modeling.

The communication and interaction structure of the market addresses the exchange of information and behavioral dependencies among the agents. For instance, the agents may perceive incoming external information simultaneously (global supply of information). On the other hand, external information may not be spread evenly across the total agent population, i. e. it is locally injected into the market. In this case, the agents may share the information on a private basis. We speak of a news diffusion process.

Concerning the agents' decision making, one may think of agents who decide independently from each other on the micro-level. The only interaction of the agents is due to the (global) price formation mechanism of the market. In contrast, the decision making of an agent may be influenced by a specific subgroup of other agents. This means, that an agent may change his decisions or expectations as a function of the local behavior of other agents.

As it can be seen from these examples, one may distinguish between *local* and *global* communication and interaction structures [Kir96, p. 1-

2]. In the following we explain these types of agent arrangements and give examples for each type.

Local arrangements. We speak of a local arrangement if each agent is coupled to a group of others with whom he communicates or interacts [Kir96, p. 1-2]. The size and the members of such a group are typically defined by a so-called neighborhood.

Neighborhood concepts often refer to a spatial arrangement of the agents. Frequently used neighborhood concepts are based upon chains and lattices [Kir96, p. 2-3]. Suppose, that the agents are arranged in form of a chain, i. e. each agent is represented by a chain link. In this case, adjacent chain links can be used to determine a simple neighborhood structure. Under the additional assumption that the first and the last link of the chain are connected, each agent (chain link) is surrounded by *two* neighbors. A more complicated structure can be generated by including two adjacent chain links on each side into the neighborhood. Hence, each agent is surrounded by *four* neighbors. Similar neighborhood concepts can be defined if the agents are arranged on a lattice [Bor01, p. 671].

As a further extension, one may think of probabilistic couplings between the agents, i. e. the neighborhood structure changes as the market process evolves. For details, alternative neighborhood structures and further readings see e. g. Kirman (1996) [Kir96, p. 2-3].

Within a predefined neighborhood, the agents communicate and interact with each other. For instance, the neighborhood structure can be used by the agents to share information: An agent who has figured out news about the traded asset may spread the information among his neighbors. Incoming news will not affect the total agent population simultaneously but is slowly distributed among the agents [Kir96, p. 2-3].

Such a news diffusion process implies that most of the locally coupled agents have an incomplete information set. Suppose, that new information is only locally injected into the market at varying positions. In this case, the information is only locally available and slowly distributed among the agents in accordance with the predefined neighborhood structure. The agents are restricted to an incomplete information set until the information is provided by their neighborhood.

The speed of the information distribution depends on the definition of the neighborhood structure. In 'sparse' neighborhoods, the news diffusion process is slow, while highly connected agents spread incoming information fast [Kir96, p. 2-3].

Besides the sharing of information, the neighborhood may also have a direct impact on the agents' decision schemes. In other words, the decision making of a single individual can be expressed as a function of neighboring decisions. For instance, an agent may adjust his price expectation to the average price estimate of his neighboring buyers and sellers [Top91, p. 786]. This can also be seen as a process of contagious decision making [Lux95, p. 882-3]. Already mentioned above, contagious decision making is also an important design issue for the agents' decision making processes.

Examples of local agent arrangements are given in Sornette and Johansen (1998) [SJ98] or Bornholdt (2001) [Bor01].

Sornette and Johansen (1998) [SJ98] use a hierarchical structure for the arrangement of the agents. Individual agents are referred to as traders of order $n = 0$. Traders of order $n = 0$ are arranged in groups of m traders. These groups are called traders of order $n = 1$. Traders of order $n = 1$ are also organized in groups of m traders with order $n = 2$. In the resulting hierarchical market structure a group of order n consists of m^n individual traders of order $n = 0$ [SJ98, p. 585 and p. 589]. Within the hierarchical structure, the agents exchange information. Furthermore, the action of a trader influences the traders at the same and lower hierarchy levels [SJ98, p. 585]. Due to a cascade effect, the decisions of the traders at lower hierarchy levels have in turn an impact on traders at higher levels [SJ98, p. 585]. "This is a situation of 'cooperative speculation' in which the information of a buy order issued from one trader is transferred to other traders and, as a result, increases their willingness to enter the market" [SJ98, p. 584].

Bornholdt (2001) [Bor01] uses a simple spin model from the physics of ferromagnetism to model the interaction of the agents. Each agent is represented by a single spin with orientation -1 or $+1$. The spin orientation represents the agents' investment attitude (buyers: 1, sellers: -1) [Bor01, p. 669]. Ferromagnetic couplings connect each agent (spin) to his local neighbors. The couplings cause local interactions among the agents in form of contagious decision making processes [Bor01, p. 669]. The neighborhood structure has the form of a lattice [Bor01, p. 671]. Remarkably, an additional coupling introduces a relationship of each spin (agent) to the global magnetization (market opinion). Since the latter coupling can be anti-ferromagnetic, a minority of agents may be in opposition to the majority opinion of the market [Bor01, p. 669-70].

Global arrangements. In the absence of a specific neighborhood structure, we speak of a global arrangement of the agents in the market. The micro-structure of a global arrangement can be characterized as follows: The agents exist in parallel, i. e. there is no form of local information processing or communication among the agents. Instead of a news diffusion process, newly incoming information is typically perceived by all agents simultaneously. Furthermore, the decisions of a single individual have no direct impact on the decision behavior of others. In other words, the decision making processes of the agents are independent from each other. One may speak of an anonymous market structure without the existence of microeconomic dependencies among the agents. – The agents "take decisions in isolation using only the information received through some general market signals, such as prices, to make their decisions" [Kir96, p. 1].

However, a global arrangement "does not deny that agents interact but, [...] they only do so through the price system" [Kir96, p. 1]. In a global arrangement, the agents submit their buying and selling orders to the market. A market maker collects the orders and computes the resulting market excess (i. e. demand - supply). As a response to the observed market excess, the market maker adjusts the price level and fills the orders of the agents at the newer market price [Far98, p. 7-8]. The interaction of microeconomic buying and selling decisions leads to price changes on the macroeconomic level of the market.

The goal of a global arrangement is to study (*i.*) the behavior of independent agents on the micro-level and (*ii.*) the interaction of their decisions on the macro-level [Far98, p. 4 and p. 15-6]. – The market dynamics in a global arrangement is due to the interaction of independent microeconomic decisions on the macroeconomic level. In contrast, the market dynamics in a local arrangement is created on the microeconomic level by the interaction of neighboring agents.

There are various examples of global agent arrangements. For instance, in the model of Farmer (1998) [Far98] the agents are designed as independent information processing units, which evaluate external data according to a specific trading strategy. The market price dynamics results from the interaction of the independent trading strategies on the macroeconomic level [Far98]. In the FX-market model of Zimmermann et al. (2001d) [ZNG01d] the agents also decide independently from each other. The interaction of the agents is only due to the market price formation mechanism. Further examples of global arrangements can be found in Levy et al. (1994) [LLS94], the Santa-Fe artificial stock market [LAHP97] or Yang (1999) [Yan99].

3.2.2 TRADING SYNCHRONEITY

Another aspect of the market structure is the temporal organization of the trading. We distinguish between synchronous and asynchronous trading schemes [Far98, LeB01a, p. 11 and p. 259].

Synchronous trading means that trading is only permitted at predefined points in time. The following example shows, how a synchronous trading mechanism proceeds: Let us consider a multi-agent model in which the agents either invest into a risky stock or a risk free asset. The market model has a discrete time grid with $t = 1, \ldots, T$ time periods. At the beginning of each time period t, new market information is revealed to the agents. During the time period t, the agents process the information and convert this into trading decisions. At the end of the time period, all agents submit their market orders simultaneously to a market maker. Before the next time period begins, the market maker computes the market excess, adjusts the price level accordingly and fills the orders of the agents. Thereafter, the adjusted market price level is revealed to the agents and the new time period begins. – The sequence repeats [Far98, p. 7-8].

As it can be seen from this example of synchronous trading, the agents submit their orders simultaneously to the market maker and, even more important, trading only takes place *between* two discrete time periods [LeB01a, p. 259]. – In-between time period t and $t + 1$, the market maker adjusts the price level and fills the orders of the agents.

Examples of synchronous trading schemes are given in Levy et al. (1994) [LLS94], Farmer (1998) [Far98], the Santa-Fe artificial stock market [LAHP97] or Zimmermann et al. (2001d) [ZNG01d].

Although synchronous trading schemes can be found in the vast majority of agent-based models, such an organization of the trading is clearly an unrealistic assumption [LeB01a, p. 259]:

> "[I]n the real world there are no fixed periods in which all trades must take place and the price gets set. Agents arrive asynchronously [at the market] and must find others or specified dealers to trade with."

In addition to asynchronous trading schemes, one is also in the need of asynchronous information processes. Typically, new information arrives asynchronously at real-world financial markets. This is in contrast to synchronous information processes which imply that new market data is only revealed at predefined time points.

Asynchronous market processes are only weakly addressed in recent multi-agent literature or remain as unsolved problems. Synchronous trading and information processing schemes can only be seen as approximations of the real world [LeB01a, p. 259]. "Future agent-based financial models will have to better address this issue along with the problem of more realistic trading mechanisms" [LeB01a, p. 259].

3.2.3 ADDITIONAL MARKET SETTINGS

Finally let us discuss two additional aspects of agent-based markets: (*i.*) the integration of transaction costs and (*ii.*) the calibration of the model parameters.

Transaction costs. Transaction costs not only include so-called commission fees or market taxes, but also information and search costs which are associated with the trading. Information costs are the time and effort an agent spends on the assessment of the traded asset. For instance, the agent has to collect and to evaluate information in order to come to a trading decision. Additionally, the agent may bear costs for utilizing sophisticated forecast methods. Search costs are the time and the effort which are necessary to find a trading partner [SvR96, EG95, p. 54-5 and p. 41-2].

While commission fees, market taxes or information costs are important parameters for the agents' decision making, search costs can usually be neglected for the trading of financial securities in organized markets.

For the sake of simplicity, the majority of multi-agent models does not incorporate transaction costs in the form of commission fees or market taxes. However, the inclusion of commission fees and market taxes is a matter of convenience. – One has to add a penalty term to the agents' objective function, which imposes a fee on the planned transactions. Proceeding this way, a profit maximizing agent will only trade if the expected return of the transaction is higher than the expenses of the trading.

Likewise to commission fees or market taxes, information costs are only weakly addressed in recent multi-agent literature. However, in order to push agent-based models towards the direction of the real world, the inclusion of information costs is clearly desirable.

The importance of information costs is illustrated in the model of Beltratti et al. (1996) [BMT96, p. 223-35]. The authors introduce two types of agents, smart and naive ones. Smart agents work with a broad spectrum of data and use sophisticated forecast techniques, while naive agents are restricted to the analysis of most recent market prices and simple forecast tools [BMT96, p. 224-5]. In contrast to naive agents, smart ones have to pay a periodical fee to maintain their superior abilities [BMT96, p. 225-6]. This can be seen as a form of information costs. The agents are able to decide if it is worth to bear the additional costs of the advanced trading strategy [BMT96, p. 226-7]. The authors found that the level of the information costs has a deep impact on the behavior of the agents and the stability of the market. For instance, if the imposed cost level is high, smart trading strategies are superseded by naive ones and the market is destabilized [BMT96, p. 228-35].

Calibration of model parameters. The calibration of the model parameters is one way to connect an artificial market model to the real world [LeB01a, p. 258]. Agent-based models "represent a complicated social situation [(e. g. a market process)] in a highly-stylized fashion" [LeB01a, p. 258]. The implied complexity of the modeling imposes a large number of parameters, which "give the researcher many degrees of freedom from which to align to any interesting empirical feature of the data that one wishes to match" [LeB01a, p. 258]. On the other hand, highly parameterized models are typically difficult to handle and it is often not clear, how the parameters are interrelated.

A first approach to the calibration of the model parameters is rather theoretical: Having constructed an agent-based model, one should think about theoretical parameter settings which most likely push the artificial market "into a well-defined homogeneous agent equilibrium" [LeB01a, p. 258]. The derived parameter settings should be used in a model test run in order to compare the expected and actual outcome of the market process. Remarkably, the latter procedure also helps to gain deeper insights into the model: "[U]nderstanding exactly where the parameter boundaries are between simple and complex behaviors is crucial to understanding the mechanisms that drive agent-based markets" [LeB01a, p. 258]. Examples of such analysis can be found in Lux (1998) [Lux98], Brock and Hommes (1997) [BH97], or Chiarella et al. (2001) [ChHe01].

A more pragmatical approach to the calibration of the model parameters is to adapt the required parameter values from the real world [LeB01a, p. 258]. This way of adjusting the model parameters becomes especially important, if the desired goal of the multi-agent model is the forecasting of real-world financial markets.

An example of the preceding approach is given in Kaizoji (2000) [Kai00]. The author estimates the model parameters on the basis of real-world financial data from the Japanese stock market [Kai00, p. 501-5]. Calibrating the model to the Japanese stock market, the author points out that the Japanese crisis (1987 – 1992) may have its "origin in the collective crowd behavior of many interacting agents" [Kai00, p. 493].

Other examples can be found in Zimmermann et al. (2001d) [ZNG01d] and Zimmermann et al. (2001b) [ZNG01b]. The authors use feedforward and time-delay recurrent neural networks to model the agents' decision making schemes. The model parameters are fitted to the dynamics of the US-Dollar / German Mark FX-market. The agents evaluate real-world financial time series to form their expectations about the future development of the FX-market. Among other input signals, the agents may consider the YEN / USD FX-rate, major stock indices (e. g. Dow Jones Ind., Nikkei 225, DAX), treasury securities (e. g. US-Bonds, German Rex-Index) and commodity prices (e. g. CRB-Price Index). Most remarkably, the multi-agent models not only allow the fitting but also the prediction of the US-Dollar / German Mark FX-rate.

3.3 PRICE FORMATION MECHANISM

Probably the most important macroeconomic design issue of agent-based markets is the modeling of the price formation mechanism, which regulates the trading of the assets. The price formation mechanism relates market orders and prices by transforming the buying and selling decisions of the agents into price changes. As shown in Fig. 6.16, the majority of agent-based models addresses the topic of market price formation in one of three different ways [LeB01a, Far98, WHD02, p. 256-7, p. 6-14 and p. 92-3].

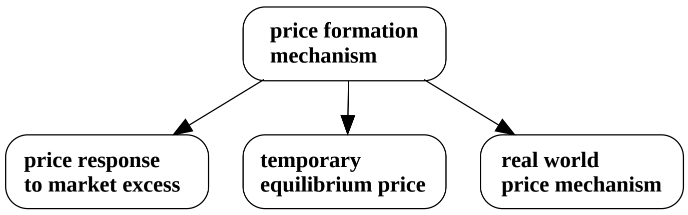

price formation mechanism
price response to market excess
temporary equilibrium price
real world price mechanism

Figure 6.16 Types of market price formation mechanisms

First, one may assume that a superposition of the agents' buying and selling decisions moves the market. Market price adjustments are performed as a response to the observed market excess (i. e. demand - supply). This kind of price formation mechanism is typically defined in terms of a so-called market impact function [Far98, p. 7-9].

Second, one may think of a market structure which allows to define a specific demand function for the agents. Given a fixed market supply, the clearing price can then be derived either analytically or computationally [LeB01a, p. 256].

Third, one may adapt actual trading mechanisms of real-world financial markets [WHD02, LeB01a, p. 92 and p. 256-7]. For instance, one may "simulate a double-auction pricing system which is the actual pricing mechanism in the New York Stock Exchange" [WHD02, p. 92].

3.3.1 PRICE RESPONSE TO EXCESS DEMAND

Probably the simplest way to model the price formation mechanism is to define a so-called market impact function [Far98, p. 7-9].

The market impact function relates the orders of the agents to market price changes. This is done in *three* steps [Far98, p. 7-8]: First, the incoming market orders of the agents are collected. Second, the market excess (i. e. the net of the market orders) is determined. Third, the market price level is adjusted as a response to the observed market excess.

The price response to an observed market excess can be described as follows [LeB01a, Far98, p. 256 and p. 7]: In case of an excess demand (i. e. demand - supply > 0) the market price increases. On the other hand, if an excess supply (i. e. demand - supply < 0) is observed, the price level decreases. The degree of the price reaction is determined by the size of the observed market excess.

For discrete time grids, a simple example of a market impact function can be formulated as

$$p_{t+1} - p_t = \alpha \cdot (D_t - S_t) \ , \tag{6.7}$$

where D_t and S_t are the demand and supply of the agents at time t, α is a positive constant ($\alpha > 0$) and p_t is the actual market price level [LeB01a, p. 256].

In each time period the agents submit their orders to the market. The market excess $D_t - S_t$ is computed by a superposition of the agents' buying and selling decisions. The price shift can be calculated as a response to the observed market excess $D_t - S_t$: If the market orders of the agents match ($D_t - S_t = 0$), no price adjustments are required ($p_{t+1} - p_t = 0$). Otherwise, the price shift $p_{t+1} - p_t$ is estimated by scaling the market excess $D_t - S_t$ with a positive constant α. An excess supply ($D_t - S_t < 0$) leads to a negative price shift $p_{t+1} - p_t < 0$, while an excess demand ($D_t - S_t > 0$) induces a positive price change $p_{t+1} - p_t > 0$ [Far98, p. 7-9].

As it can be seen from Eq. 6.7, the "price change over any given period of time depends only on the net order imbalance during that time" [Far98, p. 9]. Thus, the market impact function (Eq. 6.7) is path independent. This property of the market impact function is in contrast to a nonlinear price formation rule, in which the price may depend on a long sequence of previous net orders. Additional properties of the market impact function (Eq. 6.7) are discussed in Farmer (1998) [Far98, p. 9-11].

The definition of a market impact function (Eq.6.7) comes along with a couple of advantages [LeB01a, Far98, p. 256 and p. 9]: First, this form of market price formation is computationally fast and gives reasonable results. Additionally, "it emphasizes a market continuously in disequilibrium and it allows some amount of analytics depending on the agent structures used" [LeB01a, p. 256].

However, the market impact function (Eq.6.7) has also a few drawbacks [LeB01a, Far98, p. 256 and p. 9-10]: For instance, the price reaction is very sensitive to the value of the parameter α. If α is chosen too large, the price change may overreact to the market excess. Thus, one may observe a highly volatile price series which does not reflect real-world behavior. On the other hand, if α is too small, the price response to an observed market excess is too sluggish. This may lead to flat market price movements which likewise do not reflect the real world. As a further consequence, "the price would slowly rise while investors were never getting the shares they demanded" [LeB01a, p. 256].

A solution to this problem is provided by Zimmermann et al. (2001d) [ZNG01d, p. 738]. The authors propose to determine the parameter α by fitting the observed market excess to real world data.

Another drawback of the price formation mechanism (Eq.6.7) is that the market is assumed to be symmetric. This means, that there is no difference between buying and selling shares. According to Farmer, this assumption may be reasonable "for currency markets and many derivative markets, but probably not for most stock markets" [Far98, p. 10].

Finally, the path independence of the market impact function (i. e. the price shift depends only on the most recent market excess [Far98, p. 9-10]) neglects the existence of other potential price influences. For example, one may think of financial news that have a direct impact on the market price. As a possible solution to this problem, one may add a noise term to Eq. 6.7, which represents the impact of various external factors on the market price [Far98, p. 10].

As a slight extension to the market impact function, one may assume the existence of a market maker [LeB01a, Far98, p. 256 and p. 7-8]. The market maker collects the orders of the agents and computes the resulting market excess. Afterwards, the market maker determines a market price shift as a reaction to the observed market excess and fills the orders of the agents. The price adjustment is performed in accordance with a specific market impact function (e. g. Eq. 6.7).

In contrast to a 'stand-alone' market impact function, the observed market excess is in any case filled by the market maker. According to LeBaron, "[t]his eliminates the problem of asking what happens to agents who place orders that never get satisfied, but opens new questions about the inventory behavior of the market maker" [LeB01a, p. 256].

An example of such a price formation mechanism can be found in Farmer (1998) [Far98, p. 7-11]. Besides a log-linear price formation rule which is similar to Eq. 6.7, a risk neutral market maker is introduced, who fills the orders of the agents. Risk neutral means that the market maker sets the price *only* in response to the observed market excess and does not take into account the risk of his stock inventory. In contrast, "real market makers use their ability to manipulate the price to keep their positions as small as possible" [Far98, p. 9]. If the risk aversion of the market maker should be considered, the market impact function (Eq. 6.7) becomes dependent on the market maker's inventory, which in turn reflects previous and current market orders [Far98, p. 9].

Further examples of market impact functions are given in Zimmermann et al. (2001b) [ZNG01b], Lux (1998) [Lux98], Lux et al. (2001) [LCM01] or the Santa Fe artificial stock market [LAHP97, PAHL94]. Remarkably, the price adjustments in the model of Lux et al. (2001) [LCM01, p. 330-1] are fulfilled by a market maker in a probabilistic manner: The market maker adjusts the price to the next higher (lower) possible value (one cent per share) with a certain probability depending on the observed imbalance between demand and supply.

3.3.2 TEMPORARY EQUILIBRIUM PRICE

The idea of the second price formation mechanism is to find a market clearing price, which results from a temporary market equilibrium. Multi-agent models that use this sort of market price formation incorporate a market structure which allows to define individual demand functions for the agents. The aggregated demand function of the market is then cleared by the identification of an equilibrium price [LeB01a, Far98, p. 256 and p. 11-2]. The calculation of the temporary equilibrium price is either performed analytically, or in case of a complex nonlinear demand function, computationally [LeB01a, p. 256].

In order to clarify, how a market clearing price is calculated in a temporary equilibrium model, let us consider the multi-agent model of LeBaron (2001b) [LeB01b, p. 229-35]: The agents place their funds either in a risk free asset with *infinite* supply or a risky share with *finite* supply of $S = 1$.

The price formation is performed by computing an aggregated demand function D for shares. This function is based on the individual demands of the agents. The aggregated demand function is set equal to the market supply, which is fixed at *one* share ($S = 1$). By solving this equation we find a market clearing price.

Let us assume, that the demand for shares $d_{i,t}$ from agent i at time t is defined as a function of his actual wealth $w_{i,t}$, an information set I_t

and the most recent equilibrium price p_t (i. e. $d_{i,t}(p_t, I_t, w_{i,t})$). The summation of the agents' demand functions leads to the aggregated demand function D of the market. That is

$$D(p_t, I_t, w_{i,t}) = \sum_i d_{i,t}(p_t, I_t, w_{i,t}) \ . \tag{6.8}$$

Noticing, that the share is only supplied in a *finite* amount of $S = 1$, one may set Eq. 6.8 equal to 1, i. e. $D(p_t, I_t, w_t) = 1$. Since the wealth $w_{i,t}$ of each agent i and the set of information are given, one may solve Eq. 6.8 by finding an equilibrium price p_t, such that the aggregated demand is equal to the finite supply $(D(p_t, \bar{I}_t, \bar{w}_t) = 1)$.

The complexity of the agents' demand functions $d_{i,t}$ determines, if a temporary equilibrium price p_t can be found analytically or computationally.

In the model of LeBaron (2001b) [LeB01b], the demand functions $d_{i,t}$ are designed as complex nonlinear functions. Thus, there is no analytic way to find an equilibrium price p_t and the calculations are performed numerically. – To find an equilibrium price p_t, a nonlinear search procedure is applied which starts at the previous equilibrium price p_{t-1} and looks for the first price level that yields to a market excess of zero. Due to the nonlinear design of the demand functions $d_{i,t}$, it is not ensured that an equilibrium price p_t which solves $D(p_t, \bar{I}_t, \bar{w}_t) = 1$ is unique.

It is important to note, that the market equilibrium is only temporary. During the calculations of the market clearing price it is assumed, that the trading strategies underlying the agents' demand functions $d_{i,t}$ are taken as given. When the new equilibrium price p_t is revealed to the agents, some may adapt their trading strategies and thus, also the demand functions $d_{i,t}$ are changed. – "It is in this sense that the equilibrium is only temporary" [LeB01b, p. 234].

Let us finally turn to the advantages and drawbacks of an equilibrium based price formation. According to LeBaron, "this method settles some of the parameter questions of the [market impact function], and can allow for changing market depth, [but] it does require putting more economic structure on the market" [LeB01a, p. 256]. For instance, one is in the need of a well behaved demand function D in order to find a market clearing price. This can be seen as a strong limitation for the market design [Far98, LeB01a, p. 11-2 and p. 256].

Furthermore, the price formation mechanism implies that the market is at least in a temporary equilibrium. This implication may be appropriate for "lower-frequency price dynamics" [LeB01a, p. 256], but may not be suitable for the "representation [... of] continuous trading in high frequency financial markets" [LeB01a, p. 256] (see also Farmer (1998) [Far98, p. 11-2]).

Finally it should be noted, that this price formation mechanism comes along with high computational efforts: "Even in the simplest case where the individual demand functions are linear, this [price mechanism] involves solving a system of simultaneous equations at each time step" [Far98, p. 12]. Compared to the computational efforts and simplicity of a market impact function, this is a major drawback.

Other examples of this market price formation mechanism are given in Arifovic (1996) [Ari96], Brock and Hommes (1997) [BH97], Levy et al. [LLS94] or Chiarella et al. (2001) [ChHe01].

3.3.3 REAL-WORLD PRICE MECHANISM

The third category of market price formation mechanisms is comprised of pricing systems that replicate those of real-world financial markets [WHD02, p. 92-3]. For example, one may think of the inclusion of limit orders, the formation of bid and ask spreads, and order crossing rules. According to LeBaron (2001), this type of market price formation is "most appealing for modeling high-frequency data [..., but] it is only easy to implement when the market clearing mechanism is a mechanical rule" [LeB01a, p. 257]. In case the market clearing is "done through human intervention, as in the New York Stock Exchange specialist system, it requires the modeling of another learning and adapting agent which could get complicated" [LeB01a, p. 257].

Probably one of the most common examples within this category of price formation mechanisms is a (simplified) *double-auction market.* The basic principle of a double-auction market can be characterized as follows [FriR93]: Conventional auctions are *single* sided. This means, that a *single* seller (auctioneer) accepts bids which are submitted from a large number of buyers. In contrast to a single sided auction, a *two sided* or so-called *double-auction* market consists of many sellers *and* many buyers. During a double-auction, the agents may either submit bids (offers to buy) or asks (offers to sell) for the traded asset to the market. An example of a real-world double-auction market is the commodity trading pit at the Chicago Board of Trade.

Examples of simplified double-auction markets can be found in Chan et al. (1999) [CLeLP99], Chakrabarti (2000) [Chak00] and Beltratti et al. (1996) [BMT96].

Chakrabarti (2000) [Chak00] applies the double-auction concept to an artificial inter-bank foreign exchange market. Based on their beliefs and inventory positions, the market participants post bids and asks for the traded currency. The market is decentralized, i. e. dealers meet randomly at the market place to compare their bids and asks. Among other results, Chakrabarti finds that the "intra-day spreads and between-quote returns [of the artificial FX-market] largely conform to the empirically observed intra-day U-shaped pattern" [Chak00, p. 29].

The price formation mechanism in the stock market model of Chan et al. (1999) [CLeLP99] is also a simplified double-auction market. The market is organized as follows: During the trading periods, the agents either submit bids or asks to the market or accept an existing bid or ask. In case of an already existing bid (ask), all subsequent bids (asks) must be higher (lower) than the actual bid (ask) [CLeLP99, p. 8-9].

Each trading period consists of a number of T time intervals [CLeLP99, p. 7]. At the beginning of each time interval, the agents are sorted in a random manner. The resulting list determines the sequence in which the agents are allowed to submit either a single bid or ask to the market. Trading takes place, if a posted bid or ask is accepted by an agent, or if bid and ask cross. In the latter case, the transaction price is determined by the average of the bid and ask [CLeLP99, p. 8-9]. In most of the experiments, the authors limited the total agent population to $N = 20$ individuals. Assuming that a time period is composed of $T = 40$ time intervals, a maximum of $20 \cdot 40 = 800$ transactions may occur during any given trading period [CLeLP99, p. 9].

For simplicity, the agents are only allowed to post bids and asks for a single unit of the traded stock, i. e. the quantity of each bid and ask is fixed at one share per order. Proceeding this way, the authors are able to limit the computational efforts and complexity of the price mechanism, while "retaining the most essential features of a security market, e. g. prices as a medium of information dissemination and aggregation" [CLeLP99, p. 9]. – Nevertheless, the authors consider 'quantity' as an important choice variable. Extended versions of the market model will therefore incorporate bids and asks with variable quantity [CLeLP99, p. 9].

Finally let us comment on the price formation mechanism in the model of Beltratti et al. (1996) [BMT96, p. 179-1], which is similar to a simplified double-auction market. The price formation mechanism is decentralized, i. e. agents meet randomly at the market place to compare their price expectations.

Suppose two agents i and j meet randomly at the market. Instead of comparing bid and ask prices, the agents exchange their expectations $E_{i,t}P_{t+1}$, $E_{j,t}P_{t+1}$ about the stock price. A transaction takes place, if the agents have different price expectations: If the price expectation $E_{i,t}P_{t+1}$ of agent i is *lower* (*higher*) than the price expectation $E_{j,t}P_{t+1}$ of agent

j, agent j buys (sells) one share from (to) agent i. The transaction price is the average of the two expected values $E_{i,t}P_{t+1}$ and $E_{j,t}P_{t+1}$ [BMT96, p. 179-81].

When all agents have completed their transactions by pairs, the overall market price is determined by averaging the individual transaction prices [BMT96, p. 180]. Remarkably, this price formation rule can be seen as a non-cooperative bargaining game between two agents who have to split an expected gain (the difference of the two price expectations). Since a transaction price is determined by averaging the agents' price expectations, the expected gain is equally splitted among the two agents [BMT96, p. 180].

Although the replication of a real-world price formation mechanism is "most appealing for high-frequency data" [LeB01a, p. 257], such a pricing system is often difficult to implement and comes along with high computational efforts and complexity. Therefore, this kind of price formation mechanism is only used for very simple markets that incorporate only a small number of interacting agents. For instance, Chan et al. (1999) [CLeLP99] limit the population size to 20 in most of their experiments.

Furthermore, the structure of the double-auction market is simplified by additional constraints (e. g. limited quantities of bid and ask) [WHD02, p. 92]. However, these constraints can also be seen as an advantage: Due to the simple market design and the small market size, "these markets are carefully controlled experiments [... which] gives the artificial market builder a lot of structure to work with" [LeB01a, p. 257].

4. PRELIMINARY CONCLUSION

In this chapter we provided a detailed overview of recent developments in the field of agent-based financial markets. In a multi-agent framework the market is typically modeled from the bottom up with a large number of interacting agents [WHD02, p. 88-9]. Multi-agent models may therefore be seen as "an exiting new tool for exploring behavior in financial markets that are far from traditional notions of equilibrium, and involve behavior that is less than fully rational at times" [LeB01a, p. 254]. From this point of view, agent-based models are a new approach to build a more realistic theory of financial markets [Far99a, p. 9992].

Instead of assuming market efficiency or rational expectations as it is done in the standard equilibrium theory of financial markets, agent-based approaches enable the researchers to incorporate theories of human behavior from psychology and sociology into the modeling [Far98,

p. 5]. In other words, agent-based financial markets allow to incorporate the behavioral roots of market prices into the modeling: "Prices and returns originate from the buying and selling decisions of economic agents and thus have identifiable behavioral roots which should be accessible to economic reasoning and analysis" [LCM01, p. 328]. This opportunity underscores the potential of multi-agent models as a new approach to real-world financial markets.

As we have shown, agent-based models of financial markets are multifaceted: New researchers entering the area of multi-agent modeling typically face a large number of design questions. Since scientists "from several disciplines including economics, computer science, physics, and psychology, are contributing to this very active field" [LeB01a, p. 254], the answers to the design questions are typically heterogeneous: For instance, one may design the decision making scheme of the agents as a simple trading rule or enrich their behavior by the incorporation of a forecast model [BMT96, p. 174]. In case of rule-based agents, the buying and selling decisions are directly deduced from the trading rule, which often is adapted from real-world trading strategies. Forecasting agents try to predict the future development of the market price. The forecasts are then processed by an additional decision rule or objective function. Within the group of forecasting agents, one distinguishes between agents who rely on econometric forecasting techniques and agents who are modeled by cognitive system based approaches. In the latter case, the decision making of the agents is modeled by semantic specifications of their cognitive processes instead of being limited to the assumption of ad-hoc functional relationships [EM01, p. 759-61]. Other design issues of agent-based financial markets are the structure and organization of the market, the learning and evolution of the agents, the topic of heterogeneous decision making processes, the assets traded on the market, and the market price formation mechanism [WHD02, p. 88-93].

However, it should be noted, that there are design questions which are only weakly addressed or remain as unsolved problems [WHD02, LeB00, PAHL98, p. 122-25, p. 696-9 and p. 31]. Examples of those issues are asynchronous trading and information processing, the calibration of the underlying model parameters or more realistic trading mechanisms. Furthermore, the majority of agent-based financial markets only deals with *qualitative* results of complex economic market phenomena. For instance, multi-agent models enable us to understand stylized facts of financial markets (e. g. fat tailed return distributions, volatility clustering, persistence in volatility and volume) as the outcome of the agents' interaction [Lux98, LM99, p. 146-7 and p. 498]. However, *quantitative* results, i. e. predictions of real-world market prices, are typically *not* supplied [LeB01a, p. 259-60]. A solution of this problem is provided in the next chapters (see chp. 7 and 8). Here, we introduce *two* multi-agent models which enable us to forecast real-world price shifts on the basis of feedforward and time-delay recurrent neural networks [ZNG01d, ZNG01b].

Finally, let us comment on new directions emerging in the research field of agent-based financial markets. Of course, the most obvious direction is towards more realistic models of financial markets. Besides the aspect of forecasting, multi-agent models could be utilized as early warning indicators in order to predict or to analyze upcoming financial crises. Second, agent-based financial markets "may play a very different role in evaluating the stability and efficiency of different trading mechanisms in normal and crisis periods" [LeB01a, p. 260]. According to LeBaron (2000), "[t]his would give more robust policy answers to various current questions about trading mechanics, liquidity and depth, and transparency in all types of markets as new technologies continue to impact their landscape" [LeB01a, p. 260]. Third, it is also imaginable, that research continues in the direction of simpler multi-agent models which are still able to reveal important characteristics of real-world financial markets. Since these models are more comprehensible and easier to analyze than complex multi-agent approaches, this line of research would also support the understanding of markets as complex dynamical systems. The fourth possible direction of research addresses the topic of validation [WHD02, p. 125]. The underlying question is, how we can ensure that the results of an artificial market processes have explanatory power or a connection to real-world behavior. Finally, the research in the area of multi-agent modeling could have a "positive spillover into other areas of economic and social science [...:] These spillovers may lead to new insights about human behavior in many different situations far beyond the boundaries of finance" [LeB01a, p. 260].

Chapter 7

MULTI-AGENT FX-MARKET MODELING BY FEEDFORWARD NEURAL NETWORKS

A market can be seen as a large network of interacting agents [Bar97, p. 813-4]. The agents evaluate external information (e. g. fundamental or technical indicators) to forecast the future development of the market price. The predictions support the agents' decision making. Trading decisions are performed in accordance with a specific objective function (e. g. maximization of the expected profits). Heterogeneity among the population of agents may arise from different personal characteristics [BMT96, p. 143-4 and 174-8].

The agents decide independently from each other on the microeconomic level of the market. The only interaction among the agents is due to the market price formation mechanism [Kir96, p. 1]. On the macroeconomic level of the market, the buying and selling decisions of the agents are collected and the resulting market excess (i. e. the net of the market orders) is computed. The market price level may be adjusted as a response to observed market excess [Far98, p. 7-11]. In other words, the market-price dynamics result from a superposition of the decision-taking processes of the agents.

According to this outline, we believe that multi-agent models provide new insights into the behavior of financial markets: Market prices are explained from the bottom up by the microeconomic decisions of the agents. The interaction of the decisions through the macroeconomic price formation mechanism gives rise to the market dynamics. We may speak of a *price explanation model* on the basis of multi-agent theory [Far99a, p. 9992].

In this chapter, we present a multi-agent model of foreign exchange (FX) markets. The model is based on feedforward neural networks. As we will point out, a single neuron can be seen as an elementary decision-taking model of a single agent [Zim89, Zim94, p. 496-7 and p. 3-6]. Under this interpretation, a neural network of many neurons represents the interaction of many decisions and can thus be seen as a market process [Zim94, Bar97, p. 6-10 and p. 813-4]. Merging the economic theory of the multi-agent market model with neural networks, our models concern semantic specifications instead of being limited to ad-hoc functional relationships [EM01, p. 759-61].

Remarkably, our considerations are *not* restricted to the analysis of single FX-markets. We extend our investigations to an integrated approach, which allows to treat several FX-markets simultaneously. In our example, the agents are engaged in different FX-markets and have to find the most profitable allocation of funds among them. Such a multiple risky asset setup provides a better adaptation to real-world behavior, because it does not ignore inter-market relationships through which all financial markets are now linked together [WHD02, p. 89-90].

In addition, we will discuss two market price paradigms, which focus on the relationship between the micro- and the macroeconomic market level [Far98, p. 6-12].

The chapter is divided into *seven* sections: In section 1., we concentrate on the modeling of the agents. We describe the decision making schemes of the agents and their individual endowments (e. g. trading account, budget constraint) in case of single and multiple FX-markets.

Sections 2. through 5. discuss two market paradigms and their implementation into neural network architectures. The first market paradigm (sec. 2. and 3.) is called 'explicit price dynamics'. It considers the impact of the microeconomic decisions on the macroeconomic price dynamics. In contrast, the second market paradigm (sec. 4. and 5.) deals with the influence of the market on the microeconomic structure. We refer to this market paradigm as the 'implicit price dynamics'. Both market paradigms are modeled with feedforward neural networks. We will discuss network architectures for single and multiple FX-market modeling.

In section 6. we provide experimental results with real-world FX-data (US-Dollar, German Mark and YEN). More precisely, we study the decision behavior of the agents in US-Dollar / German Mark FX-market and point to an optimal degree of heterogeneity [BMT96, WHD02, p. 177-8 and p. 89-92]. In addition, a multiple FX-market analysis indicates, that our models are superior to traditional econometric forecasting techniques (MLP, naive forecasts).

1. MODELING THE BEHAVIOR OF AGENTS IN FX-MARKETS

A foreign exchange (FX) market is a market in which currencies are exchanged for one another, i. e. a country's currency is expressed in terms of another country's currency. FX-markets reflect the interaction of demand and supply, which arises from foreign exchange transactions. The price at which the currencies are exchanged is called the FX-rate.

For instance, on the US-Dollar / German Mark FX-market, US-Dollar (USD) is compared to German Mark (DEM). The FX-rate is either quoted in volume (USD / DEM) or price notation (DEM / USD). Typically, the domestic currency (e. g. DEM) is referred to as the 'base' currency, whereas the foreign currency (e. g. USD) are called 'quote' or 'counter' currency. For further details on FX-markets the reader is referred to Mehta (1995) [Meh95, p. 178-86].

Generally spoken, our multi-agent model consists of a number of agents that trade foreign exchange in a FX-market. The agents decide independently from each other on the microeconomic level. The only interaction among the agents is through the market price formation mechanism [Kir96, p. 1]. We assume that FX-rate changes are solely determined on the basis of the agents' foreign exchange transactions. This means, that the FX-rate is free floating [Meh95, p. 182-84].

The agents evaluate a set of external information, predict the market development and decide trading actions on the basis of their forecasts [BMT96, p. 174]. Each agent is endowed with a system of accounts, which reflect his currency holdings. The decision making schemes of the agents are homogeneous. Heterogeneity among the agent population is introduced by different information sets [BMT96, p. 178].

In the following, we describe the modeling of the agents in case of a single FX-market. We explain the agents' decision making schemes and their individual endowments. In a second step, we enhance the single FX-market framework to an integrated approach, which allows the agents to deal in several FX-markets simultaneously.

1.1 THE BEHAVIOR OF AGENTS IN A SINGLE FX-MARKET

The agents in our FX-market model choose between two different currencies: the base currency (e. g. DEM) and the counter currency (e. g. USD). Each agent is endowed with two accounts reflecting his holdings in the base and the counter currency.

Let us assume, that x_t^a(in USD) is the USD account balance of agent $a = 1, \ldots, A$ in time period t. Correspondingly, y_t^a(in DEM) is the current balance of the agent's DEM account. Both accounts are expressed in terms of the respective currency.

If agent a decides to exchange an amount α_t^a of DEM into USD, this will automatically have an effect on both account balances: While α_t^a units of DEM are subtracted from the DEM account y_t^a(in DEM), an equivalent value of USD is credited to the USD account x_t^a(in USD). The trading behavior of the agent can be modeled by two simple equations, reflecting the upcoming cash balances x^a_{t+1}(in USD) and y^a_{t+1}(in DEM). Recognizing that p_t is the FX-rate in volume notation (USD / DEM FX-rate), we obtain

$$\begin{aligned} y^a_{t+1}(\text{in DEM}) &= y^a_t(\text{in DEM}) + \alpha^a_t \ , \\ x^a_{t+1}(\text{in USD}) &= x^a_t(\text{in USD}) - p_t\alpha^a_t \ . \end{aligned} \tag{7.1}$$

Eq. 7.1 describes the trading behavior of each agent a in a very compact form. If $\alpha^a_t > 0$ ($\alpha^a_t < 0$), the agent decides to buy (sell) the base currency DEM. An equivalent amount $p_t\alpha^a_t$ of the counter currency USD is withdrawn (credited) to the USD account. For simplicity, we neglect transaction costs and assume that the currency holdings pay zero interest.

In each time period t each agent a is only allowed to perform a single trading decision α^a_t. The agent may choose among three alternatives:

1. Purchase the base currency (DEM), which equals $\alpha^a_t > 0$. Note, that buying the base currency (DEM) is equal to selling the counter currency (USD).

2. Sell the base currency (DEM), which equals $\alpha^a_t < 0$. This is equivalent to buying the counter currency (USD).

3. Retain funds invested into the base currency unchanged, which equals $\alpha^a_t = 0$.

To solve this decision problem of the agent, we refer to an elementary decision making process, which can be characterized in *three* stages [Zim89, Zim94, p. 496-7 and p. 3-4]. These stages are:

1. Information filtering resp. selective perception.

2. Forming a superposition of information resp. market evaluation.

3. Performing an action on the basis of the market evaluation.

For a detailed description of the decision making process, let us refer to the situation of an agent dealing in a FX-market [Zim89, Zim94, p. 496-7 and p. 3-4].

Due to the progress in the field of information technology, the agent is overwhelmed by a flood of information about the market. To handle the information overload, the agent has to focus his attention on a couple of news, which, in his opinion, seem to have the deepest impact on the FX-rate.

For instance, if the agent is a chartist, he would stress information about past security prices by using technical market descriptors. Consequently, fundamental indicators, which rely on financial data other than historical quotes, will be neglected by that agent. This procedure of sorting out relevant items of information is the *first* stage of the decision making process.

In the *second* stage, the agent has to analyze the collected pieces of information. By correlating the data, the agent forms his view on the future development of the observed FX-market. This market evaluation supports the third stage of the agent's decision making process.

In the *third* stage, the agent performs an action on the basis of the preceding market evaluation. For instance, the agent may expect the devaluation of the base currency DEM. In this case, a good trading strategy is to buy amounts of the counter currency USD. – Proceeding this way, the agent utilizes expected FX-rate movements to maximize his expected profit.

The outlined decision making scheme of the agent can be modeled by a single neuron [Zim94, p. 4-6]. In other words, the neuron precisely reflects the *three* stages of decision making. The congruence of the decision making process and the mathematical model of a neuron is illustrated in Fig. 7.1.

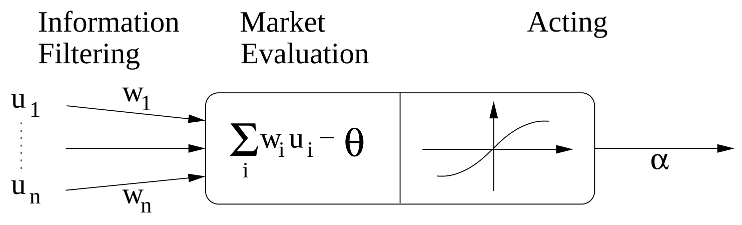


Figure 7.1 A neuron as an elementary model of decision making [Zim94, p. 4]

As shown in Fig. 7.1, the external input signals $u_1, \ldots, u_n$ are weighted by factors $w_1, \ldots, w_n$. By adjusting the weighting factor w_i to a higher value, the agent is able to emphasize the importance of an external influence u_i. Contrarily, non-essential inputs u_j can be ignored by setting their corresponding weighting factors u_j equal to zero. This procedure represents the information filtering phase of the decision making process.

The adder of the neuron reflects the superposition of information: The sum of weighted input signals can be seen as the simplest way of building a superposition of information. The weighting of the input signals refers to their importance, which is expressed by the weights w_i.

In other words, the net input of the neuron can be seen as a quantitative evaluation of the FX-market.

In the next step, the market evaluation is transformed into an action α_t^a. For this purpose, we refer to the bias θ and the nonlinear activation function of the neuron. First, the net input of the neuron is compared to the bias θ. The resulting signal is transferred through the nonlinear activation function of the neuron. The neuron's output α_t^a reflects the trading decision of the agent.

Suppose, that the net input of the neuron, which is lowered by the bias θ, is positive. In this case, the activation function may return a non-negative output ($\alpha_t^a > 0$). This can be interpreted as a decision to buy α_t^a amounts of the base currency DEM. If the lowered net input is negative, the activation function may also return a negative output ($\alpha_t^a < 0$). Correspondingly, the agent will sell α_t^a amounts of the base currency DEM.

It should be noted, that only a nonlinear activation function is able to model the crossover from a market evaluation to an economic decision [Zim94, p. 6].

In mind this interpretation of a single neuron, a neural network which may consist of many neurons can be seen as an interaction model between a large number of decision makers (i. e. a market process) [Bar97, Zim89, Zim94, p. 813-4, p. 497-8 and p. 6-10].

Besides the decision making scheme, we also have to model the agents' objective function, which guides their decision behavior in the investigated FX-market.

Let us assume, that the agents try to maximize their expected profits by anticipating FX-rate changes. For simplicity, we also presume that the agents have identical planning horizons and risk aversion characteristics [WHD02, p. 89-92].

Each agent $a = 1, \ldots, A$ tries to exploit expected changes $\hat{\pi}_{t+1}$ of the FX-rate, such that his expected profit is maximized [Ref95, p. 69-70]:

$$\begin{aligned} &\frac{1}{T}\sum_{t}^{T} \hat{\pi}_{t+1} \cdot \alpha_t^a \rightarrow \max_{\alpha_t^a} \ , \\ &\text{where} \quad \hat{\pi}_{t+1} = E\left(\ln\left(\frac{p_{t+1}}{p_t}\right)\right) \ . \end{aligned} \tag{7.2}$$

According to Eq. 7.2, each agent performs trading actions α_t^a to maximize his expected profit. The trading decisions α_t^a correspond to expected changes $\hat{\pi}_{t+1}$ of the FX-rate. For instance, if the agent expects a positive shift in the USD / DEM FX-rate ($\hat{\pi}_{t+1} > 0$), an obvious strategy is to buy the base currency DEM ($\alpha_t^a > 0$). This trading decision will maximize his expected profit.

Note, that Eq. 7.2 maximizes the average expected profit of the agent across a set of $t = 1, \ldots, T$ trading periods. Remarkably, the objective function described in Eq. 7.2 does not only focus on the correctness of underlying FX-rate forecasts $\hat{\pi}_{t+1}$, but also on the expected profit of each trading decision α_t^a. Eq. 7.2 is often optimized as an error minimization task by turning its sign.

1.2 THE BEHAVIOR OF AGENTS IN MULTIPLE FX-MARKETS

In this subsection, we extend the preceding considerations to an integrate approach that enables the agents to participate in several FX-markets simultaneously.

Let us assume, that the agents may allocate their capital among N different currencies (e. g. USD, DEM and YEN ($N = 3$)). One of these currencies is considered as a base currency (e. g. DEM). The remaining currencies (USD and YEN) are treated as counter currencies, i. e. these currencies are compared to the base currency (DEM). From this it follows, that the agents are only engaged in $N-1$ FX-markets. Within this section the exchange rates between the base currency and the counter currencies are quoted in price notation (e. g. DEM / USD and DEM / YEN FX-rates). This fact is important for the design of the agents' decision making scheme and personal endowments (e. g. accounts and additional budget constraint).

In our multi-agent model of multiple FX-markets, each agent $a = 1, \ldots, A$ spreads his funds among N different currencies. The currency holdings of each agent are reflected by a system of N accounts. For convenience, we assume that the first account is associated with the base currency (DEM), whereas the remaining accounts refer to the counter currencies. Each account is kept in terms of the respective currency.

Let $(x_t^i)_a$ reflect the current amount of currency $i = 1, \ldots, N$ that agent $a = 1, \ldots, A$ holds. Furthermore, $(\alpha_t^i)_a$ is the amount of currency i agent a desires to exchange at time period t. Each trading decision $(\alpha_t^i)_a$ is quoted in terms of the respective currency $i = 1, \ldots, N$. The accounting system of agent a can be written as

$$\begin{pmatrix} x_{t+1}^1 \\ \vdots \\ x_{t+1}^N \end{pmatrix}_a = \begin{pmatrix} x_t^1 \\ \vdots \\ x_t^N \end{pmatrix}_a + \begin{pmatrix} \alpha_t^1 \\ \vdots \\ \alpha_t^N \end{pmatrix}_a . \tag{7.3}$$

The accounting scheme described in Eq. 7.3 is basically the same for all agents $a = 1, \ldots, A$. Although the agents allocate their funds among N different currencies, they are only engaged in $N-1$ FX-markets. The first account refers to the base currency (DEM), while the remaining accounts are associated with the counter currencies (USD, YEN).

The trading behavior of a single agent a can be characterized as follows: The buying and selling decisions for each currency $i = 1, \ldots, N$ are mirrored by a single variable $(\alpha_t^i)_a$. If $(\alpha_t^i)_a$ is positive (negative) the agent desires to buy (sell) amounts of currency i. Likewise to the account administration, each $(\alpha_t^i)_a$ is expressed in terms of the associated currency. For simplicity we assume, that there are *no* transaction costs which impose a fee on the exchange of money. Furthermore, the currency holdings pay no interest.

To prevent the agents from trading unbounded amounts of the currencies, the agents are restricted to a *budget constraint* [WHD02, p. 91]. Recognizing that p_t^i is the FX-rate between currency i and the base currency in price notation (e. g. DEM / USD and DEM / YEN FX-rate), the budget constraint for each agent $a = 1, \ldots, A$ is given by[1]

$$\left(1; \cdots ; p_t^N\right)_a \cdot \begin{pmatrix} \alpha_t^1 \\ \vdots \\ \alpha_t^N \end{pmatrix}_a = 0 \ . \tag{7.4}$$

The mechanism of the budget constraint (Eq. 7.4) can be described as follows: First, we express the desired trading activities $(\alpha_t^i)_a$ of the agent in terms of the base currency. Therefore, each action variable $(\alpha_t^i)_a$ is multiplied by the corresponding FX-rate p_t^i, which is quoted in price notation. Note, that the first component of the price vector is equal to 1, because it reflects the trading decisions $(\alpha_t^1)_a$ for the base currency. Second, we have to guarantee that the sum of all buying decisions $((\alpha_t^i)_a > 0)$ is *equal* to the sum of all sales $((\alpha_t^i)_a < 0)$. Hence, the budget constraint is set equal to 0.

Since the agents must satisfy their budget constraint at all times, they cannot trade unbounded amounts of the currencies. In other words, each buying decision $(\alpha_t^i)_a > 0$ is covered by at least one equivalent selling decision $(\alpha_t^j)_a < 0$, whereby both actions have to be expressed in terms of the base currency.

Now the question arises, how the decision making processes of the agents are modeled. We propose, that each agent evaluates the involved

[1]Remind, that the FX-rate for the *base* currency in terms of the *base* currency is equal to 1 per definition.

FX-markets separately from each other. This means, that an agent has an independent decision making process for each FX-market. Recognizing that N currencies entail $N-1$ FX-markets, an agent is endowed with $N-1$ independent decision making processes. Each decision making process refers to one of the counter currencies.

The different decision making processes have in common, that they are based on the *same* elementary decision scheme. In other words, the underlying structure of the decision processes is homogeneous. In our model we assume that the decision making process for each FX-market is represented by a single neuron [Zim89, Zim94, p. 496-7 and p. 3-4]. Thus, a single agent is formed by a group of $N-1$ neurons (Fig. 7.1).

In accordance with the $N-1$ decision making processes, each agent is also endowed with a set of $N-1$ objective functions. The underlying structure of the objective functions is also identical. We refer to the expected profit maximization task, which is described in Eq. 7.2 [Ref95, p. 69-70].

Each agent tries to benefit from expected changes $\hat{\pi}_{t+1}^i$ in the different FX-rates $i = 1, \ldots, (N-1)$. In other words, the agent's objective is to maximize his expected profits by exploiting expected FX-rate shifts $\hat{\pi}_{t+1}^i$ in the involved FX-markets. The trading decisions $(\alpha_t^i)_a$ of the agent are implicitly connected to each other through the budget constraint (Eq. 7.4).

2. THE EXPLICIT PRICE DYNAMICS OF SINGLE AND MULTIPLE FX-MARKETS

Up to now, the agents participating in the different FX-markets are unable to interact with each other. This means, that the trading activities of agent a have no effect on the buying and selling decisions of other agents. In order to enable the agents to interact, we include a market clearance condition for each FX-market $i = 1, \ldots, (N-1)$, formulated as [Far98, p. 7-8]:

$$\sum_{a=1}^{A} \left(\alpha_t^i\right)_a = 0 \ . \tag{7.5}$$

The market clearance condition (Eq. 7.5) represents the macroeconomic side of each FX-market $i = 1, \ldots, (N-1)$. Let us assume, that the market clearance condition holds for a specific FX-market i. In this case, a single agent can only purchase a desired amount of currency i ($(\alpha_t^i)_{buy} > 0$), if there is at least one agent, who independently decides to sell a (congruent) amount of the currency ($(\alpha_t^i)_{sell} < 0$). Hence, the plans of the agents are indirectly associated with each other through the market clearance condition [Kir96, p. 1].

Since the decision making processes of the agents are independent from each other, it is not clear, if their buying and selling decisions entirely match [WHD02, FJ99, p. 93 and p. 6]: For instance, only a limited trading takes place, if some agents decide to purchase $\sum_{buy} \alpha_t^i = +100$ amounts of currency i and there is only a supply of $\sum_{sell} \alpha_t^i = -60$ units. Agents wishing to purchase currency i can only partly fulfill their plans, because there is a supply shortage of $\sum_{total} \alpha_t^i = -40$ units of currency i. Otherwise, if the volume of planed purchases is smaller than the overall quantity of desired sales, we observe a slack demand on the market.

In other words, "the number of shares [amount of currency] is conserved, i. e., every time an agent buys a share [unit of a currency] another agent loses that share [unit]" [FJ99, p. 6]. Note, that the larger the market the higher is the probability for a single agent to complete his trading activities, due to a plenty of decision overlaps.

The market clearance described in Eq. 7.5 can also be interpreted as an economic equilibrium condition for each FX-market [LeB01a, p. 256]: If a market is in an equilibrium, we have market clearance. This means, that demand and supply for a particular currency are balanced. On the other hand, if we denote a violation of Eq. 7.5, the market is in a disequilibrium state. – We either observe an excess demand or an excess supply.

As a response to the observed market excess (i. e. demand - supply $\neq 0$), the market price is adjusted. This motivates us to define the *explicit market price paradigm*: The observed market excess is the cause of market price changes. In other words, the market price level is directly influenced by the individual buying and selling decisions of the agents. This concept is related to the definition of a so-called market impact function [Far98, p. 7-9].

If demand and supply are balanced on a single FX-market i, there is no need to adjust the related FX-rate. – The buying and selling decisions of the agents match. We can formulate

$$\sum_{a}^{A} \left(\alpha_t^i\right)_a = 0 \quad \Rightarrow \quad \pi_{t+1}^i = 0 \ . \tag{7.6}$$

If the market clearance condition (Eq. 7.5) is violated in one of the involved FX-markets $i = 1, \dots, (N-1)$, adjustments of the related FX-rate are performed. Noticing that c_i, $i = 1, \dots, (N-1)$, is a positive constant ($c_i > 0$), the FX-rate shift required to rebalance the market can be derived by [Far98, p. 7-9]:

$$\sum_{a}^{A} \left(\alpha_t^i\right)_a \neq 0 \quad \Rightarrow \quad \pi_{t+1}^i = c_i \cdot \sum_{a}^{A} \left(\alpha_t^i\right)_a \ . \tag{7.7}$$

Let us consider the market price formation mechanism described in Eq. 7.7: In case of an excess supply ($\sum_{total}(\alpha_t^i)_a < 0$), a negative price shift is required to rebalance the market. Technically, the price mechanism computes this price shift by scaling the negative sum of the market orders with a positive constant c_i. The resulting price shift π_{t+1}^i is always smaller than zero. If we observe an excess demand, i. e. the volume of desired sales does not supply the amount of planned purchases ($\sum_{total}(\alpha_t^i)_a > 0$), a positive price shift is needed to rebalance the market. The required price shift results from multiplying the observed excess demand by the positive constant $c_i > 0$.

Note, that the degree of the price reaction depends on the size of the observed market excess. Furthermore, the price formation mechanism (Eq. 7.7) implies that each FX-rate $i = 1, \ldots, (N-1)$ is solely determined on the basis of the agents' FX-transactions. The means, that the FX-rates are free floating [Meh95, p. 182-4].

3. MODELING THE EXPLICIT PRICE DYNAMICS OF SINGLE AND MULTIPLE FX-MARKETS

In this section, we show how the explicit price dynamics can be embedded into a feedforward neural network architecture. For simplicity, let us start off with the analysis of a single FX-market. Thereafter, we expand our considerations to multiple FX-markets.

3.1 MODELING A SINGLE FX-MARKET

The neural network architecture of Fig. 7.2 depicts the explicit price dynamics in case of a *single* FX-market.

Let us explain the multi-agent FX-market model depicted in Fig. 7.2: The input cluster contains different preprocessed financial time series. The preprocessing of the input data is basically done by calculating the *momentum* (Eq. 2.7) and *force* (Eq. 2.8) of each raw input time series [NZ98, p. 374-5]. By this, we are able to capture the dynamics of the raw input time series. Note, that both preprocessing functions include an additional scaling of the transformed input data [PDP00, p. 73].

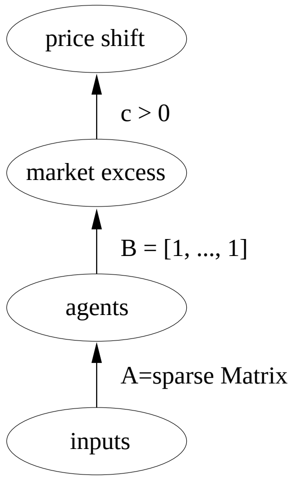


Figure 7.2 Modeling a single FX-market by a feedforward neural network. FX-rate shifts are performed as a reaction to the observed market excess. Due to visual clarity, the bias weights θ_i^a to the 'agents' cluster are suppressed.

The 'agents' layer consists of a number of A neurons. Each neuron represents the decision making process of a single agent $a = 1, \ldots, A$ [Zim89, Zim94, p. 496-7 and p. 3-4]. The neurons are equipped with tanh(.) squashing functions [Zim94, p. 5-6].

The behavior of the agents is guided by an expected profit maximization task. Therefore, the 'agents' layer is designed as an output cluster. Each neuron is equipped with a *prof-max* error function (Eq. 7.2) [Ref95, p. 69-70]. We use the FX-rate shifts π_{t+1} as target values.

Matrix A is a sparse connector, i. e. most of the weights are equal to zero. This construction reflects the information filtering phase of the agents' decision making processes [Zim94, p. 3-4]. The sparseness of matrix A generates heterogeneous decision behavior, because the agents rely on different information sets. Heterogeneity means that the behavior and decision making of the agents differs in a given market situation. Generally, heterogeneous decision making is required to adapt the underlying market dynamics [BMT96, WHD02, p. 178 and p. 89-92]. Note, that we initialize matrix A at random.

The calculation of the market excess (i. e. the net of the agents' market orders), is done by the fixed matrix $B = [1, ..., 1]$. Accordingly, the 'market excess' cluster is designed as a hidden layer with an identity activation function.

The FX-rate shift is a response to the observed market excess. We estimate the FX-rate shift by scaling the market excess with a proportional constant $c > 0$. The 'price shift' layer is designed as an output cluster. We use the FX-rate shifts π_{t+1} as target values. The output layer is equipped with the robust error function $\ln \cosh$ (Eq. 5.9) [NZ98, p. 387-88].

The network is trained by standard error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 173-75 and p. 395-99].

3.2 MODELING MULTIPLE FX-MARKETS

Next, we introduce a feedforward neural network, which serves our purpose of modeling the explicit price dynamics of multiple FX-markets. The network architecture is depicted in Fig. 7.3.

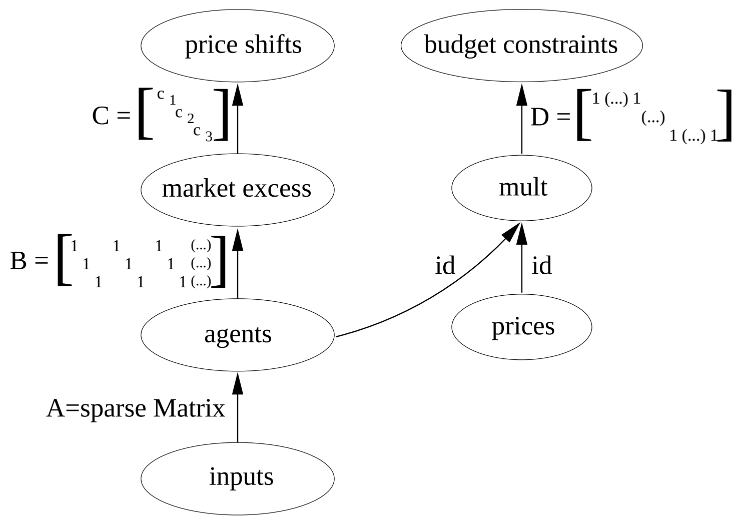


Figure 7.3 Modeling the explicit price dynamics of multiple FX-markets by a feedforward neural network. Due to visual clarity, the bias weights θ_i connected to the 'agents' cluster are suppressed.

As shown in Fig. 7.3, the neural network architecture of the multi-agent model consists of *two* combined branches. The *first* branch, on the left hand side, is used to model the price dynamics of the investigated FX-markets $i = 1, \dots, (N-1)$. The *second* branch, on the right hand side, refers to the budget constraints of the agents (Eq. 7.4).

Let us describe the first branch of the network architecture (Fig. 7.3): The input layer provides the network with different financial input signals. In the preprocessing, we calculated the scaled momentum and force of each raw input signal [NZ98, p. 374-5].

The 'agents' layer contains the decision making processes of the agents. A single agent $a = 1, \ldots, A$ is formed by a group of $N-1$ neurons. The neurons reflect the agent's decision making processes for the $N-1$ FX-markets [Zim89, Zim94, p. 496-7 and p. 3-4]. Each neuron is equipped with a tanh(.) activation function [Zim94, p. 5-6]. Remind, that each agent evaluates the involved $N-1$ FX-markets separately from each other by using independent decision making processes and objective functions.

The $N-1$ neurons of a single agent a are arranged in a sequence, i. e. they are located one after the other in the 'agents' layer. For instance, neurons 1 to $N-1$ constitute the first agent ($a = 1$), whereas the second one ($a = 2$) is composed of neurons N to $2(N-1)$.

The 'agents' layer is designed as an output cluster. The agents' decision behavior is guided by an expected profit maximization task (Eq. 7.2). Each neuron of an agent a is therefore equipped with the *prof-max* error function described in Eq. 7.2 [Ref95, p. 69-70]. We apply the different FX-rate shifts π^i_{t+1}, $i = 1, \ldots, (N-1)$, as target values.

The sparse matrix A refers to the information filtering concept [Zim94, p. 3-4]. We spread the information among the agents such that their decision behavior is heterogeneous [BMT96, WHD02, p. 178 and p. 89-92]. Each decision making process of an agent is associated with a different data subset. Due to the information filtering, each data subset contains only a small number of inputs on the average. In our experiments, the sparseness of matrix A is initialized by random.

The price formation is performed for each FX-market $i = 1, \ldots, (N-1)$ separately. First, we collect the agents' trading decisions on each FX-market and calculate the resulting market excess. This is done by the fixed matrix B. The block structure of matrix B ensures, that the market excess is computed for each FX-market separately. According to the composition of the 'agents' layer, B is a fixed matrix consisting of $N-1$ rows and $a \cdot (N-1)$ columns. The 'market excess' cluster is designed as a hidden layer. More precisely, the 'market excess' layer consists of $N-1$ hidden neurons, which are equipped with identity activation functions. Each neuron calculates the market excess of a single FX-market.

The FX-rate shift π^i_{t+1} of each FX-market $i = 1, \ldots, (N-1)$ is estimated by scaling the respective market excess with a positive constant $c_i > 0$. The constants c_i, $i = 1, \ldots, (N-1)$, are arranged in the diagonal matrix C. The 'price shifts' layer is an output cluster, which is endowed with the robust error function $\ln\cosh$ (Eq. 5.9). We use the different FX-rate shifts π^i_{t+1}, $i = 1, \ldots, (N-1)$, as target values.

The *second* branch of the network architecture (Fig. 7.3) reflects the budget constraints of the agents (Eq. 7.4): The 'mult' layer computes the products $p_t^i \cdot (\alpha_t^i)_a$ across all agents and FX-markets. Note, that the *id* connectors are fixed identity matrices.

Thereafter, the budget constraints are computed by summarizing the products $p_t^i \cdot (\alpha_t^i)_a$ for each agent $a = 1, \ldots, A$ separately. This is done by the fixed block matrix D. The 'budget constraints' layer is designed as an output cluster, which contains a number of A output neurons. Each output neuron refers to the budget constraint of a single agent. The neurons have task invariant target values of 0. We propose to apply the robust error function $\ln \cosh$ (Eq. 5.9) to the 'budget constraints' layer.

The multi-agent model (Fig. 7.3) is trained by standard error back-propagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 173-75 and p. 395-99].

4. THE IMPLICIT PRICE DYNAMICS OF SINGLE AND MULTIPLE FX-MARKETS

The implicit market paradigm claims, that the price shift is a control parameter of the market to avoid situations of a market disequilibrium. According to this paradigm, the market has a direct impact on the decision behavior of the agents.

The resulting price formation mechanism is equivalent to the market process constructed by *Walras*. Walras described the process by which a market equilibrium is established as tatonnement or 'grouping'. By this, Walras emphasized that equilibrium prices and quantities were established through repeated experiments before trading takes place [Weid00, Pat87, p. 200 and p. 864-8].

Recall, that the agents' trading activities $(\alpha_t^i)_a$ result from their decision making processes. In each market $i = 1, \ldots, (N-1)$, the decision making of an agent can be expressed as a function of miscellaneous external inputs u_t^j. Hence, we can also formulate the agent's decisions in terms of the external input signals: $(\alpha_t^i)_a = f(u_t^j)$, $\forall i = 1, \ldots, (N-1)$.

Now suppose that the price shift π_{t+1}^i is a control parameter for each market $i = 1, \ldots, (N-1)$ to avoid situations of market disequilibria. As a consequence, we have to modify the agents' decision making processes. Their decision behavior on a single market $i = 1, \ldots, (N-1)$ becomes a function of both, external inputs u_t^j *and* the market price shift π_{t+1}^i:

$$(\alpha_t^i)_a = f(u_t^j, \pi_{t+1}^i) , \quad \forall i = 1, \ldots, (N-1) . \tag{7.8}$$

The relationship described in Eq. 7.8 has also an impact on the price formation mechanism of each FX-market. In case of a single FX-market $i = 1, \ldots, (N-1)$, the market excess $ex_t^i(.)$ is now given by

$$ex_t^i(u_t^j, \pi_{t+1}^i) = \sum_a^A \left(\alpha_t^i(u_t^j, \pi_{t+1}^i)\right)_a = 0 \ ,$$
$$\text{with} \quad \frac{\partial ex_t^i(.)}{\partial \pi_{t+1}^i} < 0 \ . \tag{7.9}$$

Eq. 7.9 reflects the implicit price dynamics of each FX-market $i = 1, \ldots, (N-1)$. The price shift π_{t+1}^i can be interpreted as a control parameter of each market $i = 1, \ldots, (N-1)$ to avoid situations of market disequilibria. In other words, the macroeconomic side of each market affects the decision making processes of the agents.

The partial derivative of the market excess $ex_t^i(.)$ with respect to the price shift π_{t+1}^i incorporates a stability condition into each market $i = 1, \ldots, (N-1)$ (see Eq. 7.9). The behavior of the partial derivative implies that the market excess becomes negative (positive) if the price level is higher (lower) than the equilibrium price. "Since buyers tend to raise their bids if [market excess] is positive and sellers to lower their prices if [market excess] is negative, this behavior drives the deviating price back to its equilibrium value" [Weid00, p. 200].

To find an equilibrium price on each market $i = 1, \ldots, (N-1)$, we have to compute a price shift π_{t+1}^i such that the related market excess $ex_t^i(\pi_{t+1}^i)$ is equal to zero (i. e. $ex_t^i(\pi_{t+1}^i) = 0$). This can be done by calculating a fix-point π^i for each market $i = 1, \ldots, (N-1)$ separately.

In the following, we explain the fix-point calculation in case of a single market. Let us assume that the actual market excess is equal to zero, i. e. $ex_t^i(\pi_{t+1}^i) = 0$. For convenience let us simplify this term by writing $f(x) = 0$. One may characterize the solution of the latter equation by the fix-point x of an iterative series. The formation law of the iterative series $\{x^i\}_{i=1,\ldots}$ can be written as

$$x^{i+1} = x^i + \epsilon \cdot f(x^i). \tag{7.10}$$

Considering the function $f(x)$ and its *bounded* derivative $f'(x) < 0$, the related iterative series $\{x^i\}_{i=1,\ldots}$ in Eq. 7.10 converges to a fix-point x, if the constant ϵ is chosen on the constraints to be positive and sufficiently small ($\epsilon > 0$).

In order to proof the convergence properties of Eq. 7.10 for the function $f(x)$, we focus on the distance between the elements of the iterative series $\{x^i\}_{i=1,\dots}$ and the related fix-point x: This distance can be computed by the difference $(x^{i+1} - x)$.

Equivalent to $f(x) = 0$, we can rewrite

$$x = x + \epsilon \cdot f(x) \ . \tag{7.11}$$

Now, the difference $(x^{i+1}-x)$ can be computed by subtracting Eq. 7.11 from Eq. 7.10. Rearranging terms, we obtain

$$(x^{i+1} - x) = (x^i - x) + \epsilon \cdot \big(f(x^i) - f(x)\big) \ . \tag{7.12}$$

Expanding the difference $\epsilon \cdot \big(f(x^i) - f(x)\big)$ in Eq. 7.12 by $(x^i - x)$ leads to

$$(x^{i+1} - x) = (x^i - x) + \epsilon \cdot \frac{f(x^i) - f(x)}{(x^i - x)} \cdot (x^i - x) \ . \tag{7.13}$$

Next, by factoring out $(x^i - x)$ and employing the *mean value theorem*,[2] we obtain

$$\|x^{i+1} - x\| \leq \|1 + \epsilon \cdot f'(\tilde{x}^i)\| \cdot \|x^i - x\| \ . \tag{7.14}$$

Since the derivative $f'(\tilde{x}^i)$ is smaller than zero and bounded, the iterative series $\{x^i\}_{i=1,\dots}$ will converge to the fix-point x in case of a sufficiently small $\epsilon > 0$.

The latter contraction is due to the fact that $\|1 + \epsilon \cdot f'(\tilde{x}^i)\|$ is below 1, if $\epsilon > 0$ and $f'(\tilde{x}^i) < 0$. The resulting iteration loop will converge to a fix-point x, because the distance $\|x^{i+1} - x\|$ is equal or less than the distance $\|x^i - x\|$ multiplied by $\|1 + \epsilon \cdot f'(\tilde{x}^i)\| < 1$. This means, that the series $\{x^i\}_{i=1,\dots}$ is a contraction. By solving the iterative series (Eq. 7.10), we get a fix-point x which by Eq. 7.11 is the solution of $f(x) = 0$.

The application of the preceding iteration scheme allows us to calculate an equilibrium price shift π^i_{t+1} for each FX-market $i = 1, \dots, (N-1)$ separately.

[2]Let f(x) be defined and continuous on an interval $[a, b]$. Now, the mean value theorem states, that there is at least one point c in $[a, b]$ such that $f'(c) = \frac{f(b)-f(a)}{b-a}$.

5. MODELING THE IMPLICIT PRICE DYNAMICS OF SINGLE AND MULTIPLE FX-MARKETS

In this section, we introduce a multi-agent approach to the implicit market price dynamics. The resulting market model is based on a fix-point recurrent neural network. First, we deal with the analysis of a single FX-market. Thereafter, we expand our considerations to the modeling of multiple FX-markets.

5.1 MODELING A SINGLE FX-MARKET

Fig. 7.4 and 7.5 depict the training and testing phase of a fix-point recurrent neural network, that captures the implicit price dynamics of a single FX-market.

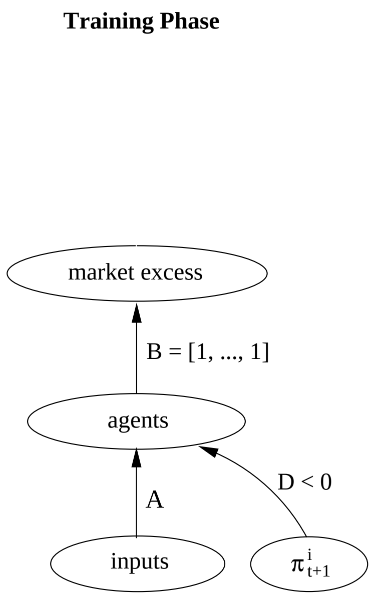


Figure 7.4 Neural network capturing the implicit price dynamics: Training.

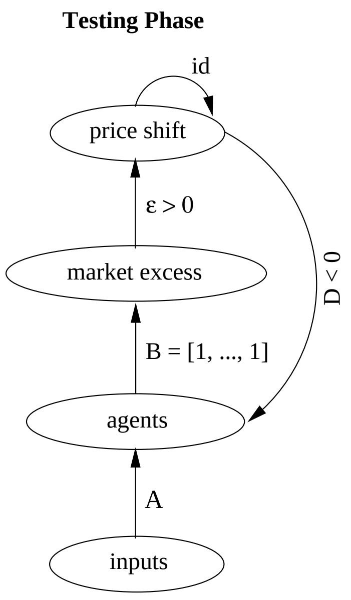


Figure 7.5 Neural network capturing the implicit price dynamics: Testing.

The training phase (Fig. 7.4) is basically done by a feedforward neural network. The input layer consists of a number of external input signals u^j_t. In the preprocessing, we calculated the scaled momentum and force of each raw input signal [NZ98, p. 374-5].

The 'agents' layer contains the decision making processes of the agents $a = 1, \dots, A$. The decision making process of each agent is a function of the external inputs u_t^j *and* the future FX-rate shift π_{t+1}. Remind, that the upcoming price shift is a control parameter of the market to avoid market disequilibria.

Matrix D provides the agents with information about the future FX-rate shift π_{t+1}. Since we have to guarantee that the derivative of the market is negative (Eq. 7.9), we only allow negative weights $d_i < 0$ in D. Matrix A is sparse. By this, we adapt the information filtering concept [Zim94, p. 3-4] and generate heterogeneous behavior [BMT96, p. 178].

The 'agents' layer is designed as an output cluster. The agents maximize their expected profits in accordance with the *prof-max* error function described in Eq. 7.2 [Ref95, p. 69-70]. We use the FX-rate shifts π_{t+1} as target values. Each agent (neuron) is endowed with a tanh(.) activation function [Zim94, p. 5-6].

The market excess is computed by the fixed matrix B, which collects the market orders of the agents. As a specialty, the 'market excess' layer consists of a single output neuron. The neuron has task invariant target values of 0. This design refers to the implicit market price paradigm: The agents optimize their expected profit, while the market adjusts the FX-rate to vanish an observed market excess. We use the robust error function $\ln\cosh$ (Eq. 5.9) to generate error signals for the 'market excess' layer.

The network training is done by error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 173-75 and p. 395-99].

In the testing phase of the multi-agent model we have to omit the future information π_{t+1}. Instead, we determine the FX-rate shift π_{t+1} by the implicit condition of Eq. 7.9. The resulting fix-point neural network (Fig. 7.5) solves the implicit condition by iterating the series π^n (Eq. 7.10).

The weights of the connections A and D are recaptured from the training phase, whereas $\epsilon > 0$ is initialized sufficiently small to guarantee convergence (Eq. 7.14). Matrix D with $d_i < 0$ enables the agents to react properly on expected FX-rate shifts. Due to the recurrent formulation, matrix D delays the FX-rate shift π_{t+1} for one time step.

5.2 MODELING MULTIPLE FX-MARKETS

Next, we introduce a fix-point recurrent network architecture to model the implicit price dynamics of multiple FX-markets. The model design is different for training and testing. The training phase is done by a feedforward neural network (Fig. 7.6), while we use a fix-point recurrent network during the testing (Fig. 7.7).

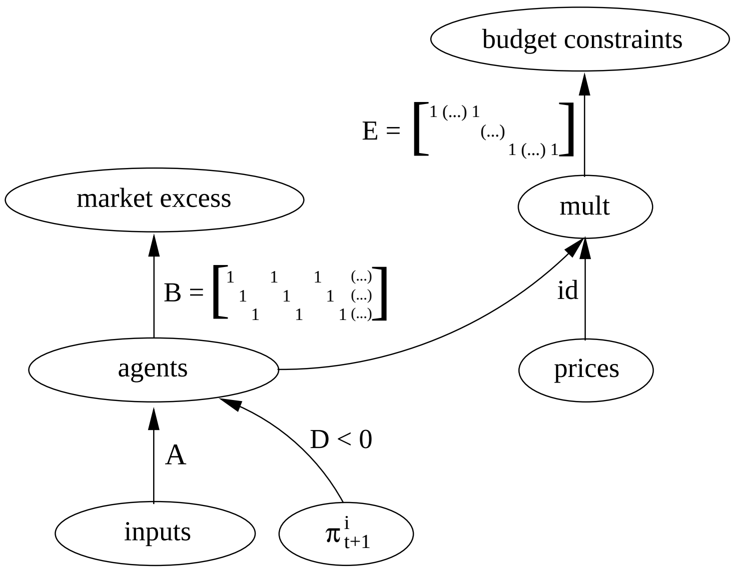

budget constraints
E = [1 (...) 1; (...); 1 (...) 1]
mult
market excess
B = [1 1 1 (...); 1 1 1 (...); 1 1 1 (...)]
id
agents
prices
D < 0
A
inputs
π^i_{t+1}

Figure 7.6 Training phase of the neural network modeling the implicit price dynamics of multiple FX-markets. Due to visual clarity, the vector of bias weights θ connected to the 'agents' cluster is suppressed.

During the training (Fig. 7.6), the agents maximize their expected profits on the different FX-markets $i = 1, \ldots, (N-1)$. Recall, that each agent evaluates the involved $N-1$ FX-markets separately from each other. Thus, a single agent is composed by a group of $N-1$ neurons. The neurons are sequentially arranged in the 'agents' layer. Each neuron is equipped with a tanh(.) squashing function [Zim94, p. 5-6].

The 'agents' layer is designed as an output cluster. We apply the *prof-max* error function (Eq. 7.2) to model the expect profit maximization tasks of the agents. The different FX-rate shifts π^i_{t+1}, $i = 1, \ldots, (N-1)$, serve as target values.

The agents' decision behavior on each FX-market $i = 1, \ldots, (N-1)$ is influenced by a set of external influences u^j_t *and* the future FX-rate shift π^i_{t+1}. – While the agents try to maximize their expected profits, FX-rate adjustments π^i_{t+1} are performed on each market $i = 1, \ldots, (N-1)$ to vanish market disequilibria.

The agents are provided with information about upcoming FX-rate shifts π^i_{t+1}, $i = 1, \ldots, (N-1)$, by connecting the 'π^i_{t+1}' input layer to the 'agents' layer using matrix D. As a stability condition for each FX-market $i = 1, \ldots, (N-1)$, the partial derivative of the market excess $ex^i_t(.)$ with respect to the FX-rate shift π^i_{t+1} has to be negative (Eq. 7.9)

[Weid00, p. 200]. Hence, we only allow negative weights $d_i < 0$ in matrix D. The sparse matrix A refers to the information filtering concept [Zim94, p. 3-4] and is used to spread the information among the agents [BMT96, p. 178]. Matrix A is initialized by random.

The market excess is calculated for each FX-market $i = 1, \ldots, (N-1)$ separately. This is done by the fixed block matrix B. The design of matrix B corresponds to the arrangement of the agents' decision making processes in the 'agents' layer.

Likewise to the single FX-market setup (Fig. 7.4), the 'market excess' layer is designed as an output cluster. The 'market excess' layer incorporates $N-1$ output neurons with task invariant target values of 0. In case of a market excess in one of the involved FX-markets $i = 1, \ldots, (N-1)$, an error flow is induced by the robust cost function $\ln\cosh$ (Eq. 5.9).

During the training, we have to prevent the agents from trading unbounded amounts of the N currencies [WHD02, p. 91]. The agents' budget constraints are modeled by the additional network branch (Fig. 7.6). Note, that this part of the network architecture corresponds to Fig. 7.3.

The training of the feedforward neural network (Fig. 7.6) is done by error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 173-75 and p. 395-99].

The testing phase of our multi-agent model is depicted in Fig. 7.7. We compute the implicit price dynamics of the involved FX-markets $i = 1, \ldots, (N-1)$ by a fix-point recurrent neural network architecture.

During the recall of the neural network (Fig. 7.6), we have to omit the usage of future information π^i_{t+1} with $i = 1, \ldots, (N-1)$. Instead, we determine the required FX-rate shift π^i_{t+1} for each FX-market $i = 1, \ldots, (N-1)$ by the implicit condition of Eq. 7.9. Each FX-rate shift π^i_{t+1} is estimated by a recurrent iterative loop, whereby all other parameters are held constant.

More precisely, the implicit condition (Eq. 7.9) is solved by the fix-point recurrent neural network, which iterates the series of Eq. 7.10 for each FX-market $i = 1, \ldots, (N-1)$ separately.

The matrices A and D are recaptured from the training phase, whereas the diagonal elements of matrix C, $\epsilon_i > 0$, are initialized sufficiently small to guarantee convergence of each fix-point recurrence (Eq. 7.14).

6. EMPIRICAL STUDY: MULTI-AGENT MODELING OF FX-MARKETS

In this section, we apply our multi-agent models to real-world FX-markets (German Mark (DEM), US-Dollar (USD) and Japanese YEN). Throughout the empirical study we assume that DEM is the base cur-

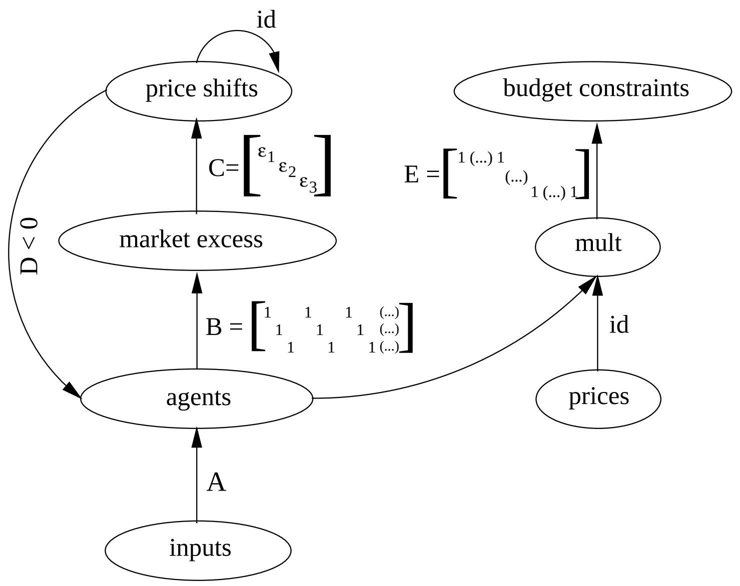


Figure 7.7 Testing phase of the neural network capturing the implicit price dynamics of multiple markets. Due to visual clarity, the bias weights θ_i connected to the 'agents' layer are suppressed.

rency, whereas USD and YEN are treated as counter currencies. Thus, we model the USD / DEM and YEN / DEM FX-markets.

This section is composed of two parts: First, we introduce the outline of the empirical study. Thereafter, we present the empirical results modeling the USD / DEM and YEN / DEM FX-markets.

6.1 OUTLINE OF THE EMPIRICAL STUDY

The basics of our empirical study can be characterized as follows: We are working on the basis of daily data from Jan. 91 to Dec. 97 (1746 data points) to forecast the daily price shifts in the USD / DEM and YEN / DEM FX-markets. The data set is divided into *two* subsets: *(i.)* the training set, which covers the time period from Jan. 91 to Aug. 96 (1479 data points), and *(ii.)* the generalization set, which covers the time from Oct. 96 to Dec. 97 (267 data points).

As a further decision, we have to assemble a database, which serves as raw input data for the agents. The selection of input data was guided by the idea, that every major currency is at least driven by *four* influences. These are the development of *(i.)* other major currencies, *(ii.)*

the national and international bond, *(iii.)* stock and *(iv.)* commodity markets [Mur91].

According to these suggestions, we composed a data set of 11 indicators, based on technical and fundamental data: Besides the USD / DEM and YEN / DEM FX-rates, the agents may consider the YEN / USD FX-rate, major stock indices (e. g. Dow Jones Ind., Nikkei 225, DAX), treasury securities (e. g. US-Bonds, German Rex-Index) and commodity prices (e. g. CRB-Price Index).

In the preprocessing, we calculated the scaled momentum and force of each input time series [NZ98, p. 374-5]. The resulting signals are used as inputs to our multi-agent models. We trained our models by standard error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 161-68, p. 173-75 and p. 395-99].

Our empirical study consists of two parts: First, we deal with the analysis of a single FX-Market. We apply the multi-agent models depicted in Fig. 7.2 and 7.5 to the USD / DEM FX-market. Second, we deal with multiple FX-markets. More precisely, we apply the multi-agent models depicted in Fig. 7.3 and 7.7 to the USD / DEM and the YEN / DEM FX-markets.

6.1.1 OUTLINE OF THE SINGLE FX-MARKET ANALYSIS

The goal of the single FX-market analysis is to study the behavior of the agents in terms of heterogeneity [BMT96, WHD02, p. 177-8 and p. 89-92]. For this purpose, we compare the average size of the agents' input data set to (*i.*) the accumulated return of the multi-agent models (Fig. 7.8) and (*ii.*) the observed market excess (Fig. 7.9).

These experiments focus on the degree of heterogeneity among the agents, which is required to capture the market dynamics: The sparseness of matrix A can be seen as a measure of the agents' heterogeneity. If matrix A is *not* sparse, the behavior of the agents is homogeneous, because they evaluate nearly identical information sets. In this case, the average size of the agents' information set is large. If matrix A is sparse, the behavior of the agents is heterogeneous, because they are limited to small information subsets. Consequently, the average size of the agents' information set is relatively small.

In principle, heterogeneous decision behavior is required to adapt the underlying market dynamics. If the agents' decisions are too homogeneous, the market tends to be highly volatile. In contrast, if the agents are too heterogeneous, one may only observe small price movements.

Both cases do not reflect real-world behavior [WHD02, BMT96, p. 89-92, 124 and p. 177-8].

6.1.2 OUTLINE OF THE MULTIPLE FX-MARKET ANALYSIS

In the second part of the empirical study, we consider a multiple FX-market approach. Here, we evaluate the performance of the multi-agent models depicted in Fig. 7.3 and 7.7 by their application to the USD / DEM and the YEN / DEM FX-markets.

We compare our multi-agent models with two benchmarks, a naive trading strategy and a 3-layer feedforward network (MLP). The benchmark comparison is done on the basis of common performance measures (hit rate, realized potential and accumulated return) [Ref95, p. 69-71].

Let us describe the benchmarks: The first benchmark simply consists of a naive forecasting strategy, which assumes market trends [Ref95, p. 70]. The other benchmark is a 3-layer MLP with one input layer, a hidden layer consisting of 20 neurons and one output layer simultaneously predicting the price shifts of the USD / DEM and YEN / DEM FX-markets. The hidden neurons are equipped with tanh(.) activation functions [Zim94, p. 5-6]. The output layer incorporates the robust error function $\ln\cosh(.)$ (Eq. 5.9). We utilize the preprocessed input signals of our multi-agent models as inputs for the MLP. The MLP is trained until convergence by error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 161-68, p. 173-75 and p. 395-99]. We optimized the MLP by EBD weight pruning [TNZ97, p. 670-72].

We refer to three common performance measures: First, we calculate the accumulated return of each model [Ref95, p. 70]. Second, we employ a 'hit rate' counting how often the sign of the FX-rate shift is correctly predicted [Ref95, p. 69]. As a third measure, we included the so-called realized potential, which is defined as the ratio of the model return to the maximum possible return [Ref95, p. 71].

6.2 RESULTS OF THE EMPIRICAL STUDY

In the following we present the results of our empirical study. First, we deal with the single FX-market analysis (US-Dollar and German Mark). Thereafter, we focus on the modeling of multiple FX-Markets (US-Dollar, Yen and German Mark).

6.2.1 RESULTS OF THE SINGLE FX-MARKET ANALYSIS

Let us analyze the degree of heterogeneity among the agents, which is needed to capture the underlying market dynamics. Fig. 7.8 compares the accumulated return of the multi-agent models with the agents' average data base size.

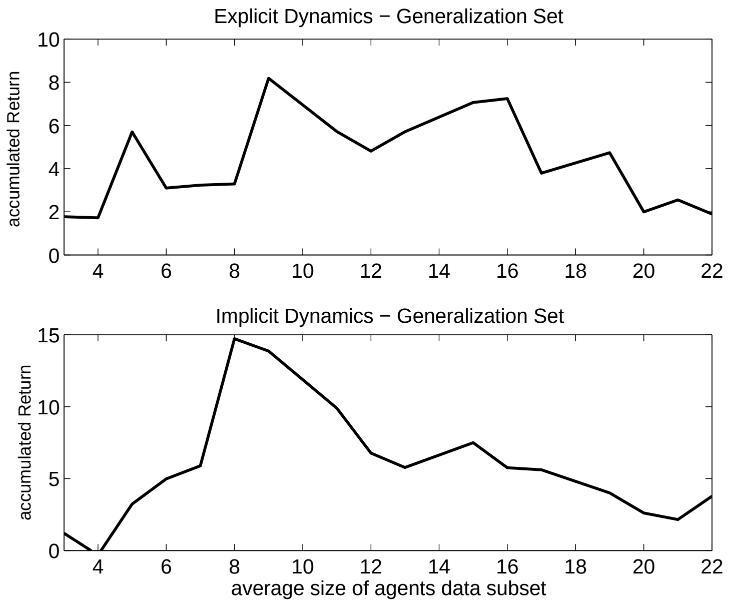


Figure 7.8 Accumulated model return calculated on the generalization set in dependence of the average size of the agents' data base.

For both market paradigms it turns out, that an optimal degree of heterogeneity is achieved, if the agents are restricted to a data subset of 9 input signals on the average. In other words, this parameterization of the information filtering process allows a superior fitting of the market dynamics. We utilize this insight for the calibration of the multi-agent models in the second part of the empirical study.

Fig. 7.9 depicts the development of the market excess (calculated on the generalization set) in dependence of the average size of the agents' data subset. Note, that we measured the market excess in absolute values.

We observe an expected behavior of the market excess in case of both market paradigms: If the average size of the agents' database is too large, the FX-market is driven by a huge market excess. In other words,

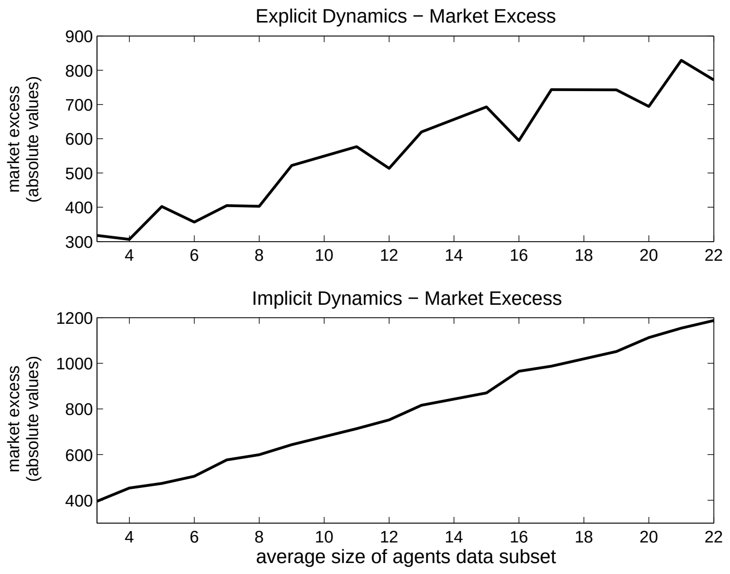


Figure 7.9 Absolute market excess of the USD / DEM FX-market in contrast to the average data base size of the agents. The market excess is measured on the test set.

if the behavior of the agents is too homogeneous, the market tends to be highly volatile. On the other hand, if the agents are too heterogeneous, the absolute level of the market excess is too small. Consequently, the FX-market is sluggish.

Fig. 7.9 indicates, that the implicit price dynamics leads to a higher market excess than in the explicit case. One may argue, that the implicit price dynamics is always associated with lower levels of market excess, because the market tries to vanish unbalances immediately by adjusting the price level. This can be seen as a minimization of the market excess. However, we claim that the latter comparison on the basis of the observed market excess is misleading: In each case the market is driven by an entirely different dynamics.

During our experiments we found that a number of 200 agents is sufficient to model the USD / DEM FX-market. A larger market population (e. g. 300 or 400) does not have an observable effect on the market dynamics. In contrast, smaller market sizes (e. g. 50 or 100) are insufficient to adapt the underlying dynamics of the FX-market. We will use this result for the calibration of the multiple FX-market approach.

6.2.2 RESULTS OF THE MULTIPLE FX-MARKET ANALYSIS

Let us proceed with the multiple FX-market analysis. We applied the multi-agent models depicted in Fig. 7.3 and 7.7 to the USD / DEM and YEN / DEM FX-markets. The empirical results are summarized in tables 7.1, 7.2 and 7.3.

Table 7.1 Hit Rates achieved on the generalization set

	Hit Rate			
	explicit	*implicit*	*MLP*	*naive strategy*
USD	66.58%	63.19%	60.34%	56.59%
YEN	46.65%	73.60%	63.71%	50.08%

The hit-rates of the USD forecasts indicate, that the explicit and the implicit approach are superior to the benchmarks. Since the hit rates of the multi-agent models are similar to each other, we may conclude that both approaches allow a good description of the USD / DEM FX-market dynamics. However one may argue, that both price dynamics cannot exist in parallel on a single market. We claim, that the USD / DEM FX-market is alternately driven by the market price regimes.

We observe a different behavior in case of the YEN / DEM FX-market. The implicit approach is not only superior to both benchmarks, but also to the model of the explicit price dynamics. Remarkably, the hit rate of the explicit approach is below those of the two benchmarks. It seems to us that the YEN / DEM FX-market is rather driven by the implicit than by the explicit price dynamics.

If we compare the accumulated returns achieved by trading USD, it becomes obvious that the naive trading strategy is superior to all other models. This result is due to an upwarding trend market in the first part of the generalization set. Remarkably, both multi-agent models are superior to the 3-layer MLP. Among the multi-agent models, the implicit approach achieves a slightly better accumulated return.

In case of the YEN / DEM FX-market, the model of the implicit price dynamics beats all other approaches. Once again, the model of the explicit price dynamics is inferior to both benchmarks. This re-indicates, that the YEN / DEM FX-market is mainly driven by the implicit market

Table 7.2 Accumulated returns achieved on the generalization set

Accumulated Return				
	explicit	*implicit*	*MLP*	*naive strategy*
USD	4.73%	6.65%	3.48%	8.73%
YEN	14.06%	25.33%	15.01%	-3.99%

dynamics, while on the USD / DEM FX-market both market paradigms can be found in an alternating sequence.

Table 7.3 Realized potentials achieved on the generalization set

realized Potential				
	explicit	*implicit*	*MLP*	*naive strategy*
USD	16.66%	23.42%	12.26%	30.75%
YEN	30.97%	55.79%	33.06%	-8.79%

Since the realized potential is derived from the accumulated model return, the preceding results are supported.

Interestingly, we found that a multiple FX-market approach (DEM, USD, YEN) is superior to a single FX-market analysis. This is due to the fact, that a model of a single market pays no attention to inter-market dependencies. For instance, a separate analysis of the USD / DEM (YEN / DEM) FX-market on the basis of the explicit price dynamics yields a hit-rate of 60.90% (43.02%). In contrast, we achieve a hit rate of 66.58% (46.65%) by referring to a multiple market approach (Tab. 7.1). In the light of these results, we suggest to use an integrated approach instead of analyzing the concerned FX-markets separately.

7. PRELIMINARY CONCLUSION

Our neural network based approach integrates two perspectives of financial market modeling: We combined explanatory multi-agent models of economic markets with an econometric forecasting framework.

We have shown that the embedding of both tasks into a neural network architecture is natural: First of all, a single neuron reflects the decision making process of a single agent. In mind this interpretation, a neural network consisting of many neurons represents the interaction of many decisions and can thus be seen as a market process.

Furthermore, we have a natural bandwidth between non-linearity and linearity. The neural network framework allows us to model different market price dynamics. Based on local specifications and global learning, the resulting agent-based models enable us to fit real-world financial data and to predict future market price movements.

Most remarkable, our multi-agent models concern semantic specifications instead of being limited to ad-hoc functional relationships [EM01, p. 759-61]. The most obvious direction of future research is towards a more realistic modeling of the agents' decision making behavior. Such an approach is presented in the next chapter.

Chapter 8

MULTI-AGENT FX-MARKET MODELING BASED ON COGNITIVE SYSTEMS

Intending to model a financial market by a multi-agent based approach, one faces at least two major tasks [LeB01a, WHD02, BMT96, p. 255-7, p. 88-93 and p. 175]: *(i.)* modeling the decision behavior of a single agent on the micro-level together with *(ii.)* constructing a price formation mechanism on the macro-level. The latter task can be solved by calculating the market excess out of the agents' trading decisions. Changes in the market price level occur in order to vanish the observed market unbalances [Far98, WHD02, p. 6-12 and p. 88-93].

Furthermore, there are many ways of modeling the agents' decision making behavior [WHD02, LeB01a, p. 90-2 and p. 255-6]. For instance, a single neuron can be interpreted as an elementary decision making scheme of a single agent [Zim89, Zim94, p. 496-7 and p. 3-6]. It unites the three key steps of a decision making process: Information filtering, the formation of a superposition of information and the execution of an action. From this point of view, a neural network, which may include hundreds of neurons, reflects the interaction of a large number of decision makers resp. a market process [Bar97, p. 813-4].

This chapter deals with multi-agent models of FX-markets. The decision behavior of the agents is based on the idea of an elementary cognitive system. As we will show, the suggested cognitive process allows us to describe and to explain the decision making of the agents in a more natural way [EM01, p. 759-61].

We propose a cognitive system with three basic features [RN95, LN77, Per01, Bar97, p. 31-48, 508-19, p. 587-619, p. 391-412 and p. 393-419]: (*i.*) perception, (*ii.*) internal processing and (*iii.*) action. The cognitive system is structurally represented by a time-delay recurrent neural network using unfolding in time and shared weights [RHW86, p. 354-7].

In a first approach (sec. 1.) we *inductively* describe the basic features of the cognitive system [ZNG01b, p. 768-70]. We model the cognitive system with a time-delay recurrent error correction neural network (ECNN, see chp. 4). Merging cognitive systems with the economic theory of the multi-agent market models, we explain the microeconomic behavior of the agents by time-delay recurrent neural networks. As we will show, our multi-agent model enables us to capture the underlying market dynamics and to fit real-world financial data.

In a second approach (sec. 2.) the basic features of the cognitive system are derived *deductively* from the assumption of homeostasis [CMcV01, p. 11-17]. Given a changing environment, homeostasis can be seen as the attempt of a cognitive agent to maintain an internal equilibrium (steady state). A structural representation of homeostasis is given by so-called zero-neurons within a time-delay recurrent neural network [ZGTN02].

As an extension to the deductive approach we deal with the identification of entities resp. binding [Vel95, Marsh95, p. 250 and p. 590]. Binding is a clustering process by which distinct characteristics (features) of an object or a dynamic system are grouped. We solve the binding problem by a variants-invariants separation in form of a bottleneck (autoassociator) neural network [RHW86, CS97, p. 335-9 and p. 101-7].

It should be annotated, that our considerations about inductive and deductive cognitive agents (sec. 1. and 2.) should be seen as necessary conditions for cognitive systems [RN95, p. 13-4].

1. AN INDUCTIVE APPROACH TO COGNITIVE AGENTS

In this section, we present a multi-agent market model in which the decision behavior of the agents is based on cognitive systems with three basic features (perception, internal processing and action). We inductively describe the basic features of the cognitive system and deal with their embedding in a structural framework. More precisely, we model the cognitive system by an error correction neural network (see chp. 4).

The market orders of the agents are collected on the macroeconomic side of the market. The net of the market orders (market excess) is used to determine price changes. In other words, the market price level is adjusted as a response to the observed market excess. By this, we are able to capture the underlying market dynamics.

In an empirical study we apply our multi-agent model to the US-Dollar / German Mark FX-Market. We compare our model to several benchmarks (linear regression, 3-layer MLP and a time-delay recurrent neural network). Different performance measures (Sharpe ratio, realized potential, hit rate) indicate that our multi-agent model is superior to the preset benchmarks.

This section is divided into five subsections: Subsection 1.1 inductively derives a basic model of a cognitive system and explains its features. In subsection 1.2, we embed the basic features of the cognitive system into a structural framework, which is given by a standard error correction neural network (ECNN). Subsection 1.3 concentrates on the structure of our multi-agent model. We explain the trading schemes of the agents and the related market price formation mechanism. Subsection 1.4 establishes the crosslink between inductive cognitive systems and multi-agent modeling. We present a higher-level ECNN architecture, which incorporates the cognitive systems of the agents and the price formation mechanism of the market. In subsection 1.5 we apply our multi-agent model to the USD / DEM FX-market.

1.1 NECESSARY CONDITIONS FOR COGNITIVE SYSTEMS

Let us proceed with a inductive description of a basic cognitive systems, which might help to model the decision making of economic agents. We propose a cognitive system with three interdependent features [Per01, RN95, Bar97, GS83, p. 101-4, 126-7, p. 31-9 and p. 393-419]: (*i.*) perception, (*ii.*) internal processing and (*iii.*) action.

The features specify necessary conditions for a cognitive agent [Cra43, RSMH86, RN95, GS83, p. 57, p. 40-4 and p. 13-4]. According to the concept of mental models proposed by Kenneth Craik (1943) [Cra43], an agent should at least incorporate three features: (*i.*) a mechanism that translates stimulus (perception) into an internal representation, (*ii.*) an internal model that processes the internal representation in order to derive new internal states, and (*iii.*) a mechanism that in turn transforms the new internal states into action. Craik describes the benefits of these features as follows [Cra43, p. 57]:

> "If the organism [agent] carries a 'small-scale' model of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to emergencies which face it."

As shown in Fig. 8.1, the basic features of the cognitive system are closely interrelated [RSMH86, Bar97, RN95, p. 40-4, p. 404-6 and p. 31-7]: Perception and action have an overlap with the internal processing. Suppose, an agent perceives relevant information about his environment. The perception may cause changes in the internal state, which may in turn lead to actions [Cli95, p. 627]. Hence, the best way to understand perception as well as action is by analyzing their overlap with the internal

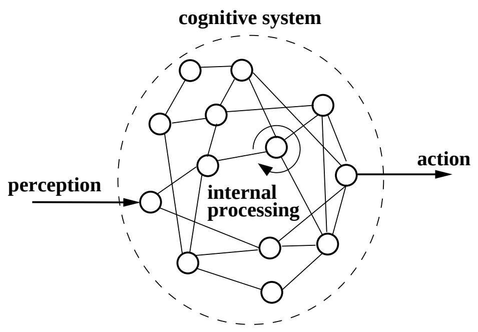


Figure 8.1 Interrelated features of the cognitive system

processing. In the following, we explain the features of the cognitive system in greater detail.

1.1.1 PERCEPTION

In general, perception may be defined as the observation of the system environment [LN77, p. 87-9]. We distinguish conscious from unconscious perception [Bar97, RN95, Per01, p. 411-9, p. 32-3 and p. 393-5]. In conscious perception, the cognitive system compares selective observations to an internal expectation. Only the resulting difference has a further impact on the system. In other words, attention solely focuses on the difference between the observation and the internal expectation [Met00, Bar97, p. 112-3 and p. 416-9]. Conscious perception is of minor interest, if there is only a small difference between the internal expectation and the external observation.

Unconscious perception enters the system directly, i. e. it corresponds to stimulus-response [RN95, Bar97, p. 202-3 and p. 411-8]. Unconscious perception is therefore *not* balanced with internal expectations. In other words, unconscious perception is not within the attention of the system and thus, no internal expectations are generated as a compensation. This complies with a stimulus response mechanism.

1.1.2 INTERNAL PROCESSING

In our sense, the internal processing mainly consists of an internal model of the external world, which combines perception and memory [RN95, RSMH86, LN77, Bar97, GS83, p. 41-2, 203-4, p. 40-4, p. 589-95 and p. 404-9]. In other words, the internal model can be seen as a reflection of the surrounding environment, or at least of its most relevant aspects.

The internal model generates expectations, which are in turn used for conscious perception. For the construction of these expectations, the internal model evaluates actual and past perceptions [RN95, p. 203-4]. Hence, an internal memory is required. This memory can be seen as a merging of all relevant past information from which an internal expectation is formed [LN77, CS97, p. 303-8 and p. 312-3, 317]. The memory is crucial for a cognitive system, unless there is another way to build an internal dynamics.

The internal memory can be seen as a knowledge data base in which perceptions are stored by their chronological order [RN95, p. 203-4 and p. 217]. The temporal sequence of the perceptions is required, because without a certain chronological order many events are incomprehensible [CS97, p. 395-8]. For instance, the internal model may refer to the memory in order to identify an object along different time periods. Otherwise, an object that changes its location from t to $t+1$ would be recognized as a completely new one. In addition, the knowledge data base enables the cognitive system to supplement imperfect conscious perceptions [LN77, p. 366-77]. For example, a noisy or incomplete perception of an object may be corrected by referring to the memory, which contains a complete description of the latter.

As a result, the internal processing delivers decision support in form of internal expectations. This is in turn the basis for action [Cra43, RSMH86, GS83, p. 57 and p. 40-4]. Interestingly, the internal model may also stay in an internal equilibrium (i. e. a homeostatic or dynamical stable state) [CMcV01, p. 11-17]. This means, that the internal processing evaluates perceptions without giving any kind of decision support. We will explain this property of the cognitive system later on (see sec. 2.).

1.1.3 ACTION

To initiate actions, the internal expectations are evaluated by a specific objective function (e. g. profit or utility maximization) [RN95, RSMH86, p. 31-3, p. 41-5, p. 471 and p. 40-2, 48]. Hence, actions are always goal-oriented. The structure of the underlying objective function depends on the preferences of the individual [RN95, p. 32].

Perception and action are decoupled: External stimuli only induce actions if they disturb the cognitive system [CMcV01, Bar97, p. 11-17 and p. 405]. Due to this decoupling, the actions of the cognitive system are driven by the abstract objective function rather than by a simple stimulus-response mechanism.

Remarkably, the cognitive system is able to perceive the impact of its own actions [Jua99, p. 25-7]. Note, that this is also a necessary condition for self-consciousness [Vel95, Per01, Met00, p. 247-50, p. 393-5 and p. 78-9].

Consider the example of an agent trading on a stock market: The decision support of the agent's internal model may consist of an expectation about the future development of the stock price. This forecast must be turned into an action (trading decision). Therefore, the agent evaluates the internal price expectation by his objective function (e. g. expected profit maximization). Based on the latter evaluation, the agent may perform an action like buying or selling a particular amount of shares.

An action is always prepared by the internal processing [Cra43, p. 55-7]. This can be seen as an overlap between the two features (see Fig. 8.1). Another overlap is between action and perception, because the system immediately perceives its own actions [Jua99, p. 25-7].

1.2 MODELING COGNITIVE SYSTEMS BY ERROR CORRECTION NEURAL NETWORKS

We suggest to model the basic features of the cognitive system by an error correction neural network (ECNN, see chp. 4). For convenience, the ECNN is reconsidered in Fig. 8.2.

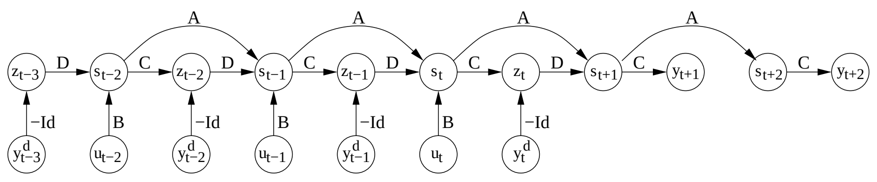


Figure 8.2 Transferring the proposed features of the cognitive system into the structural framework of an ECNN. The ECNN consists of an autonomous part (coded in the matrix A), which models the recursive structure of the underlying dynamical system. Additionally, we have an error correction mechanism (coded in C, D), which is handled as an auxiliary model input in order to compensate external shocks. Note, that only the difference between the internally expected value y_τ and the observation y^d_τ has an impact on the state transition s_τ. Besides the error correction inputs $y_\tau - y^d_\tau$, external input signals u_τ are transferred into the system by using matrix B. These inputs influence the state transition directly, because they are not linked to the error correction mechanism of the ECNN. The ECNN is an unfolding in time neural network architecture, which uses shared weights [RHW86, p. 354-7].

In the following we show that the characteristics of the ECNN depicted in Fig. 8.2 match the proposed features of the cognitive system.

1.2.1 PERCEPTION

Conscious and unconscious perception are structurally represented by the different input flows of ECNN. Input signals either have to pass the error correction mechanism or are directly transferred into the system.

During conscious perception, inputs are compared to a certain internal expectation. The cognitive system only pays attention to the difference between the internal expectation and the external observation. In other words, only the resulting difference between both values may cause changes in the system.

Conscious perception is modeled by the error correction mechanism of the ECNN. Here, an input signal y^d_τ is compared to a certain internal expectation $y_\tau = Cs_\tau$. The difference $y^d_\tau - y_\tau$ is given to the system in form of an input signal. Consequently, only the difference between the internal expectation y_τ and the external observation y^d_τ may influence the internal state $s_{\tau+1}$ of the network.

Incomplete or missing conscious perceptions are automatically corrected by the ECNN. At each future time step $t + \tau$ of the unfolding, the ECNN offers forecasts $y_{t+\tau}$, because there is no compensation for the internal expectations.

Unconscious perceptions u_τ are directly transferred into the system. In other words, unconscious perception are *not* compared to an internal expectation of the system. This corresponds to a stimulus response mechanism.

The structural representation of unconscious perception is given by the external inputs u_τ of the ECNN. These input signals enter the internal state s_τ of the system directly without being affected by the error correction part. As it can be seen from this construction, the inputs u_τ have a direct impact on internal state s_τ of the ECNN.

From a more technical point of view, the ECNN can be seen as an integrated forecast model for *all* conscious perceptions. To avoid the construction of an omniscient model of the world, the learning should be focused on the most important conscious perceptions. Another possibility is to omit the error correction mechanism for most external observations, i. e. handling them by unconscious perception. Hence, the ECNN only forms internal expectations about the most relevant external stimuli. We obtain a conscious description of the external world with a focus on its most relevant aspects.

If most of the inputs are processed in a conscious way, the cognitive system has a predominantly analytical picture of the world, because most of the input is evaluated against the internal model. In the dual case of mainly unconscious perception, the cognitive system has a more 'emotional' picture of the world. Concerning the behavior of agents, one may argue that a more analytical description of the world corresponds to the behavior of a financial analyst, while the more 'emotional' picture can be attributed to a trader, who often relies on market rumors or moods [San99]. The more inputs are handled consciously, the more complete the internal 'world model' of system has to be.

1.2.2 INTERNAL PROCESSING

The internal processing of the cognitive system has its structural representation in the autonomous dynamics of the ECNN. Recall, that the ECNN is formulated as a state space model [Hay94, MC01, p. 739-46 and p. 44-5]. For a dynamic system, such as the ECNN, the state represents "a set of quantities that summarizes all the information about the past behavior of the system that is needed to uniquely describe its future behavior, except for the purely external effects arising from the applied input (excitation)" [Hay94, p. 739].

According to this description, the state of the ECNN represents the internal model as well as the memory of the cognitive system. – The internal dynamics of the ECNN reflects the external world. This corresponds to the internal model of the cognitive system. The memory generated by the state of the ECNN supports the formation of internal expectations. It can be seen as a superposition of all relevant past information. This is analogous to the memory of the cognitive system.

The internal model combines perception and memory. It forms internal expectations, which are used for conscious perception. The internal expectations of the cognitive system are derived by the output equation $y_\tau = Cs_\tau$ of the ECNN (see Fig. 8.2).

Similar to the internal model of the cognitive system, the ECNN is able to supplement imperfect conscious perceptions. If a conscious perception is incomplete or missing the ECNN automatically generates a replacement. At all future time periods $t + \tau$, the ECNN offers forecasts, because we have no compensation of the internal expectation per definition.

Overshooting forces the ECNN to focus on the autonomous part of the dynamics. This may also improve the learning of the cognitive system. If the cognitive system has to form internal expectations about the near future (overshooting), the system is in the need of exploring long-term dependencies instead of focusing on short-term events. Nevertheless, short-term influences or external shocks are handled by the error correction mechanism of the network.

1.2.3 ACTION

Up to now, the feature 'action' is not represented in the structural framework of the ECNN (see Fig. 8.2): The ECNN has neither an action variable (e. g. indicating a trading decision) nor a particular objective function (e. g. expected profit maximization), which evaluates the internal expectations. In the following we integrate an action variable directly into the ECNN framework, such that acting becomes a part of the underlying dynamics.

The ECNN depicted in Fig. 8.3 incorporates an additional error correction branch, which models the action of the cognitive system.

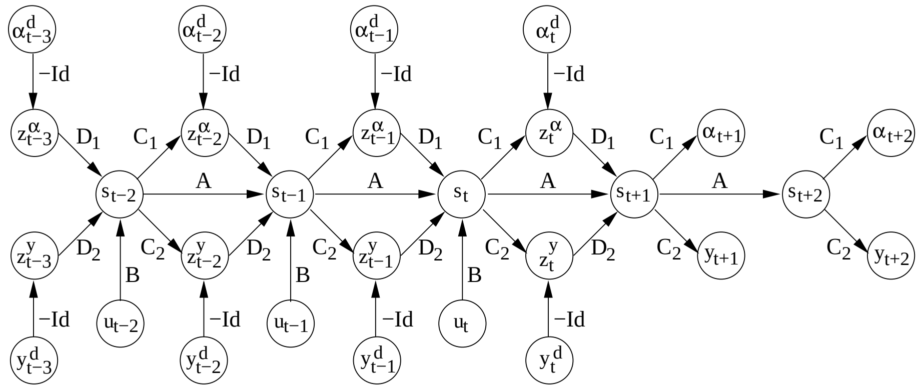


Figure 8.3 Integration of action into the structural environment of the ECNN

Besides the internal expectations $y_\tau = C_2 s_\tau$, the cognitive system suggests actions $\alpha_\tau = C_1 s_\tau$. The actions α_τ can be seen as intentions of the cognitive system (also called intended actions α_τ) [Jua99, RN95, p. 187-8 and p. 657].

The system perceives the corresponding state of the environment y^d_τ *and* the actually executed action α^d_τ. Recall, that the cognitive system immediately perceives its own actions. The internal expectation y_τ is compared to the observation y^d_τ, whereas the intended action α_τ is compared to the observed action α_τ. Only the resulting differences $y_\tau - y^d_\tau$ and $\alpha_\tau - \alpha^d_\tau$ have an impact on the cognitive system.

If the intended action α_τ cannot be executed, the observed action α^d_τ and the intention α_τ differ. For instance, a cognitive agent may intend to buy a certain amount α_τ of shares. If the market order of the agent is not filled, the intention α_τ and the observed action α^d_τ differ. The observed difference $\alpha_\tau - \alpha^d_\tau$ may induce changes in the system.

At each future time step of the unfolding $t + \tau$, the cognitive system generates not only forecasts $y_{t+\tau}$, but also intended actions $\alpha^d_{t+\tau}$. Due to the additional error correction branch, action is embedded into the internal state. This means, that action is an integral part of the system dynamics.

The separate error correction branch of the ECNN enables us to assign a different error function to the action variable α_τ. Since we want to model trading decisions, we applied the *prof-max* cost function (Eq. 7.2) [Ref95, p. 69-70]. The behavior of an agent is therefore driven by an expected profit maximization task.

The design of the ECNN depicted in Fig. 8.3 implies that the cognitive system consciously perceives its own activities as soon as they are executed [Jua99, p. 25-7]. In doing so, the internal model reflects not only the environment, but it is also conscious of itself [Per01, p. 393-5]. This corresponds to the overlap between action and perception.

1.3 AGENTS' TRADING SCHEME & MARKET PRICE FORMATION

Let us revisit the basic elements of multi-agent modeling in case of a single FX-market. In particular, we discuss the agents' trading scheme and the price formation mechanism of the market (see chp. 7). As an example we consider the US-Dollar / German Mark FX-market. Throughout this section we assume that DEM is the base currency, whereas USD is treated as counter currency.

Each agent $a = 1, \ldots, A$ has two accounts x^a_t(in USD), y^a_t(in DEM), which reflect his dispositions in USD and DEM. The accounts are expressed in terms of the respective currencies. For simplicity, we neglect transaction costs and assume that the currency holdings pay zero interest.

Recognizing that p_t is the USD / DEM FX-rate and α^a_t is the amount of money an agent desires to exchange, the upcoming cash balances can be written as

$$\begin{aligned} y^a_{t+1}(\text{in DEM}) &= y^a_t(\text{in DEM}) + \alpha^a_t \ , \\ x^a_{t+1}(\text{in USD}) &= x^a_t(\text{in USD}) - p_t\alpha^a_t \ . \end{aligned} \tag{8.1}$$

Periodically an agent chooses among three trading alternatives α^a_t:

1. Purchase the base currency (DEM), which equals $\alpha^a_t > 0$.

2. Sell the base currency (DEM), which equals $\alpha^a_t < 0$.

3. Retain invested funds unchanged, which equals $\alpha^a_t = 0$.

To solve this decision making task, the agent evaluates his expectation about the future development of the market by his objective function. In other words, the agent turns his internal price forecast into an investment decision [Ref95, p. 70].

We assume, that the behavior of each agent $a = 1, \ldots, A$ is driven by an expected profit maximization task [Ref95, p. 69-70]:

$$\frac{1}{T} \sum_{t}^{T} \hat{\pi}_{t+1} \cdot \alpha_t^a \rightarrow \max_{\alpha_t^a} \ , \quad \text{with } \hat{\pi}_{t+1} = E\left(\ln\left(\frac{p_{t+1}}{p_t}\right)\right) \ . \tag{8.2}$$

where $\hat{\pi}_{t+1}$ is the expected FX-rate shift and α_t^a is the agent's trading decision. Each agent $a = 1, \ldots, A$ tries to exploit expected changes $\hat{\pi}_{t+1}$ of the FX-rate, such that his expected profit is maximized. Suppose, that the agent expects a positive shift in the USD / DEM FX-rate ($\hat{\pi}_{t+1} > 0$). The trading decision that will maximize the agent's expected profit is to buy the base currency DEM ($\alpha_t^a > 0$).

Up to now, the agents are unable to interact, i. e. the activities of agent a have no effect on the trading decisions of other agents. The interaction of the agents can be established by the *market clearance condition* formulated in Eq. 8.3 [Far98, p. 7-8]:

$$\sum_{a}^{A} \alpha_t^a = 0 \ . \tag{8.3}$$

The market clearance condition (Eq. 8.3) represents the macroeconomic side of the FX-market. An agent can only buy USD ($\alpha_t^a < 0$) if there are other agents who are selling a congruent amount ($\alpha_t^a > 0$). Hence, the plans of the agents are indirectly associated with each other [Kir96, p. 1].

In a market equilibrium no price adjustments are needed, because the trading activities of the agents match (Eq. 8.4). Otherwise, a price shift π_{t+1} is performed as a response to the observed market excess (i. e. demand - supply) [Far98, p. 7-9]. Noticing, that $c > 0$ is a constant, the price shift is estimated by Eq. 8.5.

$$\sum_{a}^{A} \alpha_t^a = 0 \quad \Rightarrow \quad \pi_{t+1} = 0 \ . \tag{8.4}$$

$$\sum_{a}^{A} \alpha_t^a \neq 0 \quad \Rightarrow \quad \pi_{t+1} = c \cdot \sum_{a}^{A} \alpha_t^a \ . \tag{8.5}$$

Eq. 8.5 models the price formation mechanism of the FX-market. A change in the FX-rate can be seen as a reaction to an observed market excess. The FX-rate shift π_{t+1} is estimated by scaling the market excess $\sum \alpha_t^a$ with a positive constant $c > 0$. The degree of the price reaction depends on the size of the observed market excess.

Note, that this multi-agent framework is not only a descriptive model of the underlying market dynamics, but also an explanatory approach [Far99a, p. 9991-2].

1.4 FX-MARKET MODELING BY INDUCTIVE COGNITIVE AGENTS

The cognitive system based multi-agent approach depicted in Fig. 8.4 serves our purpose of FX-market modeling. The ECNN includes the cognitive systems of *all* agents and the market price formation mechanism described in Eq. 8.5. In the following we explain, how the agents are arranged in the network architecture. Thereafter, we focus on the modeling of the price formation mechanism.

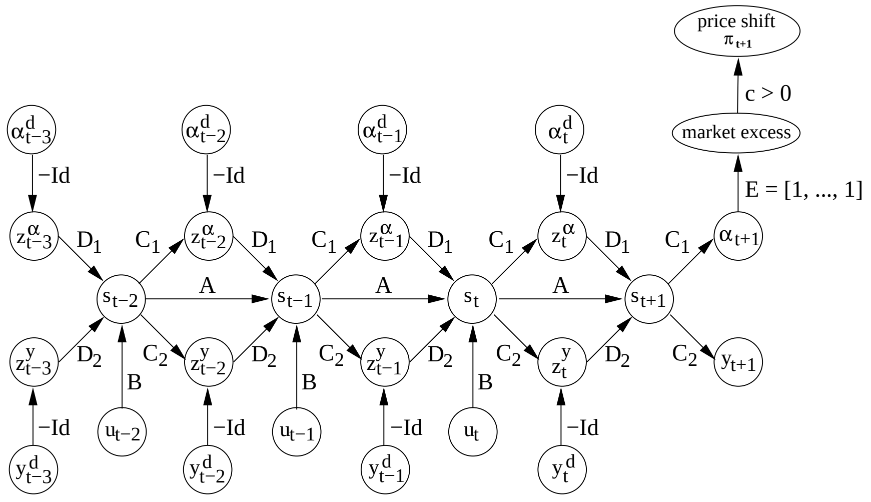


Figure 8.4 FX-market modeling by a multi-agent approach based on cognitive systems

1.4.1 ARRANGEMENT OF THE AGENTS

The cognitive system of each agent $a = 1, \ldots, A$ is structurally represented by an ECNN as depicted in Fig. 8.3. At each time step, the agent forms an intended action $\alpha_\tau = C_1 s_\tau$ and compares it with the actually executed action α_τ^d. The intended actions α_τ can be seen as the market orders of the agent. The actions of the agent are driven by an expected profit maximization task (Eq. 8.2). The agent consciously perceives the FX-market development y_τ^d and observes other external influences u_τ unconsciously.

Now the question arises, how the cognitive systems of the agents can be arranged in a unified FX-market model. An obvious approach would consist of building a separate cognitive system for each agent taking part in the FX-market. Afterwards, the market orders (intended actions) of the agents can be collected in order to derive FX-rate changes.

However, we choose a slightly different approach to build an integrated FX-market model. Instead of generating a separate ECNN for each agent, we propose to merge all market participants in *one* higher-level network architecture (Fig. 8.4). Of course, we have to guarantee, that the cognitive systems of the agents are separated from each other. In other words, the network design has to ensure that the agents exist in parallel without influencing each other. The *only* interaction among the agents is due to market price formation mechanism.

The higher-level network architecture depicted in Fig. 8.4 separates the agents by block diagonal matrices A, C_1, C_2, D_1 and D_2. A single block in each matrix refers to one agent. For instance, agent $a = 1$ is modeled by the first block in *each* matrix, whereas agent $a = A$ is represented by the last block in each matrix. Due to the block design, the agents interact only through the price formation mechanism. Note, that this design also implies that the layers of the neural network are constructed appropriately.

Heterogeneity among the agents is introduced by different sets of external influences [BMT96, WHD02, p. 178 and p. 89-92]. Heterogeneous decision making is required to adapt the underlying market dynamics. Recall, that each agent observes external influences u_τ unconsciously. Unconscious perceptions enter the cognitive systems of the agents through matrix B. In order to generate heterogeneous decision behavior, we spread the external influences u_τ among the agents. Therefore we designed matrix B as a randomly initialized sparse connector. In other words, the sparseness of matrix B can be seen as a measure of the agents' heterogeneity. For further details of heterogeneous decision making, the reader is referred to chapters 6 and 7.

1.4.2 MARKET PRICE FORMATION

The price formation mechanism consists of two steps [Far98, LeB01a, p. 7-9 and p. 256]: First, we collect the market orders α_τ of the agents and compute the resulting market excess (i. e. demand - supply). Second, we determine a FX-rate shift $\pi_{\tau+1}$ as a response to the observed market excess.

The calculation of the market excess (i. e. the net of the agents' market orders α_τ) is done by the fixed matrix $E = [1, \ldots, 1]$. The market excess cluster is designed as a hidden layer with an identity activation function.

The FX-rate shift is estimated by scaling the observed market excess with a proportional constant $c > 0$. The 'price shift' layer is an output cluster, which is equipped with the robust error function ln cosh (Eq. 5.9) [NZ98, p. 387-88]. We use the observed FX-rate shifts π_{t+1} as target values.

Finally note, that we propose to train the ECNN by using the shared weights extension of error backpropagation together with *vario-eta* learning rule [RHW86, NZ98, p. 354-7 and p. 396].

1.5 EMPIRICAL STUDY: MODELING THE USD/DEM FX-MARKET

In this subsection we present an empirical study modeling the US-Dollar / German Mark FX-Market. We apply the cognitive system based multi-agent model depicted in Fig. 8.4 to predict the weekly development of the USD / DEM FX-rate. The performance of our multi-agent approach is evaluated in a benchmark comparison. The next paragraphs outline the design of the empirical study and describe the obtained results.

1.5.1 OUTLINE OF THE EMPIRICAL STUDY

The primary purpose of the empirical study is to demonstrate the application of our multi-agent approach to a real-world FX-market. We forecast the weekly changes in the USD / DEM FX-rate. The prediction accuracy of the multi-agent model is compared to several benchmarks (e. g. linear regression model, 3-layer MLP, basic recurrent neural network and an ECNN). The benchmark comparison is based on common performance measures (e. g. Sharpe Ratio, realized potential or hit rate). In the following we describe the experimental set-up (data, performance measures) and the different models (multi-agent approach, benchmarks).

Data. We work on basis of weekly data from Jan. 1991 to Aug. 1997 [343 data points]. The complete data set is divided into two subsets: The training set (in-sample-period) ranges from Jan. 1991 to Dec. 1995 [258 data points]. The generalization set (out-of-sample period) covers the time period from Jan. 1996 to Aug. 1997 [85 data points]. German Mark is used as the base currency, whereas US-Dollar is treated is counter currency.

Input Signals & Preprocessing. The selection of the external inputs u_t is based on the idea, that a major currency is driven by at least *four* influences [Mur91]. These are the development of (*i.*) other currencies, (*ii.*) bond, (*iii.*) stock and (*iv.*) commodity markets. The used input time series are summarized in Tab. 8.1.

Table 8.1 Raw input time series used for the multi-agent model and the benchmarks

Model Inputs	
Japanese Yen / USD FX-rate	Japanese Yen / DEM FX-rate
British Pound / USD FX-rate	British Pound / DEM FX-rate
3-month interest rate USA	3-month interest rate Germany
3-month interest rate Japan	10-years interest rate USA
10-years interest rate Germany	10-years interest rate Japan
Dow Jones Stock Index	German DAX 30
Nikkei 225	FTSE 100
CRB-Future Index	Gold Price per Oz. in USD

As shown in Tab. 8.1, we use specific economic data of the USA, Japan, Germany and Great Britain to predict the development of the USD / DEM FX-market. In total, we have a number of 16 time series, which serve as raw input signals for the multi-agent model and the benchmarks.

In the preprocessing, we calculate the momentum (relative change, see Eq. 2.7) and force (normalized curvature, see Eq. 2.8) of each raw input time series [NZ98, p. 374]. The transformed time series are scaled such that they have a mean of zero and a variance of one [PDP00, p. 73]. We obtain a number of 32 preprocessed input time series for the multi-agent model and the benchmarks.

Multi-agent model & benchmarks. The multi-agent model is depicted in Fig. 8.4. The market model consists of a number of 200 agents. Each agent is modeled by a cognitive system with three basic features (perception, internal processing and action). The cognitive system of each agent is represented by an ECNN as shown in Fig. 8.3. We arranged the cognitive systems of the agents in the higher-level architecture shown in Fig. 8.4. The cognitive systems are separated by block diagonal matrices.

The agents consciously perceive the development of the USD / DEM FX-market and observe the external influences u_τ (Tab. 8.1) unconsciously. To generate heterogeneous decision behavior, each agent is limited to an average number of 8 external inputs u_τ. We spread the external inputs u_τ among the agents by the sparse matrix B (see Fig. 8.4). The sparseness of matrix B is initialized by random.

Recall, that the experiments in chapter 7 indicated that an average number of 8 input signals generates an optimal degree of heterogeneity among the agents. Furthermore, we found that a population of about 200 agents is sufficient to model the underlying dynamics of the USD / DEM FX-market.

We use the market price forecast of the multi-agent model for the benchmark comparison. In other words, the forecast results from the buying and selling decisions of the different agents (see Fig. 8.4, 'price shift' layer). The multi-agent model is trained until convergence by using *vario-eta* learning and backpropagation through time [NZ98, Hay94, p. 354-7 and p. 751-4].

We compare the multi-agent model to several benchmarks: (*i.*) a standard ECNN (see Fig. 4.6), (*ii.*) a basic time-delay recurrent network (see Fig. 4.4), (*iii.*) a 3-layer feedforward neural network (MLP, see Fig. 2.8) and (*iv.*) a linear regression model [PDP00, p. 201-11]. The benchmarks use the preprocessed time series shown in Tab. 8.1 as input signals. Further characteristics of the benchmarks are outlined in Tab. 8.2.

Except the linear model, all benchmarks are endowed with the robust error function $\ln\cosh$ (Eq. 5.9) [NZ98, p. 387-88]. The linear regression model uses squared errors [Wei90, p. 179]. The unfolding in time neural networks incorporate an oversized overshooting branch. Although we only use the one week forecast horizon $(t+1)$, the additional future time steps act as a further regularization method for the learning.

Remarkably, the standard ECNN (Tab. 8.2) can be interpreted as a single agent participating in the USD / DEM FX-market. The agent consciously perceives the USD / DEM FX-rate and forms internal expectations about the future market development (see Fig. 8.2). Therefore, a comparison of this benchmark with the integrated multi-agent market model should be interesting.

Table 8.2 Description of the benchmarks used in the empirical study

Benchmarks	
Model	*Description*
Error correction neural network (ECNN)	The architecture of the ECNN is shown in Fig. 4.6. The unfolding in time incorporates a number of 12 time steps ($t-6$ to $t+6$). This means, that we have six past time steps and an additional overshooting branch of six future time steps. Each time step of the unfolding represents one week. The network is trained by vario-eta learning and standard error backpropagation using shared weights.
Recurrent neural network (RNN)	The basic unfolding in time network is depicted in Fig. 4.4. The truncation length of the finite unfolding in time is equal to that of the ECNN, i. e. we choose $t-6$. The RNN also incorporates an overshooting branch of six future time steps $t+1,\ldots,t+6$. The network is trained by backpropagation through time and *vario-eta* learning rule.
3-layer MLP	The 3-layer feedforward network (MLP, see Fig. 2.8) incorporates a hidden layer with 20 neurons. We use tanh activation functions [Zim94, p. 5-6]. The output layer generates a one week prediction of the USD / DEM FX-rate shift. The network is trained by error backpropagation using pattern-by-pattern learning rule [Hay94, NZ98, p. 161-68 and p. 395-99]. We optimized the MLP by EBD weight pruning [TNZ97, p. 670-72].
Linear regression	The linear regression model forecasts the one week USD / DEM FX-rate shift on the basis of the external influences u_τ [PDP00, p. 201-11]. The statistical significance of the regression coefficients is evaluated by t-tests [PDP00, p. 279-83]. We analyzed the model residuals by the Durbin-Watson test statistic to detect the presence of autocorrelation [PDP00, p. 297-307]. The parameters of the linear model are estimated by ordinary least squares (OLS) [PDP00, p. 211-20].

Performance measures. The prediction accuracy of the different models is measured on the basis of *five* common performance measures: (*i.*) hit rate, (*ii.*) Sharpe ratio, (*iii.*) accumulated return of investment, (*iv.*) realized potential and (*v.*) annualized return [Ref95, p. 70-74]. All performance measures are calculated on the out-of-sample period (Jan. 96 – Aug. 97).

The hit rate counts how often the sign of the forecast is correct.

The Sharpe ratio is defined as the excess return of the model (difference between the model return r_M and a risk-free return r_f) divided by the standard deviation σ_M of the model return, i. e. $SR = (r_M - r_f)/\sigma_M$ [EG95, PDP00, p. 639 and p. 267-8]. The average return of a risk-free asset is 4% during the out-of-sample period (Jan. 96 – Aug. 97).

The accumulated model return is computed on the basis of a simple technical trading strategy. If we expect a positive (negative) return, we take a long (short) position. The accumulated model return is defined as the sum of the returns gained from the different trades.

The realized potential is the ratio of the accumulated model return to the maximum possible accumulated return.

The annualized return is the average yearly profit of the preceding trading strategy.

1.5.2 RESULTS OF THE EMPIRICAL STUDY

Let us turn to the results of our empirical study. The performance measures calculated for the different models on the generalization set are assembled in Tab. 8.3. The accumulated model returns are depicted in Fig. 8.5. Fig. 8.6, left, compares the actual weekly changes in the USD / DEM FX-rate during the out-of-sample period with the corresponding forecasts of the multi-agent model. The predictions of the FX-rate shifts are based on the market excess observed in the agent-based market (see Fig. 8.6, right).

Comparing the performance of our multi-agent model with the different benchmarks, it becomes apparent, that the agent-based approach is superior to the traditional econometric methods (linear regression, MLP). In other words, the comparison indicates that the simple linear model and the MLP are not able to describe the underlying dynamics of the FX-market appropriately.

As shown in Tab. 4.5, the multi-agent model is also superior to the basic unfolding in time RNN. This becomes especially obvious when we consider the calculated Sharpe Ratios, realized potentials and annualized returns. The performance measures obtained for the agent-based market are almost twice as high as those of the RNN.

Interestingly, the multi-agent approach and the standard ECNN (single agent) have a similar performance profile. Only the Sharpe ratio and the realized potential of the multi-agent model are slightly higher than

Table 8.3 Performance measures of the multi-agent model and the benchmarks. All performance criteria are calculated on the generalization set.

Performance Evaluation				
Model	*Hit Rate*	*Sharpe Ratio*	*Realized Potential*	*Annualized Return*
Multi-agent model	65.06%	0.5845	42.49%	13.48%
Error correction neural network (ECNN)	66.27%	0.5353	34.77%	13.74%
Recurrent neural network (RNN)	66.18%	0.3009	21.41%	7.70%
3-layer MLP	61.36%	0.1632	25.54%	4.69%
linear regression	57.95%	0.0652	21.03%	1.90%

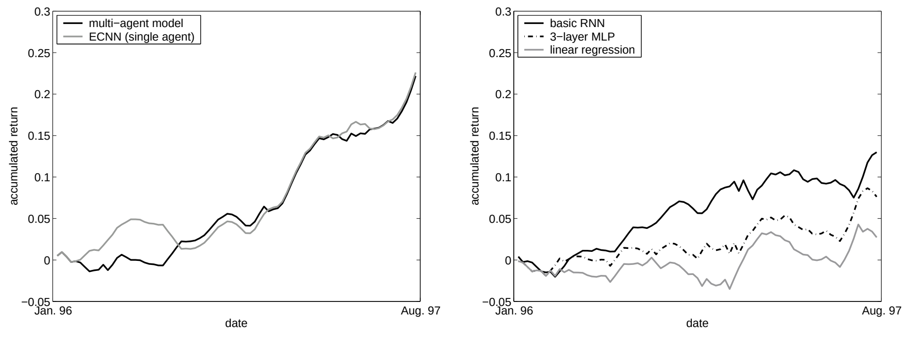


Figure 8.5 Out-of-sample accumulated return of the multi-agent model and the different benchmarks

those of the ECNN. From an econometric point of view, one may argue that the ECNN exploits all the information contained in the data such that an appropriate description of the FX-market dynamics is provided. In terms of multi-agent theory, we may say that the single agent (represented by the ECNN) has a rich internal model, which allows him to form accurate expectations about the future market development.

The agents in the market model show a similar behavior: Based on their price expectations and related actions, we are able to give a superior description of the underlying market dynamics. As indicated by the

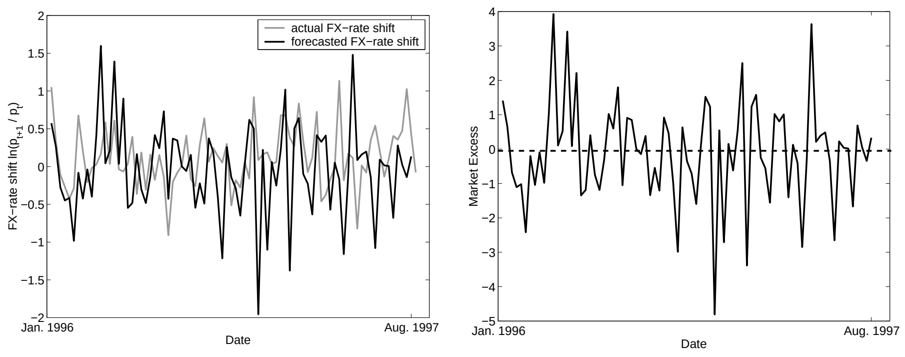


Figure 8.6 Actual weekly changes in the USD / DEM FX-rate and corresponding forecasts of the multi-agent model, left, and market excess observed in the multi-agent model, right. Recall, that the multi-agent model predicts changes in the FX-rate by scaling the observed market excess (see Eq. 8.5).

Sharpe ratio, the forecasts of the multi-agent model are more 'stable' than those of the single agent. We may conclude that this observation is due to an ensemble effect [NZ98, Perr93, p. 381-83]: In contrast to the single agent (ECNN), the forecast of the multi-agent model is based on the internal expectations and actions of a large number of agents. This 'averaging' may improve the prediction accuracy. Furthermore, the realized potential reveals that the multi-agent model is more capable of forecasting larger FX-rate movements than the single agent.

2. A DEDUCTIVE APPROACH TO COGNITIVE AGENTS

In this section we derive the three basic features (perception, internal processing and action) of the cognitive system *deductively* from the assumption of homeostasis [CMcV01, p. 11-17]. Homeostasis is a central property of a cognitive system. Given a changing environment, let us tentatively say that homeostasis can be seen as the attempt of the system to maintain an internal equilibrium (i. e. a dynamically stable state). The basic features constitute necessary conditions of a cognitive system [Cra43, RSMH86, RN95, p. 57, p. 40-4 and p. 13-4].

As a structural representation of homeostasis, we propose so-called zero-neurons within a time-delay recurrent neural network (see chp. 4). On this basis, we can also model the other basic features of the cognitive system. Our main interest is to describe the decision behavior of economic agents in a more realistic manner.

This section consists of *five* subsections: First, we derive the properties perception, internal processing and action of a cognitive system deductively from the assumption of homeostasis (sec. 2.1). Thereafter, we model the suggested cognitive system with a time-delay recurrent neural network (sec. 2.2).

In subsection 2.3 we deal with the identification of entities (so-called binding problem) [Vel95, Marsh95, p. 250 and p. 590]. Binding is a process of grouping coherent features into objects, i. e. "deciding which local features belong together as parts of an object and representing the parts of an object as coherent" [Marsh95, p. 590]. Our approach to the binding problem is based on the idea of variants-invariants separation [RHW86, CS97, p. 335-9 and p. 101-7].

In subsection 2.4 we introduce a multi-agent model, in which the decision behavior of the agents is modeled with homeostatic dynamic systems. The multi-agent model is based on time-delay recurrent neural networks (see chp. 4). As a remarkable property, our cognitive system based approach allows the fitting of real-world financial data. Subsection 2.5 deals with an empirical study modeling the USD / DEM FX-Market. Based on common performance measures (Sharpe ratio, hit rate, realized potential), it turns out that our multi-agent model is superior to several benchmarks (linear regression, MLP, basic RNN).

2.1 APPROACHING DEDUCTIVE COGNITIVE SYSTEMS

In this subsection we derive the suggested features 'perception', 'internal processing' and 'action' deductively from an important property of the cognitive system. That is homeostasis.

2.1.1 HOMEOSTASIS

Let us define homeostasis as a dynamically stable state, in which external influences (perception) are balanced with internal expectations such that an internal equilibrium is maintained [CMcV01, Har76, Koe67, Lan65, p. 11-17]. Since the cognitive system is *not* isolated from the environment, perception may disturb the internal stable state (behavior). In order to keep the internal equilibrium, the cognitive system is forced to develop an internal model and to initiate actions, which counterbalance the disturbances. Homeostasis can therefore be seen as a self-regulation mechanism of the cognitive system [CMcV01, p. 11-17].

It should be obvious, that the three properties perception, internal processing and action constitute necessary conditions for a cognitive system [Cra43, RN95, p. 57 and p. 13-4].

2.1.2 PERCEPTION

We distinguish between conscious and unconscious perception [Bar97, RN95, p. 411-9, p. 32-3]. Conscious perception means, that the cognitive system compares selective observations to a certain internal expectation. We may say that these external disturbances are balanced (or compensated) with internal expectations. Only the resulting error, i. e. the difference between the observation and the internal expectation, has an impact on the system [Met00, p. 112-3].

In contrast, unconscious perception is not balanced with internal expectations. Since there is no compensation with an internal expectation, unconscious perception is directly transferred into the cognitive system. This corresponds to stimulus response [RN95, Bar97, p. 202-3 and p. 411-8].

2.1.3 INTERNAL PROCESSING

The internal processing mainly consists of an internal model of the external world [RSMH86, RN95, LN77, p. 40-4, 203-4 and p. 589-95 and p. 404-9]. In order to maintain the internal equilibrium, homeostasis implies the construction of an internal model, which balances conscious perception with internal expectations. Without this balance, the system would be driven by stimulus-response.

For the construction of internal expectations, the system also requires an internal memory [RN95, p. 203-4]. Without a memory, the mapping of an internal dynamics is impossible. The memory can be seen as a merging of all relevant past information from which the internal expectation is formed [CS97, p. 312-3 and p. 317].

In case of missing or incomplete conscious perceptions, a homeostatic system relies on the internal expectations in order to supplement the imperfect observations [LN77, p. 366-77].

2.1.4 ACTION

Actions of the cognitive system are prepared by the internal processing [Cra43, p. 55-7]. More precisely, internal expectations are evaluated by a certain objective function, which depends on the preferences of the individual (e. g. utility or expected profit maximization). From this it follows, that actions are always goal-oriented.

In a homeostatic cognitive system perception and action are decoupled [CMcV01, p. 11-17]: incoming stimuli do not lead to actions if they do not disturb the internal equilibrium. In other words, homeostatic cognitive agents "are able to think about a problem without regard to circumstances unless the circumstances become demanding upon their attention" [Bar97, p. 405].

Due to the decoupling of perception and action, the behavior of a homeostatic cognitive system is driven by the abstract objective function rather than by a simple stimulus-response mechanism. Remarkably, "[s]uch independence is not found in a feedforward network [i. e. stimulus response system], where the input is progressively transmitted through stages to the output" [Bar97, p. 405].

Through the interaction with the environment, the system consciously perceives the impact of its own actions [Jua99, p. 25-7]. This is also a necessary condition for self-consciousness [Vel95, Met00, p. 247-50 and p. 78-9]. The result of the executed action is compared to the original intention [Jua99, p. 187-8]. A difference between the intended and the executed action may have an additional impact on the system.

Fig. 8.7 summarizes our considerations on homeostatic cognitive systems.

2.2 MODELING DEDUCTIVE COGNITIVE SYSTEMS

We pointed out, that all features of the cognitive system can be derived from homeostasis. The same applies for the structural representation of the cognitive system: by deriving an appropriate structural mapping of homeostasis, we are able to model all other feature of the system within an integrated structural framework.

This subsection is organized as follows: First, we point to a structural representation of homeostasis and show, how the other features can be embedded into this model. Thereafter, we translate our considerations into a recurrent neural network architecture.

2.2.1 A STRUCTURAL REPRESENTATION OF DEDUCTIVE COGNITIVE SYSTEMS

A structural representation of the homeostatic cognitive system is depicted in Fig. 8.8.

Homeostasis. We suggest to model homeostasis by so-called zero-neurons in a recurrent neural network framework. Zero-neurons are input-output neurons with constant, task invariant target values of 0 and input weights fixed at -1. Zero-neurons can be equipped with any common squashing and error function. However, due to methodical reasons (e. g. robust estimation of the system parameters), we propose

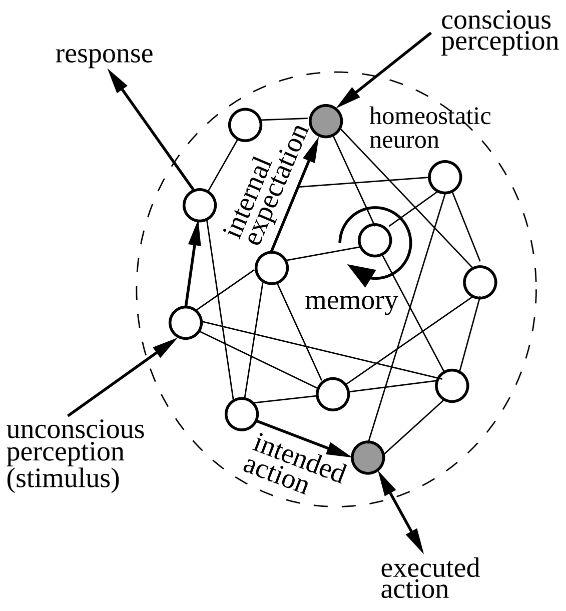


Figure 8.7 A homeostatic cognitive system. Conscious perception forces the system to build an internal model, which generates internal expectations as a compensation. Unconscious perception influences the cognitive system directly in the form of stimulus-response. The system memory, which is due to the recurrent formulation, is a merging of all relevant past information. A memory is crucial for a homeostatic system, unless there is another way to build an internal dynamics. The cognitive system is able to decouple perceptions and actions. Actions are not necessarily linked to stimulus-response, but are initiated in accordance with a certain objective function. The system compares the intended and the executed action. An observed difference has an additional impact on the system.

hyperbolic tangent squashing and ln cosh(.) cost functions [NZ98, p. 375-76 and p. 387-88].

A straightforward way to put the cognitive system into a homeostatic state would be to cut off *all* connections to external input signals. In this case, the cognitive system is completely isolated from the surrounding environment and a flat dynamical stable state would be obtained immediately. Nevertheless, this is only a trivial solution of homeostasis. If the cognitive system is *not* isolated, it has to balance the external disturbances in order to maintain an internal equilibrium.

Perception. Conscious perceptions y_τ^d are always connected to a zero-neuron. Due to the fixed input weights of the zero-neuron, a trivial solution of homeostasis by disconnecting the input y_τ^d is avoided. In

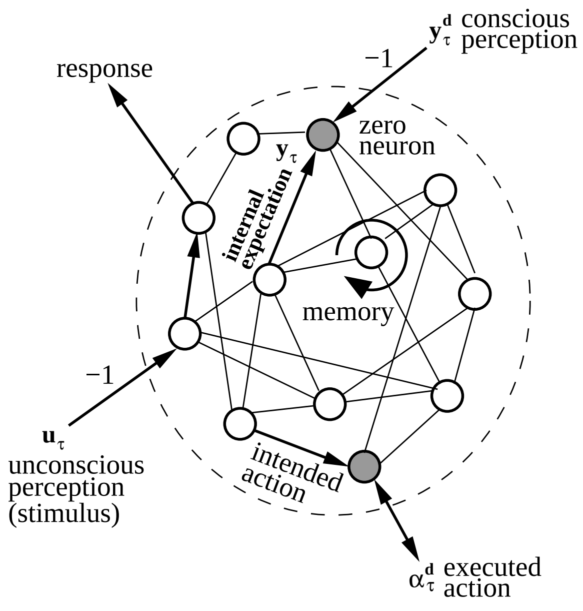

y^d_τ conscious perception
-1
response
zero neuron
y_τ
internal expectation
memory
-1
u_τ unconscious perception (stimulus)
intended action
α^d_τ executed action

Figure 8.8 Modeling of a homeostatic cognitive system. Conscious perceptions y^d_τ and actions α^d_τ are transferred into the system by fixed input weights of -1. Zero-neurons (marked gray) force the construction of internal expectations y_τ and intended actions α_τ. Intended actions are chosen via a higher-level objective function. Note, that intended actions α_τ may differ from the actual executed actions α^d_τ due to certain environmental conditions. Unconscious perceptions u_τ are directly transferred into the system. For simplicity, we also assume fixed input weights of -1 for these signals. The system memory is due to the recurrent formulation.

fact, the input y^d_τ induces an error flow, which forces the internal model to construct an expectation y_τ as a compensation. An error flow can only be avoided, if the internal expectation y_τ is equal to the external disturbance y^d_τ. Only the difference $y_\tau - y^d_\tau$ has an impact on the system. This matches the preceding definition of conscious perception [Met00, p. 112-3].

Unconscious perceptions u_τ are directly connected to the hidden neurons of the recurrent network. Hence, an error flow is not induced and the system is not forced to build internal expectations as a compensation. Nevertheless, unconscious perception u_τ has a direct impact on the cognitive system. This corresponds to the principle of stimulus-response [RN95, Bar97, p. 202-3 and p. 411-8].

Internal processing. The internal processing is structurally represented by a recurrent neural network similar to the one depicted in

Fig. 8.8. The connections among the neurons are the only tunable parameters. Due to the recurrent construction, the system is able to accumulate a superposition of all relevant past information, i. e. a memory. The memory is crucial for a homeostatic cognitive system, unless there is no other way to establish an internal dynamics. If we assume, that the hidden neurons depicted in Fig. 8.8 are not recursively connected to each other, we are back in the framework of feedforward neural networks. In this case, the zero-neurons behave like a stimulus response mechanism [Bar97, p. 405].

Action. In addition to the internal expectations y_τ, the cognitive system suggests actions α_τ. The cognitive system perceives the resulting state of the environment y_τ^d and the actually executed action α_τ^d. If the intended action α_τ cannot be executed, the observed action α_τ^d and the intention α_τ differ [Jua99, p. 187-8]. The observed difference $\alpha_\tau - \alpha_\tau^d$ may have an impact on the behavior of the cognitive system.

2.2.2 UNFOLDING IN TIME OF DEDUCTIVE COGNITIVE SYSTEMS

A concretization of the structural framework depicted in Fig. 8.8 is shown in Fig. 8.9. We propose to use a finite unfolding in time neural network to model the homeostatic cognitive system.

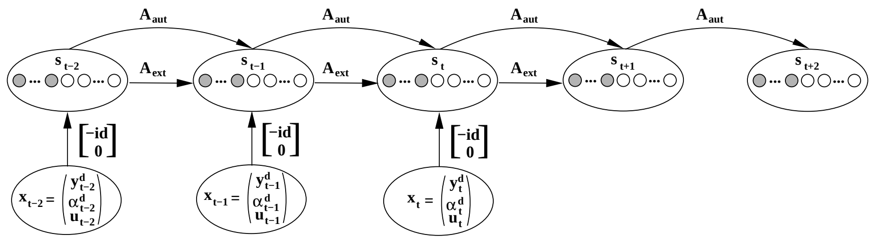

A_{aut}
A_{aut}
A_{aut}
A_{aut}
s_{t-2}
A_{ext}
s_{t-1}
A_{ext}
s_t
A_{ext}
s_{t+1}
s_{t+2}
$\begin{bmatrix}-id\\0\end{bmatrix}$
$\begin{bmatrix}-id\\0\end{bmatrix}$
$\begin{bmatrix}-id\\0\end{bmatrix}$
$x_{t-2} = \begin{pmatrix}y^d_{t-2}\\ \alpha^d_{t-2}\\ u_{t-2}\end{pmatrix}$
$x_{t-1} = \begin{pmatrix}y^d_{t-1}\\ \alpha^d_{t-1}\\ u_{t-1}\end{pmatrix}$
$x_t = \begin{pmatrix}y^d_t\\ \alpha^d_t\\ u_t\end{pmatrix}$

Figure 8.9 A structural representation of the homeostatic cognitive system shown in Fig. 8.8. The time-delay recurrent neural network incorporates zero-neurons to model homeostasis. An internal state s_τ of the network consists of zero-neurons (marked gray) and conventional (plain) hidden neurons. The recursive design of the network architecture is able to set up a memory. The shared weight matrices A_{ext} and A_{aut} include the only tunable parameters of the system. Matrix A_{aut} codes the autonomous part of the system, whereas A_{ext} codes the externally driven part. Note, that **-id** is a fixed negative identity matrix. All elements of matrix **0** are fixed at zero. The matrix **0** is used to adjust different dimensionalities between the input layer x_τ and the internal state s_τ.

Let us consider the structural representation of the homeostatic cognitive system depicted in Fig. 8.9: The internal state s_τ of the system consists of a sequence of zero-neurons (marked gray), followed by a series of ordinary hidden neurons (unmarked). The zero- and hidden neurons of the system are equipped with tanh squashing functions.

Perception. Conscious and unconscious perceptions are external inputs to the network. For convenience, we collected the different perceptions in the input layer x_τ.

Conscious perceptions y^d_τ are arranged in a sequence located in the first part of the input layer x_τ. Each conscious perception y^d_τ is connected to a separate zero-neuron. Remind, that the zero-neurons are located in the first part of the internal state s_τ. The input connections are fixed at 1 (see Fig. 8.9, matrix **-id**). Due to this construction, conscious perceptions y^d_τ induce an error flow, which forces the system to form internal expectations y_τ as a compensation. The system can only avoid an error flow, if the internal expectation y_τ is equal to the external disturbance y^d_τ, i. e. $y_\tau - y^d_\tau = 0$. As a side-effect of the fixed input weights -1, the internal expectations y_τ are a 'natural' picture of the related conscious perceptions y^d_τ.

Unconscious perceptions u_τ are directly connected to the hidden neurons of the recurrent network. In mind that the sequence of hidden neurons is located in the back part of the internal state (see Fig. 8.9), we arranged the unconscious perceptions u_τ accordingly in the input layer x_τ. Each external input u_τ is only linked to a single hidden neuron. The input weights are fixed at -1. Unconscious perceptions u_τ do not induce an error flow, because they are only processed by conventional hidden neurons. Consequently, the system is not forced to generate internal expectations. Nevertheless, unconscious perceptions u_τ may have a direct impact on the system.

Action. Besides the internal expectations y_τ, the cognitive system suggests actions α_τ. These actions can be seen as intentions of the cognitive system. The system compares the executed actions α^d_τ to its intentions α_τ. In case of a difference between the intended and the executed action, an error flow is induced, which may influence the system in the future.

The observed actions α^d_τ are arranged in a sequence adjacent to the conscious perceptions y^d_τ (see Fig. 8.9, input layer x_τ). Executed actions α^d_τ are connected to separate zero-neurons. Hence, the system is able to measure the difference between an intention α_τ and the observed action α^d_τ. An error flow is induced, if the executed action α^d_τ and the intention α_τ differ, i. e. $\alpha_\tau - \alpha^d_\tau \neq 0$.

As an example, consider a cognitive agent trading on a stock market. A market order of the agent is represented by his intended action α_τ.

If the market order is filled, the intention and the executed action are equal, i. e. $\alpha_\tau - \alpha_\tau^d = 0$. If the market order is *not* filled, the agent's intention α_τ and the executed action α_τ^d differ, i. e. $\alpha_\tau - \alpha_\tau^d \neq 0$. This may have an impact on the behavior of the agent.

For all future time steps of the unfolding $t+\tau$, we have no observables for the conscious perceptions $y_{t+\tau}^d$ and executed actions $\alpha_{t+\tau}^d$ per definition. Thus, the system offers forecasts $y_{t+\tau}$ and goal-oriented intentions $\alpha_{t+\tau}$. By the application of overshooting (see chp. 4), the system may be enforced to construct a series of forecasts $y_{t+\tau}$ and intentions $\alpha_{t+\tau}$. Overshooting means that we iterate the autonomous part of the system into future direction. The network architecture shown in Fig. 8.9 codes the autonomous part in matrix A_{aut}. Hence, we may add overshooting time steps by the reapplication of matrix A_{aut}.

Dimensionality of the internal state. Technically, we have to guarantee an appropriate dimension $\dim(s)$ of the internal state s_τ. For each conscious perception y_τ^d *and* action α_τ^d we need a single zero-neuron. In addition, each unconscious perception u_τ requires an additional hidden neuron.

Assuming, that we have a number of p conscious perceptions, q actions and r unconscious perceptions, the dimension $\dim(s)$ of the internal state is $\dim(s) = p + q + r$. A more convenient formulation is obtained by defining that the dimension $\dim(s)$ of the internal state is equal to the dimension of the input layer x_τ ($\dim(x) = p + q + r$). Note, that the internal state s_τ consists of $p + q$ zero-neurons and r hidden neurons.

To support the internal processing, we propose to add additional hidden neurons to the back part of the internal state. Suppose, that we enlarge the internal state by a number of v hidden neurons. As a consequence, the dimension $\dim(s)$ of the internal state is larger than the dimension $\dim(x)$ of the input layer. In this case, we have to ensure that the connector between the input layer x_τ and the internal state s_τ is of an appropriate size. Hence, we add a number of fixed zero connections, which corresponds to the number of additional hidden neurons (see Fig. 8.9, matrix $\mathbf{0}$).

Internal structure of matrices A_{ext} and A_{aut}. Let us now focus on the internal structure of the shared transition matrices A_{aut} and A_{ext} (see also Fig. 8.9). The matrices are independent from one another and include the only tunable parameters of the system. Matrix A_{aut} codes the autonomous part of the system, whereas A_{ext} codes the externally driven part. The internal structure of the matrices A_{aut} and A_{ext} is depicted in Fig. 8.10.

$$s_\tau \leftarrow \overset{\displaystyle s_{\tau-1}}{\begin{array}{c|ccc} & y_{\tau-1}-y^d_{\tau-1} & \alpha_{\tau-1}-\alpha^d_{\tau-1} & \text{internal processing}_{\tau-1} \\ \hline y_\tau - y^d_\tau & 0 & 0 & 0 \\ \alpha_\tau - \alpha^d_\tau & 0 & 0 & 0 \\ \text{internal processing}_\tau & D_2 & D_1 & 0 \end{array}} \qquad s_\tau \leftarrow \overset{\displaystyle s_{\tau-1}}{\begin{array}{c|ccc} & y_{\tau-1}-y^d_{\tau-1} & \alpha_{\tau-1}-\alpha^d_{\tau-1} & \text{internal processing}_{\tau-1} \\ \hline y_\tau - y^d_\tau & 0 & 0 & C_2 \\ \alpha_\tau - \alpha^d_\tau & 0 & 0 & C_1 \\ \text{internal processing}_\tau & 0 & 0 & \dot{A} \end{array}}$$

$$\text{Matrix } A_{ext} \qquad\qquad\qquad\qquad \text{Matrix } A_{aut}$$

Figure 8.10 Internal structure of the shared transition matrices A_{aut} and A_{ext}

As shown in Fig. 8.10, matrix A_{ext} consists of the sub-blocks D_1 and D_2. Matrix A_{aut} is composed of the sub-blocks C_1, C_2 and $\dot{A}$. The sparse internal structure of the matrices A_{aut} and A_{ext} results from the formal independence of conscious perceptions y^d_τ, internal expectations y_τ, executed actions α^d_τ, intentions α_τ and unconscious perceptions u_τ. Let us consider the different sub-blocks in greater detail.

The internal processing depends on the previous internal state $(\dot{A}s_{\tau-1})$. Furthermore, the observed differences $y_{\tau-1} - y^d_{\tau-1}$ and $\alpha_{\tau-1} - \alpha^d_{\tau-1}$ may have an impact on the internal processing through $D_2(y_{\tau-1} - y^d_{\tau-1})$ and $D_1(\alpha_{\tau-1} - \alpha^d_{\tau-1})$. Remind, that the internal processing is also influenced by unconscious perceptions u_τ, which enter the internal state directly (see Fig. 8.9, input matrix $-id$).

The system generates internal expectations $y_\tau = C_2 s_{\tau-1}$ and intended actions $\alpha_\tau = C_1 s_{\tau-1}$. The internal expectations $y_\tau = C_2 s_{\tau-1}$ are used for the compensation of conscious perceptions y^d_τ (see Fig. 8.9, input matrix $-id$). The intended actions $\alpha_\tau = C_1 s_{\tau-1}$ are compared to the executed actions α^d_τ. Recall, that these comparisons take place at the zero-neurons of the internal state. In the future $(t+\tau)$, we have no compensation of the internal expectations $y_{t+\tau}$ and $\alpha_{t+\tau}$. Hence, the system offers forecasts $y_{t+\tau}$ and intended actions $\alpha_{t+\tau}$.

For all future time steps of the unfolding, the cognitive system has no information about the differences $y_{t+\tau} - y^d_{t+\tau}$ and $\alpha_{t+\tau} - \alpha^d_{t+\tau}$. Hence, the system has to assume that the internal model is correct. Consequently, matrix A_{ext} coding the externally driven part is skipped. This corresponds to the functioning of the cognitive system depicted in Fig. 8.3.

In future time this system skips the matrices D_1 and D_2 for the same reasons.

Let us summarize the preceding considerations by formulating a system of equations that describes the behavior of the homeostatic cognitive system (see Fig. 8.9 and 8.10). In matrix notation the homeostatic cognitive system can be written as

$$\begin{pmatrix} y_\tau \\ \alpha_\tau \\ s_\tau \end{pmatrix} - \begin{pmatrix} y_\tau^d \\ \alpha_\tau^d \\ \begin{bmatrix} u_\tau \\ 0 \end{bmatrix} \end{pmatrix} = \tanh\left(\mathbf{A} \left[\begin{pmatrix} y_{\tau-1} \\ \alpha_{\tau-1} \\ s_{\tau-1} \end{pmatrix} - \begin{pmatrix} y_{\tau-1}^d \\ \alpha_{\tau-1}^d \\ \begin{bmatrix} u_{\tau-1} \\ 0 \end{bmatrix} \end{pmatrix} \right] \right) , \quad (8.6)$$

$$\text{with} \quad \mathbf{A} = \mathbf{A_{ext}} + \mathbf{A_{aut}} = \begin{pmatrix} 0 & 0 & C_2 \\ 0 & 0 & C_1 \\ D_2 & D_1 & \dot{A} \end{pmatrix} .$$

As shown in Eq. 8.6, the internal processing s_τ is a mapping from the previous state $s_{\tau-1}$ (incorporating the influence of former external disturbances $u_{\tau-1}$), actual external influences u_τ and the observed differences $(y_{\tau-1} - y_{\tau-1}^d)$ and $(\alpha_{\tau-1} - \alpha_{\tau-1}^d)$. Remind, that all input signals are transferred into the system by fixed input weights of -1. We adjust different dimensionalities between the internal state s_τ and the external influences u_τ by $[u_\tau, 0]'$. Internal expectations y_τ are computed by $y_\tau = C_2 s_{\tau-1}$, whereas intended actions α_τ are derived by $\alpha_\tau = C_1 s_{\tau-1}$. In future time $t + \tau$, there is no compensation for the internal expectations $y_{t+\tau}$, $\alpha_{t+\tau}$ and thus the system offers forecasts $y_{t+\tau}$, $\alpha_{t+\tau}$. Note, that we skip matrix A_{ext} for all future time steps.

Interestingly, the internal design of matrices A_{aut} and A_{ext} (Fig. 8.10) bears resemblance to the structure of the cognitive system depicted in Fig. 8.3. For convenience, the notations in Fig. 8.10 and Fig. 8.3 have the same meaning. As an example, let us consider the formation of internal expectations y_τ. In the ECNN (Fig. 8.3) internal expectations are derived by $y_\tau = C_2 s_\tau$. A similar functioning can be found in matrix A_{aut} (Fig. 8.10). Here, internal expectations y_τ are generated by $y_\tau = C_2 s_{\tau-1}$.

The homeostatic cognitive system (resp. its structural representation in Fig. 8.9) models the decision behavior of a single agent trading on a financial market. The intended actions α_τ represent the market orders of the agent. The agent consciously perceives the market development and observes other external influences u_τ unconsciously.

2.3 IDENTIFICATION OF ENTITIES: BINDING

The identification of entities (so-called binding problem) is a process of grouping coherent features into objects [HMcCR86, Vel95, vdMS, p. 88-90 and p. 250]. As an example let us refer to visual processing [Marsh95, p. 590]. Here, we have to identify distinct features that cohere as a specific object. Furthermore, we have to separate different objects from one another and from the background. In other words, the binding problem involves two tasks: "[*i.*] deciding which local features belong together as parts of an object and [*ii.*] representing the parts of an object as coherent" [Marsh95, p. 590].

The underlying problem is to discover techniques or criteria, which can be used for the identification and categorization of entities. For instance, one may search for structural similarities or define more abstract binding criteria to solve the task. We try to solve the binding problem by a variants-invariants separation in form of a bottleneck (autoassociator) neural network [RHW86, CS97, p. 335-9 and p. 101-7]. Our interest is on the prediction of high-dimensional dynamical systems. We want to identify distinct structures within the high-dimensional dynamics, which in turn enable us to simplify the forecast problem [ZN01, p. 341-2].

This section is organized as follows: First we introduce the basic idea of variants-invariants separation (see also chp. 4). A suitable coordinate transformation in form of a bottleneck neural network allows us to separate time variant from invariant structures. Thereafter, we integrate the variants-invariants separation into the cognitive system depicted in Fig. 8.9. This is done by a redesign of the shared transition matrices A_{ext} and A_{aut}.

2.3.1 VARIANTS-INVARIANTS SEPARATION

Consider a high-dimensional dynamic system. The complexity of such a system can be reduced, if we are able to separate the dynamics into time variant and invariant structures [ZN01, p. 341]. The dimensionality of the time variants should be lower than that of the complete dynamic system.

By a variants-invariants separation, we also simplify the prediction of the high-dimensional dynamics: Only the variants of the system have to be forecasted, while the invariants remain constant over time. The recombination of predicted variants and unchanged invariants provides us with a forecast of the original dynamics [ZN01, p. 341]. In the broadest sense, the identified variants can be interpreted as the principle components of the high-dimensional dynamics. The basic principle of variants-invariants separation is illustrated in Fig. 8.11 [ZN01, p. 342].

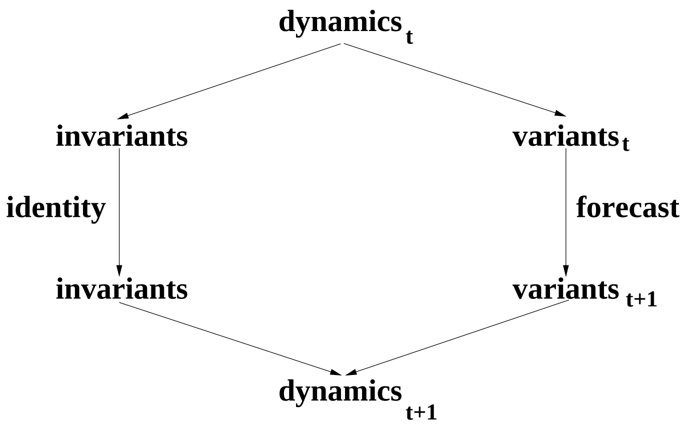


Figure 8.11 Variants-invariants separation of a high-dimensional dynamical system

The concept of variants-invariants separation is especially useful in economic applications. As an example, let us consider a yield curve consisting of ten or more interest rates for different maturities. We can simplify the prediction task by extracting a small number of variants out of the yield curve dynamics (e. g. in case of the German yield curve only 3 variants are important [ZNG00c, p. 266]). The separation of time variant from invariant structures can be seen as a principle component analysis (PCA) of the yield curve dynamics. After we have predicted the variants, a recombination with the invariants gives rise to the future development of the complete yield curve [ZNG00c, p. 265-7].

2.3.2 VARIANTS-INVARIANTS SEPARATION BY NEURAL NETWORKS

The variants-invariants separation can be modeled by a coordinate transformation, which translates the high-dimensional space of observed state representations Y into a low-dimensional inner state space S (i. e. $\dim(Y) > \dim(S)$) [ZN01, p. 341]. The general description of the variants-invariants separation can be stated without referring to neural networks. Nevertheless the realization is a typical neural network approach due to its property to concatenate functional systems [ZN01, p. 342]. Fig. 8.12 depicts a coordinate transformation in form of a neural network architecture [RHW86, Spe00, CS97, ZN01, p. 335-9, p. 121-4, p. 101-7 and p. 341-2].

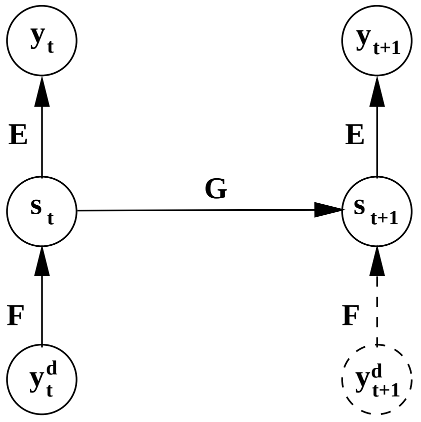


Figure 8.12 Variants-invariants separation by a bottleneck (autoassociator) neural network

Let us consider the network architecture shown in Fig. 8.12. The internal state variables s_t, s_{t+1} play the role of the variants. The dimensionality of the internal state is therefore smaller than the dimensionality of the input vector y^d ($\dim(s) < \dim(y^d)$). The compressor matrix F separates the dynamics into variants and invariants. The decompressor matrix E combines the forecast of the variants with the invariants to reconstruct the complete dynamics. Matrix G is used to forecast the internal state variables s_t [ZNG01a, p. 341-2].

2.3.3 COGNITIVE SYSTEMS & BINDING OF ENTITIES

In the context of binding, we may refer to the identified invariant structures as entities. In this sub-section we integrate the binding of entities (in form of a variants-invariants separation) into the homeostatic cognitive system depicted in Fig. 8.9. Our approach is based on a redesign of the shared transition matrices A_{aut} and A_{ext}. As we will explain, the variants-invariants separation is modeled through a coordinate transformation which is implicitly included in the restructured matrices. The new internal structure of the matrices A_{aut} and A_{ext} is depicted in Fig. 8.13.

Suppose, that the dimension of the observed conscious perceptions y^d_τ is higher than that of the internal state (i. e. $\dim(y^d) > \dim(s)$). Such an arrangement implies a coordinate transformation. The compression (i. e. the identification of the variants) is done by the sub-block F (see matrix A_{ext}), whereas the decompression (i. e. the recombination

$$
\begin{array}{c|ccc|}
 & y_{\tau-1}-y^d_{\tau-1} & \alpha_{\tau-1}-\alpha^d_{\tau-1} & \text{internal processing}_{\tau-1} \\
\hline
y_\tau - y^d_\tau & 0 & 0 & 0 \\
\alpha_\tau - \alpha^d_\tau & 0 & 0 & 0 \\
\text{internal processing}_\tau & F & D & 0 \\
\hline
\end{array}
\qquad
\begin{array}{c|ccc|}
 & y_{\tau-1}-y^d_{\tau-1} & \alpha_{\tau-1}-\alpha^d_{\tau-1} & \text{internal processing}_{\tau-1} \\
\hline
y_\tau - y^d_\tau & 0 & 0 & E \\
\alpha_\tau - \alpha^d_\tau & 0 & 0 & C \\
\text{internal processing}_\tau & 0 & 0 & \dot{A} \\
\hline
\end{array}
$$

$$\overbrace{\qquad\qquad}^{s_{\tau-1}} \qquad s_\tau \leftarrow \left\{ \qquad\qquad \right.$$

Matrix A_{ext} Matrix A_{aut}

Figure 8.13 Integrating a variants-invariants separation into the homeostatic cognitive system by a redesign of the transition matrices A_{aut} and A_{ext}. Recall, that matrix A_{aut} codes the autonomous part of the system, whereas A_{ext} codes the externally driven part.

of variants and invariants) is done by the sub-block E (see matrix A_{aut}). The sub-blocks F and E correspond to the compressor-decompressor matrices of the bottleneck network shown in Fig. 8.12.

Remind, that the internal state of the system depends on the previous internal state ($\dot{A}s_{\tau-1}$), the observed differences $y_\tau - y^d_\tau$ and $\alpha_\tau - \alpha^d_\tau$, and the external disturbances u_τ. Conscious perceptions y^d_τ, executed actions α^d_τ and external influences u_τ are transferred into the system by the fixed matrix **-id** (see Fig. 8.9).

The cognitive system generates internal expectations y_τ about the conscious perceptions y^d_τ through $Es_{\tau-1}$. Intended actions α_τ are derived by $Cs_{\tau-1}$. The internal expectations y_τ and α_τ are compared to the observables y^d_τ and α^d_τ. The resulting differences enter the system through $F(y_\tau - y^d_\tau)$ and $D(\alpha_\tau - \alpha^d_\tau)$.

In the future the system has to assume that the internal model is correct, because the error corrections $y_{t+\tau} - y^d_{t+\tau}$ and $\alpha_{t+\tau} - \alpha^d_{t+\tau}$ are missing per definition. As a consequence, we skip the matrix A_{ext} coding the externally driven part of the system. Since a compensation of the internal expectations is not available, the cognitive system suggests forecasts $y_{t+\tau}$ and intended actions $\alpha_{t+\tau}$.

2.4 FX-MARKET MODELING BY DEDUCTIVE COGNITIVE AGENTS

Our intend in this subsection is to build a multi-agent model of a FX-market which is based on the preceding considerations about homeostatic cognitive systems. We embed the cognitive systems of the agents trading on the market and the related market price formation mechanism into a neural network architecture. The trading scheme of the agents and the price formation mechanism are described in section 1.3. The decision making behavior of each agent is modeled by a homeostatic cognitive system as shown in Fig. 8.9. The resulting multi-agent model is depicted in Fig. 8.14.

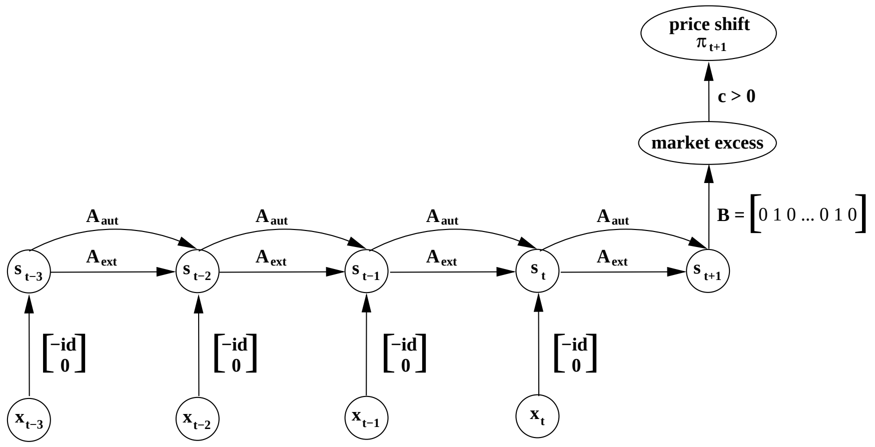


Figure 8.14 Multi-agent market modeling based on homeostatic cognitive agents. The homeostatic cognitive systems of the agents (Fig. 8.9) are integrated into a higher-level neural network architecture. The separation of the agents' cognitive systems is realized by block-diagonal matrices. Note, that A_{aut}, A_{ext}, **-id** and **0** are matrices of appropriate sizes. The block-design implies that the only interaction among the agents is due to the market price formation mechanism. Changes in the FX-rate can be seen as a response of the market to an observed market excess (i. e. demand - supply).

The multi-agent model shown in Fig. 8.14 includes the homeostatic cognitive systems of *all* agents (see Fig. 8.9) and the market price formation mechanism (see Eq. 8.5). Let us consider the different elements of the network architecture in greater detail.

2.4.1 ARRANGEMENT OF THE AGENTS

The decision behavior of a single agent is modeled by a homeostatic cognitive system. The structural representation of the cognitive system is given by the recurrent neural network depicted in Fig. 8.9. The agent consciously perceives changes in the FX-rate and observes other external influences u_τ unconsciously. The market orders of the agent are mapped by the intended actions α_τ. The agent's objective function is expected profit maximization (see Eq. 8.2).

The network architecture shown in Fig. 8.14 separates the agents by block diagonal matrices. In other words, the agents exist in parallel without influencing each other. The only interaction among the agents is through the market price formation mechanism.

A state s_τ incorporates the zero- and hidden neurons of *all* agents. All input connections are fixed at -1 (Fig. 8.9, input matrix $-id$). The block diagonal matrices A_{aut} and A_{ext} include the only tunable parameters of the agents' cognitive systems. Matrix A_{aut} codes the autonomous part of the cognitive systems, whereas A_{ext} reflects the externally driven part (see Fig. 8.10). A single block in each matrix refers to one of the agents $a = 1, \ldots, A$. For instance, agent $a = 1$ is modeled by the first block in *each* matrix, whereas agent $a = A$ is represented by the last block in each matrix. Of course, this design also implies that the internal state s_τ is constructed appropriately.

Heterogeneity among the agents arises from different information sets [BMT96, WHD02, p. 178 and p. 89-92]. More precisely, we spread the external influences u_τ among the agents. Technically, we randomly choose weight connections of unconscious perceptions u_τ from the input matrix $[-id, 0]$ and turn the fixed input weights to zero. Further details on heterogeneous decision making can be found in chapters 6 and 7.

2.4.2 MARKET PRICE FORMATION

Changes in the FX-rate are performed as a response to the observed market excess (i. e. net of the agents' market orders) [Far98, LeB01a, p. 7-9 and p. 256]. The price formation mechanism can be described in two steps: First, the market orders α_{t+1} of the agents are collected via the fixed connector B. The design of matrix B corresponds to the internal state of the agents (see Fig. 8.9). Recall, that the action variables α_τ are located adjacent to the conscious perceptions. The summation of the orders α_{t+1} through matrix B yields the market excess. The 'market excess' cluster is a hidden layer with identity activation function.

In the second step, the market clearing FX-rate shift π_{t+1} is estimated by scaling the market excess with a positive constant $c > 0$ (see Eq. 8.5).

The 'price shift' layer is designed as an output cluster. We apply the robust error function $\ln\cosh$ (see Eq. 5.9) and use observed FX-rate shifts as target values.

The multi-agent model is trained with error backpropagation through time and *vario-eta* learning rule [RHW86, NZ98, p. 354-7 and p. 396].

2.5 EMPIRICAL STUDY

One of the most remarkable properties of our multi-agent approach is the fitting of real-world financial data. In this section we apply the multi-agent model depicted in Fig. 8.14 to the US-Dollar / German Mark FX-market. The prediction accuracy of the multi-agent model is evaluated in a benchmark comparison. The next paragraphs outline the design of the empirical study and describe the obtained results.

2.5.1 OUTLINE OF THE EMPIRICAL STUDY

The primary purpose of the empirical study is to illustrate the prediction accuracy of the multi-agent model (Fig. 8.14) in contrast to that of several benchmarks. We forecast the weekly changes in the USD / DEM FX-rate. Our experimental set-up is **equal** to that used in section 1.5. Let us briefly reconsider the basic outline of the empirical study.

Data and input signals. We work on the basis of weekly data to forecast the weekly changes in the USD / DEM FX-rate. The complete data set is from Jan. 91 to Aug. 97. The training set ranges from Jan. 91 to Dec. 95, whereas the generalization set covers the time period from Jan. 1996 to Aug. 1997.

The raw input time series used for our multi-agent model and the different benchmarks are summarized in Tab. 8.1. We calculated the scaled momentum (Eq. 2.7) and force (Eq. 2.8) of each raw input time series [NZ98, PDP00, p. 374 and p. 73].

Multi-agent model & benchmarks. Our experiments are concerned with the cognitive system based multi-agent model depicted in Fig. 8.14. In this application, the model incorporates a number of 200 agents. The cognitive system of each agent is modeled by a homeostatic cognitive system as shown in Fig. 8.9. The market model separates the agents by block diagonal matrices.

The agents perceive the development of the USD / DEM FX-market consciously and are unconsciously influenced by the external disturbances u_τ (see Tab. 8.1). Heterogeneous decision behavior is introduced by spreading the available information among the agents. To spread the information, we randomly disconnect unconscious perceptions from the agents' cognitive systems by turning the related input connections to zero (see Fig. 8.9, input matrix $[-id, 0]$). Previous experiments indicated, that the average size of the agents' data sets should be limited to approx. 8 external influences u_τ (see chp. 7).

The forecast of the USD / DEM FX-rate shift (see Fig. 8.14, 'price shift' layer) is used for the benchmark comparison. In other words, the forecast results from the trading activities of the different agents. We trained the multi-agent model until convergence using backpropagation through time in combination with *vario-eta* learning-rule [NZ98, Hay94, p. 354-7 and p. 751-4].

We compare the multi-agent model to several benchmarks: First of all, we refer to a single homeostatic cognitive agent (Fig. 8.9). As a second benchmark, we utilize the multi-agent model depicted in Fig. 8.4. Third, we compare our multi-agent approach to another cognitive agent. The cognitive system of this agent is represented by the ECNN depicted in Fig. 8.2. Furthermore, we refer to *three* econometric benchmarks: (*i.*) a basic time-delay recurrent network (see Fig. 4.4), (*ii.*) a 3-layer feedforward neural network (MLP, see Fig. 2.8) and (*iii.*) a linear regression model [PDP00, p. 201-11].

The homeostatic cognitive agent (Fig. 8.9) unconsciously perceives all external influences u_τ and observes the FX-market development consciously. We use the agent's internal expectation about the USD / DEM FX-rate shift for the benchmark comparison. The experimental set-up of the other cognitive system based multi-agent model (Fig. 8.4) is described in section 1.5. The design of the ECNN (single agent, Fig. 8.2) and the econometric benchmarks is summarized in Tab. 8.2.

Performance measures. We evaluate the prediction accuracy of the different models on the basis of *five* common performance measures: (*i.*) hit rate, (*ii.*) Sharpe ratio, (*iii.*) accumulated return of investment, (*iv.*) realized potential and (*v.*) annualized return [Ref95, p. 70-74]. All performance indicators are calculated on the generalization set (Jan. 96 – Aug. 97). Further details on the performance measures can be found in section 1.5.

2.5.2 RESULTS OF THE EMPIRICAL STUDY

In the following we come up with the results of our empirical study. The performance measures calculated for the different models on the out-of-sample period (Jan. 96 – Aug. 97) are summarized in Tab. 8.4. In Fig. 8.15 we compare the accumulated return of the multi-agent model to the benchmarks. Note, that the results obtained for the different benchmarks are equal to those reported in section 1.5.

Table 8.4 Out-of-sample performance measures calculated for the multi-agent model and the different benchmarks

Performance Evaluation				
Model	*Hit Rate*	*Sharpe Ratio*	*Realized Potential*	*Annualized Return*
Deductive multi-agent model (Fig. 8.14)	66.27%	0.5499	34.87%	13.83%
Single agent (Fig. 8.9)	64.46%	0.4932	28.60%	12.87%
Inductive multi-agent model (Fig. 8.4)	65.06%	0.5845	42.49%	13.48%
Single agent (ECNN, Fig. 8.2)	66.27%	0.5353	34.77%	13.74%
Recurrent neural network (RNN)	66.18%	0.3009	21.41%	7.70%
3-layer MLP	61.36%	0.1632	25.54%	4.69%
linear regression	57.95%	0.0652	21.03%	1.90%

The performance measures indicate, that our multi-agent model (Fig. 8.14) is superior to the econometric forecast methods (linear regression, MLP, basic RNN). This becomes especially apparent, if we look at the calculated Sharpe ratios. The Sharpe ratio of the deductive multi-agent model is by a factor of 1.8 higher than that of the best econometric model (basic RNN). We may conclude, that our multi-agent model (Fig. 8.14) allows a good description of the FX-market dynamics, while the forecasts of the econometric methods are more unreliable.

Comparing the performance profile of the deductive multi-agent model to those of the single agents (Fig. 8.2 and 8.9), we find that the market model slightly outperforms both agents. We may say that the predictions of the multi-agent model are more 'stable' than those of the single agents. The calculated Sharpe ratios may serve as an evidence for this claim. – Recall, that the forecasts of the multi-agent model are composed of the expectations of different agents. Hence, one may argue that the slightly higher Sharpe ratio of the multi-agent model is due to an ensemble

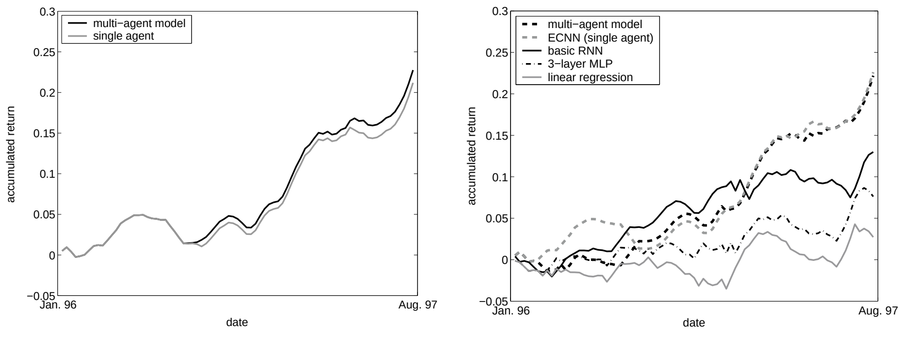


Figure 8.15 Out-of-sample accumulated return of the multi-agent model and the different benchmarks

effect [NZ98, Perr93, p. 381-83]. However, all models achieve a superior description of the FX-market dynamics by exploiting all information contained in the data.

As shown in Tab. 8.4, the multi-agent models (Fig. 8.4 and 8.14) have similar performance profiles. This is not astonishing, because both models can be seen as two sides of the same coin: In the first model (Fig. 8.4) we describe the features of the cognitive system inductively. The second model (Fig. 8.14) derives the features deductively from the assumption of homeostasis. The similar behavior of both approaches is also indicated by the performance of the single agent models (Fig. 8.2 and 8.9). For instance, the out-of-sample accumulated return of the two agents is nearly identical (Fig. 8.15).

3. PRELIMINARY CONCLUSION

We have developed two multi-agent FX-market models based on the idea of an elementary cognitive system. As a most remarkable property of our multi-agent models, we are in a position of explaining market prices on the basis of the decision processes of various market participants. Our approaches emphasize semantic specifications instead of ad-hoc functional relationships [EM01, p. 759-61]. We utilized time-delay recurrent neural networks, to merge multi-agent theory, cognitive systems and econometrics into an integrated framework of market modeling.

The decision making of the agents is modeled as a cognitive process with three basic features: perception, internal processing and action. In the first approach, we inductively described the features of the cognitive system. The structural integration of the proposed cognitive system into an error correction neural network (ECNN) turned out to be 'natural'.

Second, we derived the basic features of the cognitive system deductively from the assumption of homeostasis. Homeostasis is an internal equilibrium, in which external influences are compensated with internal expectations. Thus, the cognitive system overcomes poor stimulus-response mechanisms. As a structural representation of homeostasis, we introduced zero-neurons within a time-delay recurrent neural network.

Both multi-agent approaches allow the fitting of real-world data. Our experiments with the US-Dollar / German Mark FX-market indicated, that the proposed multi-agent models are superior to several benchmarks (e. g. linear regression, standard 3-layer MLP, basic RNN). The benchmark comparison is based on an out-of-sample period of one and a half years. We applied common performance measures (e. g. Sharpe ratio, realized potential, hit rate) to evaluate the forecast accuracy of the models.

As a future direction of research, one may try to enrich the behavior of the agents by the integration of utility functions. The concept of utility is well-known in the economic theory of choice [EG95, Weid00, SvR96, p. 4-9, p. 204-5 and p. 256-265]. Utility functions allow us to give a detailed analysis of the agents' behavior and goals. By this, long-term forecasting and strategic planning should be supported. In addition, one may think of more realistic market price formation mechanisms.

Chapter 9

SUMMARY AND CONCLUSION

The internal dynamics of financial markets results from the interaction of many agents optimizing their utility. In other words, the macroeconomic market price dynamics can be captured by building a superposition of the microeconomic decision-taking processes of the agents. Thus, a natural way of explaining and even predicting market prices is to analyze the decision making of the agents on the microeconomic level and to study their interaction on the macroeconomic side of the market. Typically, such a multi-agent approach works well, if a single decision can be attributed to only a small number of qualitative influences. Or if the number of agents is very small. However, if additional quantitative aspects are included into the modeling or the market size grows, this way becomes more and more intractable, because the degrees of freedom increase rapidly. As a consequence, the bridge between the microeconomic level and the macroeconomical price formation is weak, and it is often not possible to forecast of real-world price movements.

Alternatively, the internal dynamics of financial markets may be captured or forecasted by referring to quantitative variables and relative straightforward inter-variable relationships. For instance, one may apply a linear regression analysis to model the (linear) dependencies between the market price returns and a set of external influences. In this case, one does not directly take into account the decision-taking processes of the agents. Instead it is assumed that a certain kind of return generating process is responsible for the observed market price movements. In other words, one abstracts from the fact that market prices originate from the buying and selling decisions of many agents and tries to describe the price formation by an ad-hoc functional relationship. Since the presumed functional relationship can be attributed to the agents' behavior, the employed data generating process can be seen as an abstract representation of the underlying decisions and interactions of the agents. Although such an econometric approach has not the explanatory power of an agent-based model, one has the chance to integrate a lot of quantitative dynamics into the model building. Thus, the fitting of real-world price movements is usually possible.

Our Vision

The vision of this work was to establish a crosslink between explanatory agent-based models of financial markets and econometric techniques, which allow a quantitative identification and prediction of the market price dynamics. For this purpose, we applied feedforward and time-delay recurrent neural networks, which allow the fitting of high dimensional nonlinear models and thus, represent an appropriate framework for the modeling of complex economical systems. Using neural networks for the integration of the micro-economic decisions into one macro-economic market model, we are able to do both, capturing the underlying market dynamics and fitting real-world financial data. Most remarkably, our multi-agent models give us a rise to the dynamics of real-world financial markets from the perspective of single agents interacting with other market participants.

To realize this vision of an agent-based financial market on the basis of neural networks, this thesis is composed of *seven* main chapters. Chapters 2 to 5 are written from an econometric point of view and introduce the neural network framework that is used to develop our multi-agent models. Chapters 6 to 8 focus on the economic theory of agent-based financial markets. Here, we deal with recent developments in this new interdisciplinary research field and show, how neural networks can be utilized for the modeling of the agents' decision making schemes.

More precisely, chapter 2 gives a basic introduction to neural networks and outlines important developments within this research field. Among other things, we focus on 3-layer feedforward neural networks which are frequently used in econometrics. It turns out, that 3-layer feedforward neural networks have a few drawbacks, e. g. complicated input preprocessing, limited abilities to represent temporal structures, and overfitting. As a remedy, chapters 3 and 4 consider the modeling of dynamical systems on the basis of feedforward and time-delay recurrent neural networks that incorporate prior knowledge about the specific task or first principles into the model building. Chapter 5 deals with the training of neural networks. Following our modeling philosophy, we point out that training is *not* only a synonym for the minimization of the overall network error function. Chapter 6 outlines recent developments in the research area of multi-agent modeling. Major design questions of agent-based financial markets are sorted out and it is shown, how these design issues are addressed in recent multi-agent literature. Chapter 7 introduces our first approach to multi-agent modeling on the basis of feedforward neural networks. We point out, that a single neuron can be seen as an elementary (but not trivial) decision-taking model of a single agent. Under this interpretation, a neural network of many neurons represents the interaction of many (economic) decisions (i. e. a market process). In Chapter 8, the decision-taking processes of the agents are based on elementary cognitive systems with three basic features: perception, internal processing and action. A cognitive system is structurally represented by a time-delay recurrent error correction neural network. In an enhanced version of this concept, we develop a multi-agent model based on the idea of *homeostatic* cognitive systems. Here we derive the features of the cognitive system deductively from the assumption of homeostasis. Homeostasis is an internal equilibrium, in which external influences are compensated with internal expectations. As a structural representation of homeostasis, we introduce zero-neurons within a time-delay recurrent neural network. In the following we refocus on the different chapters by summarizing the contents and major results.

Neural Networks: Introduction & Historical Notes

A 3-layer feedforward neural network is probably one of the most popular network architectures which is also frequently utilized in econometrics. Typically, this standard architecture consists of one input layer gathering the information from outside the neural network, one hidden layer incorporating the computational units of the network, and one output layer providing the network output as a response to a specific activation pattern. The information flow is stringently from the neurons of the input layer to those of subsequent layers.

Standard 3-layer feedforward neural networks have several drawbacks: Using a 3-layer feedforward neural network, a time series identification task is always transferred into a pattern recognition approach. Here one implicitly assumes, that a complete description of the underlying time series dynamics is included in the training data. If the data is unreliable, noisy or does not contain all relevant information about the underlying dynamics, a pattern recognition approach is clearly disadvantaged. A second drawback of the standard architecture is the representation of temporal structures. Feedforward neural networks only allow to map an input vector to a corresponding target vector. This implies that temporal structures can only be invented through a complicated input preprocessing. Another well-known problem of 3-layer feedforward neural networks is overfitting. Overfitting is the loss of the neural network's generalization performance, because of the high complexity of the network together with insufficiently small training data sets.

As a solution for the outlined problems, one may think of specific learning schemes (e. g. early or late stopping) together with optimization techniques for the network structure (e. g. weight pruning methods). Alternatively, one may think about neural networks as joint model building frameworks: Besides the learning from data, one may integrate prior knowledge about the underlying dynamical system into the modeling. Furthermore, the modeling can be enriched by the integration of first principles. For instance, spatial network architectures using unfolding in time and shared weights can be employed to model temporal structures, whereas a separation of time variants and invariants enables us to improve the modeling of high-dimensional dynamical systems. These additional elements of the model building may be incorporated into the neural network in form of architectural enhancements.

Modeling Dynamical Systems by Feedforward Nets

We believe that a model building which solely relies on the information contained in the data is often misleading. This is especially true for economic applications: Here, one typically has to identify a high-dimensional non-linear dynamic system on the basis of noisy data or insufficiently small data sets. As a remedy, we propose neural networks which include prior knowledge about the dynamic system in form of architectural enhancements. The 11-layer architecture is an example of a feedforward neural network that is in line with the latter model building philosophy.

The 11-layer feedforward neural network is especially designed for the modeling of dynamic systems. It is composed of several building blocks, which incorporate prior knowledge about the dynamic system into the modeling. The building blocks are integrated into the feedforward neural network in form of architectural extensions. Most remarkably, the 11-layer architecture achieves a maximal effect of synergy by the joint application of all building blocks.

Among the different building blocks of the 11-layer architecture, a network internal preprocessing based on hyperbolic tangent squashing is used to handle outliers. The input preprocessing is simplified by calculating only the momentum and force of each raw input signal. By the application of a square layer, we merge the (global) difference analysis of multi-layer perceptrons and the (local) similarity analysis of RBF networks. Additional information about the underlying dynamical system is incorporated into the 11-layer architecture by the concept of forces and embeddings. This idea is based on the well-known Takens theorem. Finally, besides robust error functions, the 11-layer architecture uses the technique of ensemble forecasting. In case, the errors of the different forecasts are uncorrelated, an averaging allows us to improve the model performance.

As a result, we obtain a powerful class of functions for the modeling of dynamical systems. The 11-layer architecture is able to provide us with a better description of the underlying system dynamics and thus, improves forecasting. We illustrated the prediction accuracy of the 11-layer architecture by its application to a real world problem. Forecasting the semi-annual development of the German bond index, it turned out that the 11-layer architecture is superior to a standard 3-layer feedforward neural network.

Modeling Dynamical Systems by Recurrent Networks

In this chapter we focus on time-delay recurrent error correction neural networks for the modeling of open systems. Likewise to the 11-layer feedforward approach, time-delay recurrent neural networks can be seen as a joint model building framework. Besides the integration of prior knowledge about the dynamical system, we incorporate so-called first principles into the model building. Prior knowledge as well as first principles are given to the recurrent networks in form of architectural enhancements.

Error correction neural networks include the last model error as an additional input. If we are unable to identify the underlying system dynamics due to missing external influences or noise, the last model error is an indicator of the model's misspecification. Since we use the model error as a measure of unexpected shocks, the learning of false causalities is lowered and the generalization performance is improved.

Instead of solving the system identification task by employing an optimization algorithm, we use a finite unfolding in time, that transforms the temporal identification task into a spatial architecture. The unfolded network is trained by a shared weights extension of standard error backpropagation. Remarkably, state space models allow the inclusion of memory effects. Our experiments indicated that the learning behavior of such a network has a bias on the external driven part. To focus the learning on the autonomous part of the dynamics, we propose an extension of the unfolding in time architecture called overshooting.

Further extensions to the ECNN are first principles like undershooting, unfolding in space & time, or a variants-invariants separation:

Undershooting: The relationship between the model and the data time grid is an important prestructuring element for the modeling of dynamical systems. We show that a refinement of the model time grid relative to the wider-meshed time grid of the observed data provides deeper insights into the dynamics. Such a refinement of the model time grid can be derived from the principle of uniform causality. Undershooting is a time-delay recurrent neural network based representation of this principle.

Unfolding in space & time: Here, we deal with the problem of optimal state space reconstruction. Starting off from a Takens embedding, we try to define a time independent, nonlinear coordinate transformation of the state space, which allows us to proceed with a smoother trajectory. The network architecture has to combine an unfolding in space (transformation of the internal space so that the trajectory is smoother than the observations) with an unfolding in time.

Variants-invariants separation: The complexity of a high dimensional dynamical system can be significantly reduced, if it is possible to separate the dynamics into time variant and invariant structures. Clearly, we only have to predict the variants of the system, because the invariants remain constant over time. A recombination of the variants and invariants provides us with a forecast of the complete system. Bottleneck neural networks allow the extraction of a maximal number of invariants, such that the high dimensional forecasting problem is simplified.

Depending on the specific application, a combination of ECNNs and at least one of the preceding first principles may improve the forecasting performance. We believe that an ECNN is a promising framework for financial forecasting.

Training of Neural Networks by Backpropagation

This chapter deals with the training of feedforward and time-delay recurrent neural networks. Standard error backpropagation is an efficient method to compute the partial derivatives of the network error function with respect to the weights. The shared weights extension of standard backpropagation enables us to train time-delay recurrent neural networks.

Training neural networks, we are interested in the identification of time invariant structures in varying time series. Standard learning algorithms like error backpropagation in combination with a particular learning rule (e. g. pattern-by-pattern or vario-eta learning) generate a hypothesis of such time invariant structures. Fitting our model $y_{t+1} = f(x, w)$ to the observed data x, forecasting is only feasible, if we assume that the explored structures are time-invariant (so-called invariance hypothesis). Unfortunately, a pure fitting of the data often leads to inadequate results, because the underlying dynamics may drift over time. As a remedy, we propose to revise the structures that are generated during the learning. Only if we cannot falsify the invariance hypothesis, the forecasts of our model are reliable.

The invariance hypothesis can be tested by instability pruning, which sorts out instable weights having a non-stationary gradient distribution. Additionally, one may apply partial learning, which supports the learning of localized structures and improves the generation of invariant structures in the sense of sparse network architectures.

Multi-Agent Modeling: A Guide to Literature

This chapter gives a brief survey of major design issues that have to be addressed in multi-agent market modeling. The issues focus on the micro- and macroeconomic structure of the market.

Microeconomic design issues are related to the modeling of the agents. Among other design questions, we have to consider the modeling of the agents' decision making schemes, related objective functions, heterogeneous decision making and the topic of learning and evolution:

Decision making schemes: The spectrum of decision making schemes ranges from simple rule-based agents to agents that incorporate econometric forecast models. Instead of assuming ad-hoc functional relationships, the decision making of the agents should be modeled by semantic specifications. This leads to cognitive system based approaches.

Objective functions: One may distinguish between explicitly and implicitly formulated objective functions. Both types are typically related to profit, wealth or utility maximization tasks. In case of an explicit representation, an agent has a well defined objective function which is directly included in his decision making process. We speak of an implicit representation, if the objectives are only indirectly incorporated into the agent's decision making scheme.

Heterogeneity: In general, heterogeneous decision making is required to adapt the underlying market dynamics. In the majority of cases, heterogeneous decision making is introduced by (*i.*) a varying information basis, (*ii.*) different parameter settings, (*iii.*) miscellaneous agent types, or (*iv.*) the learning and evolution of the agents.

Learning & evolution: This design issue describes, how the agents evolve their trading strategies or adapt to changing market conditions. In the simplest case, the agents do not modify their behavior. In other setups, the agents may switch between different types of trading strategies or may evolve their behavior by evolutionary algorithms. Agents who incorporate a forecast model typically utilize gradient based optimization methods.

Macroeconomic design issues address the modeling of the market structure and the interactions among the agents. Major building blocks are the assets traded in the market, the market structure and organization, and the price formation mechanism:

Assets: The types of assets underlying multi-agent models correspond to those of real world financial markets. One can distinguish between single and multiple risky asset setups. In a single risky asset setup, the agents allocate their capital either in a risk free asset or in a risky security which is traded on the market. In a multiple risky asset setup, the agents spread their funds among an enhanced number of securities.

Market structure: This design issue mainly addresses the communication and interaction among the agents, and the topic of trading synchroneity. Basically, one can distinguish between local and global market arrangements. A localized arrangement is characterized by a specific neighborhood structure, whereas in a globalized arrangement the agents only interact through the price formation mechanism. Trading synchroneity governs the time intervals at which trading is permitted. Synchronous trading permits trading only at predefined time points, while asynchronous trading does not assume fixed time schedules for the trading.

Price formation: The price formation rule regulates the trading of the assets, i. e. the market orders of the agents are related to price changes. The price formation is usually addressed in one of three different ways: First, one may assume that price adjustments occur in response to the observed market excess. The second way is to establish a temporary market equilibrium, so that a market clearing price can be found either analytically or computationally. The third way is to adapt the actual trading mechanisms of real-world financial markets.

By sorting out important design issues of multi-agent models, we outlined the current state of this research field. However, there are design questions which are only weakly addressed or remain as unsolved problems. Examples of those issues are asynchronous information processing schemes, the calibration of the underlying model parameters, or more realistic trading mechanisms. Remarkably, the majority of agent-based financial markets only deals with *qualitative* results of complex economic market phenomena. *Quantitative* results (i. e. the prediction of market price changes) are typically *not* supplied. The most obvious direction of future research is towards more realistic models of financial markets.

Multi-Agent Modeling by Feedforward Networks

Our first multi-agent model is based on feedforward neural networks. The approach utilizes the idea, that a single neuron can be seen as an elementary decision-taking model of a single agent.

Suppose, that an elementary model of (economic) decision making consists of *three* stages: information filtering, market evaluation and acting on the rated information. To clarify this decision making scheme, let us consider a trader dealing in a foreign exchange market.

Typically, this trader is overwhelmed by a flood of information, such as market overviews, various technical indicators and other fundamental data from different information sources. In order to handle the information overload, the agent has to focus his attention on a couple of news, which seem to have the deepest impact on the FX-rate (information filtering). By analyzing and correlating the selected data, the agent forms his view on the development of the FX-market (market evaluation). In the third stage of the decision making process, the market evaluation is transformed into a trading activity.

The outlined decision making scheme can be modeled by a single neuron, which precisely reflects the three stages of decision making. The process of filtering unneeded information can be adapted by adjusting the weights of the different input signals. Important external information is emphasized by adjusting the associated weighting factors to a higher value, while non-essential information is ignored by setting the corresponding weights equal to zero. The market evaluation is represented by the adder of the neuron: The sum of the weighted input signals can be seen as the simplest way of building a superposition of information, i. e. a market evaluation. The transformation of the market evaluation into an action is modeled by the activation function and the bias of the neuron. Depending on the quantitative value of the market evaluation (net input of the neuron), the activation function may switch to 1 or -1. The output of the neuron can be interpreted as the action of the trader (e. g. 1: buy shares, -1: sell shares).

This interpretation of a single neuron provides an important interface between economics and the mathematical theory of neural networks: If we assume, that a single neuron reflects the decision making process of a single trader, a neural network with hundreds of neurons describes a market process.

In our feedforward neural network approach, the decision making of each agent is represented by a single neuron. A market order of an agent corresponds to the output of the neuron modeling his decision making behavior. Each discrete time period, the agents submit buying and selling orders to the market. To keep things simple, price adjustments are perform as a reaction to the observed market excess. An agent can only buy a certain number of shares if other agents are selling an equivalent amount.

We extended our approach to a multiple risky asset setup, which allows to treat several markets simultaneously. Thus, our analysis take into account inter-market relationships through which all financial markets are now linked together as component parts of a larger whole. Furthermore, our multi-agent approach allows the fitting of real world data. An empirical study modeling different real-world FX-markets indicates, that the proposed multi-agent model is superior to common econometric forecasting techniques.

Multi-Agent Modeling by Cognitive Systems

In this chapter we introduce an uncommon way of multi-agent market modeling which is based on recurrent neural networks. The decision making of the agents is modeled by cognitive systems with three basic features: perception, internal processing and action. These features constitute necessary conditions of a cognitive system. In a first approach, we derive the features of the cognitive system *inductively.* As a structural representation of the inductive approach, we apply error correction neural networks. In a second approach, the three features of the cognitive system are derived *deductively* from the assumption of homeostasis. Given a changing environment, homeostasis can be seen as the attempt of a cognitive agent to maintain an internal equilibrium. Similar to the inductive approach, we model the deductive cognitive system with a recurrent neural network.

Let us focus on the *deductive* approach to cognitive systems. The goal is to model the decision making of an agent with a homeostatic dynamic system. Based on the assumption of a homeostatic state, we derive the properties perception, internal processing and action:

Homeostasis: Homeostasis may be defined as a dynamically stable state, in which external influences (perception) are balanced with internal expectations such that an internal equilibrium is maintained. Perception may disturb the internal stable state. In order to keep the internal equilibrium, the cognitive system is forced to develop an internal model and to initiate actions, which counterbalance the disturbances.

Perception: We distinguish conscious from unconscious perception. In conscious perception, the system compares selective observations to an internal expectation. Only the resulting difference has an impact on the system. Unconscious perception enters the system directly, i. e. it corresponds to stimulus-response. Unconscious perception is therefore not balanced with internal expectations.

Internal processing: The internal processing mainly consists of an internal model of the external world. To maintain the internal equilibrium, homeostasis implies the construction of an internal model, which balances conscious perception with internal expectations. Without this balance, the system would be driven by stimulus-response. For the construction of internal expectations, an internal memory is required. The memory can be seen as a merging of all relevant past information from which the internal expectation is formed.

Actions: To initiate actions, the internal expectation is evaluated by an objective function (e. g. profit or utility maximization). Hence, actions are always goal-oriented. In the homeostatic cognitive system perception and action are decoupled: incoming stimuli not necessarily lead to actions if they do not disturb the internal equilibrium. Due to this decoupling, the actions of a homeostatic cognitive system are driven by the objective function rather than by a simple stimulus-response mechanism.

As a structural representation of the cognitive system, we refer to a recurrent neural network framework:

Homeostasis: We suggest to model homeostasis by zero-neurons within the recurrent neural network. Zero-neurons are input-output neurons with constant, task invariant target values of 0 and input weights fixed at -1.

Perception: Conscious perceptions are always connected to a zero-neuron. Due to the fixed input weights of the zero-neuron, conscious perceptions induce an error flow, which forces the internal model to construct a corresponding expectation as a compensation. Only the difference between the conscious perception and the internal expectation has an impact on the system. Unconscious perceptions are directly connected to the hidden neurons of the recurrent network. Although unconscious perception has a direct impact on the cognitive system, the system is not forced to build internal expectations as a compensation. This corresponds to the principle of stimulus-response.

Internal Processing: The internal processing of the cognitive system is structurally represented by the recurrent neural network. Due to the recurrent construction, the system is able to accumulate a superposition of all relevant past information, i. e. a memory, which is required to achieve a dynamically stable state.

Actions: In addition to the internal expectations, the cognitive system suggests actions. Likewise to the internal expectations, actions are modeled as outputs of the recurrent neural network. The cognitive system perceives the actual state of the environment and the results of the executed actions. If an intended action cannot be fulfilled, the observed action and the intention differ.

The decision making of a single agent is modeled with a homeostatic cognitive system. The aggregation of the agents' decisions leads to a multi-agent based approach of market modeling. The cognitive agents are arranged in a higher-level time-delay recurrent neural network. The higher-level network architecture separates the agents by block diagonal matrices. Due to the block design, the agents interact only through the price formation mechanism, which is also included in the network. Again, price adjustments are performed as a reaction to the observed market excess.

Cognitive system based agent models are an approach of capturing semantic specifications of the agents' decision schemes in a structural framework. This is truly a more realistic modeling of the agents' behavior as suggested in the feedforward neural network approach. As a remarkable property, our multi-agent approach allows the fitting of real world data. We applied our multi-agent model to the German Mark / US-Dollar foreign exchange market. The empirical study indicated, that our multi-agent model is superior to several benchmarks in an out-of-sample period of one and a half years. Due to these encouraging results, we believe that cognitive system based multi-agent models are a promising framework for financial forecasting.

Contribution of this work

Let us comment on some major contributions of this thesis to the current research in financial forecasting and multi-agent modeling.

Financial data is often covered with noise or does *not* contain all information which is required to perform a comprehensive analysis of the underlying dynamic system. Furthermore, we often have not enough data to build a good forecast model for a complex economic system. In other words, the quality and quantity of the underlying data restricts the performance of the forecasts. As a remedy, we propose to enrich the model building by the incorporation of prior knowledge about the specific application. The learning from data is only *one* part of this process. The 11-layer feedforward neural network and the time-delay recurrent error correction neural network are two examples of this model building philosophy. Remarkably, such a joint model building framework does *not* only provide superior forecasts, but also a deeper understanding of the underlying problem. On this basis, it is also possible to analyze and to quantify the uncertainty of the predictions. This is especially important for the development of decision support systems.

In the research area of multi-agent modeling, the topic of forecasting remains as an unsolved problem. The majority of agent-based models focuses only on so-called stylized facts of financial markets (e. g. fat-tailed return distributions or volatility clustering). In other words, only qualitative analysis of complex economic market phenomena are provided, while quantitative results (i. e. forecasting of real-world market prices) are typically not supplied. The main point the reader should draw from this thesis is that it is possible to forecast real world price movements on the basis of a multi-agent model. Remarkably, all our neural network based multi-agent models showed an excellent balance between being complex enough to model a rich microscopic structure and a regular mathematical structure to fit an integrated model to financial data. The performance of our multi-agent approaches is underlined by several empirical studies of FX-markets. Compared to traditional econometric benchmarks, it turned out that our agent-based markets are superior forecasting instruments.

In the majority of agent-based markets, the decision making schemes of the agents are modeled by simple decision rules or the assumption of ad-hoc functional relationships. We provided a more realistic approach to the modeling of the agents' decision making which is based on the idea of a simple cognitive process. Agents perceive the development of the market and initialize actions on the basis of their internal expectations. In other words, our cognitive system based approach captures semantic specifications of the agents' decision schemes in a structural framework, which is given by time-delay recurrent neural networks. This is a more realistic way of modeling the behavior of real-world market participants.

Outlook and Further Research

We may conclude this thesis with some thoughts on how the research on our multi-agent models may be extended.

The most obvious research direction is towards more realistic multi-agent models of financial markets. Concerning the macroeconomical side of an agent-based market, one may think of price formation and trading mechanisms that replicate those of real-world financial markets. For instance, more realistic price mechanism may incorporate limit orders, the formation of bid and ask spreads, or order crossing rules. Alternatively, the market clearing can be performed by a specialist system or an adaptive market maker. On the microeconomical side of the market, the myopic one step decision making schemes of the agents could be extended to multiple step decision schemes. This means, that the agents should be allowed to consider inter-temporal plans or multi-period preferences. Furthermore, one may allow the agents to set up an adaptive network of communication or collaboration, in which they exchange their expectations or private information. In this case, the market dynamics is not only created by the interaction of the agents on the macroeconomic level, but also arises from the agents' interrelation and communication on the microeconomic level.

Beyond that, another possible direction of research is the analysis of risk. In a multi-agent model, the risk of a single agent can be defined as the uncertainty of his expectations (e. g. incorrect market price forecasts). From this point of view, risk is an implicit measure of the overall complexity of the environment in contrast to the forecasting abilities of a single agent. Remarkably, an agent-based market allows us to explain at least a part of the market complexity by an analysis of the other agents behavior. Further research in this direction may open up new views on market risks.

References

[AHS85] Ackley D. H., Hinton G. E., and Sejnowski T. J.: *A Learning Algorithm for Boltzmann Machines*, Cognitive Science, 9, 1985, pp. 147-169.

[Art95] Arthur W. B.: *Complexity in economic and financial markets*, in: Journal of Complexity, 1995, pp. 20-25.

[Ari96] Arifovic J.: *The behavior of the exchange rate in the genetic algorithm and experimental economics*, in: Journal of Political Economy, Vol. 104, 1996, pp. 510-541.

[BPS96] Bak P., Paczuski M. and Shubik M.: *Price Variations in a Stock Market with Many Agents*, Santa Fe Institute Working Paper, No. 96-09-075, 1996, p. 1-59.

[Bar97] Bar-Yam Y.: *Dynamics of Complex Systems*, Addison-Wesley, Massachusetts, 1997.

[BGH98] Barnett W. A., Gallant A. R., Hinich M. J., Jungeilges J. A., Kaplan D. T. and Jensen M. J.: *A Single-Blind Controlled Competition Among Tests for Nonlinearity and Chaos*, in: Journal of Econometrics, Vol. 82, 1998, pp. 157-192.

[BM92] Beltratti, A. and S. Margarita: *Evolution of trading strategies among heterogeneous artificial economic agents*, in: J.-A. Meyer, S.W. Wilson & H.L. Roitblat (Eds.), From Animals to Animats (2), Proceedings of the 2nd International Conference on Simulation of Adaptive Behaviour (SAB '92), MIT Press 1993, pp. 494 - 501.

[BMT96] Beltratti, A., Margarita, S. and Terna, P.: *Neural Networks for economic and financial modeling*, International Thomson Computer Press, London, UK, 1996.

[Bis95] Bishop C. M.: *Neural networks for pattern recognition*, Clarendon Press, Oxford, 1995.

[BL92] Black F. and Litterman R.:*Global Portfolio Optimization*, Financial Analysts Journal, Sep. 1992.

[Bol86] Bollerslev T.: *Generalized autoregressive conditional heteroscedasticity*, in: Journal of Econometrics, No. 31, 1986, pp. 307-327.

[BCK92] Bollerslev T., Chou R. Y. and Kroner K. F.: *ARCH modeling in finance: a review of the theory and empirical evidence*, in: Journal of Econometrics, No. 52, 1992, pp. 5-59.

[Bor01] Bornholdt S.: *Expectation bubbles in a spin model of markets: Intermittency from frustration across scales*, in: International Journal of Modern Physics C, Vol. 12, No. 5, 2001, pp. 667-674.

[BJ76] Box G. E. and Jenkins G. M.: *Time series analysis, forecasting and control*, Revised edition, Holden-Day series in time series analysis and digital processing, Holden-Day, Oackland, 1976.

[BLeB96] Brock W. A. and LeBaron B.: *A dynamic structural model for stock return volatility and trading volume*, Review of Economics and Statistics 78, 1996, pp. 94-110.

[BDS87] Brock W. A., Dechert W. D., Scheinkman J. A. and LeBaron B.: *A test for independence based on the correlation dimension*, Econometric Reviews, Vol. 15, 1987, pp. 197 - 235.

[BH97] Brock W. and Hommes C.: *A rational route to randomness*, in: Econometrica, Vol. 65, 1997, pp. 1059-1095.

[BrLo88] Broomhead D.S. and Lowe D.: *Multivariable functional interpolation and adaptive networks.* in: Complex Systems, Vol. 2, 1998, pp. 321-355.

[Bub99] Bubhaus C. and Rieger H.: *A prognosis oriented microscopic stock market model.* Physica A, Vol. 267, 1999, pp. 443-452.

[CMZ97] Caldarelli G., Marsili M. and Zhang Y.-C.: *A prototype model of stock exchange*, in: Europhysics Letters, Vol. 40, No. 5, 1997, pp. 479-484.

[CK01] Calvert D. and Kremer St.: *Networks with Adaptive State Transitions*, in: Kolen J. F. and Kremer St. [Eds.]: *A Field Guide to Dynamical Recurrent Networks*, IEEE Press, 2001, p. 15-25.

[CEFG91] Casdagli M., Eubank S., Farmer J. D., and Gibson J.: *State space reconstruction in the presence of noise*, Physica D, Vol. 51, 1991, pp. 52-98.

[Cast00] Castiglione F.: *Diffusion and Aggregation in an Agent based Model of Stock Market Fluctuations*, in: International Journal of Modern Physics C, Vol. 11, No. 5, 2000, pp. 865-880.

[Chal97] Chalmers D.: *Facing up the problem of consciousness*, in: Explaining Consciousness: The Hard Problem, ed. J. Shear, MIT Press, 1997.

[Chak00] Chakrabarti R.: *Just Another Day in the Inter-bank Foreign Exchange Market*, in: Journal of Financial Economics, Vol. 56, No. 1, 2000, pp. 29-64.

[Chak99] Chakrabarti R. and Roll R.: *Learning from others, reacting and market quality*, in: Journal of Financial Markets, Vol. 2, 1999, pp. 153-178.

[CLeLP99] Chan N. T., LeBaron B., Lo A. and Poggio T.: *Agent-based models of financial markets: A comparison with experimental markets*, MIT, Artificial Markets Project, Paper No. 124, September 1999.

[Car94] Caruana R.: *Learning Many Related Tasks at the Same Time with Backpropagation*, in: Advances in Neural Information Processing Systems (NIPS 1994), Tesauro G., Touretzky D. and Leen T. [Eds.], Vol. 7, MIT Press, 1995, pp. 657-664.

[ChYe01] Chen S.-H., and Yeh C.-H.: *Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market*, in: Journal of Economic Dynamics and Control, Vol. 25, 2001, pp. 363-393.

[CheM98] Cherkassky, V. and Mulier, F.: *Learning from Data - Concepts, Theory and Methods.* John Wiley & Sons, New York, 1998.

[ChHe01] Chiarella C. and He X. Z.: *Asset Price and wealth dynamics under heterogeneous expectations*, in: Quantitative Finance, Vol. 1, 2001, pp. 509-26.

[Cra43] Craik K.: *The Nature of Explanation*, Cambridge University Press, Cambridge 1943.

[CS97] Churchland P. S. and Sejnowski T. J.: *Grundlagen zur Neuroinformatik und Neurobiologie*, The Computational Brain in deutscher

Sprache, in: Bibel W., von Hahn W. and Kruse R. [Eds.]: Computational Intelligence, Vieweg Verlag, 1997.

[CMcV01] Clancy J. and McVicar A.: *Physiology and Anatomy: A homeostatic approach*, second edition, Edward Arnold, London 2001.

[Cli95] Cliff D.: *Neuroethology, Computational*, in: M. A. Arbib [Ed.]: *The Handbook of Brain Theory and Neural Networks*, MIT-Press, 1995, pp. 626-630.

[ClBr97] Cliff D. and Bruten J.: *Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets*, Technical Report HPL-97-141, Hewelett-Packard Laboratories, Bristol, UK., 1997.

[CPS89] Cutler D. M., Poterba J. M. and Summers L. H.: *What Moves Stock Prices?*, in: Journal of Portfolio Management, Vol. 15, No. 3, 1989, pp. 412.

[Cow90] Cowan J. D.: *Neural Networks: The Early Days*, in: Advances in Neural Information Processing Systems (NIPS), Vol. 2, Touretzky D., [Ed.], Morgan Kaufmann, 1990, pp. 828-842.

[DeBT95] DeBondt W. F. M. and Thaler R. H.: *Financial decision-making in markets and firms: A behavioral perspective*, In: Robert A. Jarrow, Voijslav Maksimovic & William T. Ziemba (eds.): Finance, Handbooks in Operations Research and Management Science, Vol. 9, chp. 13, North Holland, Amsterdam, 1995, pp. 385-410.

[DeG93] De Grauwe P., Dewachter H. and Embrechts M.: *Exchange Rate Theory: Chaotic Models of Foreign Exchange Markets*, Blackwell, Oxford. 1993.

[MY95] De la Maza M. and Yuret D.: *A model of stock market participants*, in: Eds. J. Biethahn and V. Nissen, Evolutionary Algorithms in Management Applications, Springer Verlag, Heidelberg, 1995, pp. 290-304.

[Diba88] Diba B. T. and Grossman H. I.: *The Theory of Rational Bubbles in Stock Prices*, in: Ecocomic Journal, Vol. 98, 1988, pp. 746-754.

[DF79] Dickey D. A. and Fuller W. A.: *Distribution of the Estimators for Autoregressive Time Series with a Unit Root*, in: Journal of the American Statistical Association, Vol. 74, 1979, pp. 427-431.

[DO00] Drossu R. and Obradovic Z.: *Data Mining Techniques for Designing Neural Network Time Series Predictors*, in: I. Cloete and J. M. Zurada [Eds.]: *Knowledge-Based Neurocomputing*, MIT Press, Cambridge, Massachusets, 2000, pp. 325-67.

[EM01] Edmonds B. and Moss S.: *The Importance of Representing Cognitive Processes in Multi-agent Models*, in: Dorffner, G., Bischof, H. and Hornik, K. [Eds.]: Proceedings of the International Conference on Artificial Neural Networks (ICANN), Spinger, Lecture Notes in Computer Science, Vol. 2130, Vienna, 2001 pp. 759-766.

[EG95] Elton E.J. and Gruber M.J.: *Modern Portfolio Theory and Investment Analysis*, John Wiley and Sons, 1995.

[Elm90] Elman J. L.: *Finding structure in time*, Cognitive Science, 14, 1990, pp. 179-211.

[Eng82] Engle R. F.: *Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation*, in: Econometrica, Vol. 50, 1987, pp. 987-1007.

[EP01] Engle, R., and A. Patton: *What Good is a Volatility Model?*, in: Quantitative Finance, Vol. 1, Institute of Physics Publishing, 2001, pp. 237-245.

[EpAx97] Epstein J. M. and Axtell R.: *Growing Artificial Societies*, MIT Press, Cambridge, MA. 1997.

[Fam70] Fama E. F.: *Efficient Capital Markets: A Review of Theory and Empirical Work*, in: Journal of Finance, Vol. 25, No. 2, 1970, pp. 383-417.

[Fam91] Fama E. F.: *Efficient Capital Markets II*, in: Journal of Finance, Vol. 46, No. 5, 1991, pp. 1575-1617.

[Fam97] Fama, E. F.: *Market Efficiency, Long-Term Returns, and Behavioral Finance*, Center for Research in Security Prices Working Paper 448, University of Chicago, 1997.

[Far98] Farmer, J.D.: *Market Force, Ecology, and Evolution*, Santa Fe Institute Working Paper, 98-12-116, 1998.

[Far99a] Farmer J. D. and Lo A.: *Frontiers of finance: Evolution and efficient markets*, in: Proceedings of the National Academy of Sciences 96, 1996, pp. 9991-9992.

[FJ99] Farmer, J. D. and Joshi, S.: *The Price Dynamics of Common Trading Strategies*, Santa Fe Institute Working Paper, 00-12-069, 1999.

[Fla97] Flake G.: *Square unit augmented, radially extended multilayer perceptrons*, in: Orr G., Müller K.-R., and Caruana R. [Eds.], Tricks of the Trade: How to Make Algorithms Really Work, LNCS State of the Art Surveys, Springer-Verlag, 1998, pp. 145-164.

[FriR93] Friedman, D. and Rust J. [Eds.]: *The Double Auction Market: Institutions, Theories, and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings, Vol. 14, Addison-Wesley, New York, 1993.

[Fri91] Friedman D.: *Evolutionary Games in Economics*, in: Econometrica, Vol. 59, 1991, pp. 637-666.

[GS83] Gentner D. and Stevens A. L. [Eds.]: *Mental Models*, Lawrence Erlbaum Associates, Hillsdale (NJ), 1983.

[Gra81] Granger, C. W. J.: *Some properties of time series data and their use in econometric model specification*, in: Journal of Econometrics, 16, 1981, pp. 121-130.

[GraE87] Granger, C. W. J. and Engle, R. F.: *Co-Integration and Error Correction: Representation, Estimation, and Testing*, in: Econometrica 55, 1987, pp. 251-276.

[GraP83] Grassberger P. and Procaccia I.: *Measuring the strangeness of a strange attractor*, in: Physica D, Vol. 9, 1983, pp. 189-208.

[Ger89] Gershenfeld N. A.: *An experimentalist's introduction to the observation of dynamical systems*, in: B. L. Hao, editor, *Directions in Chaos*, World Scientific, Vol. 2, Singapore, 1989, pp. 310-384.

[GFC92] Gibson J., Doyne Farmer J., Casdagli M., and Eubank St.: *An analytic approach to practical state space reconstruction*, Physica D, 57, 1992.

[Gil91] Giles C. L., Miller C. B., Chen D., Chen H., Sun G. Z., and Lee Y. C.: *Grammatical Inference Using Second-Order Recurrent Neural Networks*, in: Proceedings of the International Joint Conference on Neural Networks, Seattle, Washington, IEEE Publication, vol. 2, 1991, p. 357.

[GS93] Gode D. K. and Sunder, S.: *Allocative efficiency of markets with zero intelligence traders*, Journal of Political Economy 101, 1993, pp. 119-137.

[Gro88] Grossberg St.: *Neural Networks and Natural Intelligence.* MIT Press, Cambridge, MA., 1988.

[GroSt80] Grossman S. J. and Stiglitz, J. E.:*On the Impossibility of Informationally Efficient Markets*, in: The American Economic Review, 1980, pp. 393-408.

[Har76] Hardy R. N.: *Homeostasis*, Edward Arnold, London, 1976.

[Hay96] Haykin S.:*Adaptive Filter Theory*, Prentice-Hall International, 3rd edition, 1996.

[Hay94] Haykin S.: *Neural Networks. A Comprehensive Foundation*, Macmillan College Publishing, New York, 1994. second edition 1998.

[Heb49] Hebb, D.: *The Organization of Behavior*, NY: John Wiley & Sons, 1949.

[HMcCR86] Hinton G. E., McClelland J. L. and Rumelhart D. E.: *Distributed Representations*, in: D.E. Rumelhart, J. L. McClelland, et al., Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Vol. 1: Foundations, Cambridge: M.I.T. Press, 1986, pp. 77-109.

[HBFS01] Hochreiter S., Bengio Y., Frasconi P. and Schmidhuber J.: *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-term Dependencies*, in: A Field Guide to Dynamical Recurrent Networks, Eds. Kolen, J.F.; Kremer, St.; IEEE Press, 2001, pp. 237-243.

[Hol92] Holland, J. H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 2nd edition, MIT Press / Bradford Books, 1992.

[Hom01] Hommes C.: *Financial markets as nonlinear adaptive evolutionary systems*, in: Quantitative Finance, Vol. 1, 2001, pp. 149-167.

[Hop92] Hopfield J.J.: *Neural networks and physical systems with emergent collective computational abilities*, in: Proceedings of the National Academy of Sciences, No. 79, pp. 2554-2558, 1982.

[HSW92] Hornik K., Stinchcombe M. and White H.: *Multi-layer Feedforward Networks are Universal Approximators*, in: White H. et al. [Eds.]: Artificial Neural Networks : Approximation and Learning Theory, Blackwell Publishers, Cambridge, MA, 1992.

[Hub96] Huberman B. A.: *Computation as Economics*, in: Proceedings of the Second International Conference in Economics and Finance, 1996, available at *http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2039*, pp. 1-21.

[JS96] Jamal K. and Sunder S.: *Bayesian equilibrium in double auctions populated by biased heuristic traders*, in: Journal of Economic Behavior and Organization, Vol. 31, 1996, pp. 273-91.

[Jua99] Juarrero, A.: *Dynamics in Action*, Intentional Behavior as a Complex System, MIT Press, Cambridge, MA., London 1999.

[JPB00] Joshi S., Parker J. and Bedau M. A.: *Technical Trading Creates a Prisoner's Dilemma: Results from an Agent-Based Model*, in: Proceedings of the 6th International Conference Computational Finance, Eds. Y. Abu-Mostafa, B. LeBaron, A. W. Lo and A. S. Weigend, MIT Press, 2000, pp. 465-479.

[Kai00] Kaizoji T.:*Speculative bubbles and crashes in stock markets: an interacting-agent model of speculative activity*, in: Physica A, Statistical Mechanics and its Applications, Special Issue, Vol. 287, No. 3-4, 2000, pp. 493-506.

[KGV83] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.: *Optimisation by simulated annealing*, in: Science, 220, 1983, pp. 671-680.

[Kir96] Kirman, A.: *Some Observations on Interaction in Economics*, Talk given on the Workshop on Structural Change, Manchester Metropolitan University, May 1996. Available in electronical form at *ftp://cfpm.org/pub/workshop/kirman.ps*, pp. 1-6.

[Kir02] Kirman A. and Teyssiere G.: *Bubbles and Long Range Dependence in Asset Prices Volatilities*, in: C. Hommes, R. Ramer and C. Withagen [Eds.], Equilibrium, Markets and Dynamics, 2002, forthcoming.

[KCG90] Kuczma M., Choczewski B., Ger R.: *Iterative Functional Equations*, Cambridge Uni. Pr., 1990.

[Koh88] Kohonen T.: *Self-Organization and Associative Memory*, second edition, Springer, Berlin, 1988.

[KK01] Kolen J. F. and Kremer St. [Eds.]: *A Field Guide to Dynamical Recurrent Networks*, IEEE Press, 2001.

[Kol01] Kolen J. F.: *Dynamical Systems and Iterated Function Systems*, in: Kolen J. F. and Kremer St. [Eds.]: *A Field Guide to Dynamical Recurrent Networks*, IEEE Press, 2001, p. 57-81.

[Koe67] Koestler A.: *The Ghost in the Machine*, Arkana, London, 1967.

[Lan65] Langley L. L.: *Homeostasis*, Reinhold, New York, 1965.

[LDS90] LeCun Y., Denker J. S. and Solla S. A.: *Optimal Brain Damage*, in: Advances in Neural Information Processing Systems (NIPS 1990), Touretzky D. [Ed.], Vol. 2, Morgan Kaufmann, 1991, pp. 598-605.

[LAHP97] LeBaron B., Arthur W. B., Holland J. H., Palmer R. G. and Tayler P.: *Asset Pricing Under Endogenous Expectations in an Artificial Stock Market*, Santa Fe Institute Working Paper, 96-12-093, 1996. Also published in: The Economy as an Evolving Complex System II, Eds.: W. B. Arthur, S. Durlauf, and D. Lane, Addison-Wesley, 1997.

[LAP99] LeBaron B., Arthur W. B. and Palmer R. G.: *The Time Series Properties of an Artificial Stock Market*, in: Journal of Economic Dynamics and Control, Vol. 23, 1999, pp. 1487-1516.

[LeB01a] LeBaron B.: *A Builder's Guide to Agent Based Financial Markets*, in: Quantitative Finance, Vol. 1, No. 2, 2001, pp. 254-261.

[LeB01b] LeBaron B.: *Evolution and Time Horizons in an Agent-Based Stock Market*, in: Macroeconomic Dynamics, Vol. 5, 2001, pp. 225-254.

[LeB00] LeBaron B.: *Agent-based computational finance: Suggested readings and early research*, in: Journal of Economic Dynamics and Control, Vol. 24, 2000, pp. 679-702.

[LeB99] LeBaron B.: *Building Artificial Markets With Artificial Agents: Desired goals and present techniques*, in: Karakoulas G. [Ed.], Computational Markets, MIT Press, 1999, forthcoming, preprint available at *http://people.brandeis.edu/ blebaron/id27.htm.*

[Let99] Lettau M. and Uhlig H.: *Rule of thumb and dynamic programming*, in: American Economic Review, Vol. 89, 1999, pp. 148-174.

[Let97] Lettau M.: *Explaining the facts with adaptive agents: The case of mutual fund flows*, in: Journal of Economic Dynamics and Control, Vol. 21, 1997, pp. 1117-1148.

[LLS94] Levy M., Levy H. and Solomon S.: *A microscopic model of the stock market: cycles, booms, and crashes*, in: Economics Letters, Vol. 45, 1994, pp. 103-111.

[LLS00] Levy M., Levy H. and Solomon S.: *Microscopic Simulation of Financial Markets*, Academic Press, New York, 2000.

[LN77] Lindsay P. H. and Norman D. A.: *Human Information Processing: An Introduction to Psychology*, second edition, Academic Press, New York, London, 1997.

[LipB89] Lippmann R. P. and Beckman P.: *Adaptive Neural Net Preprocessing for Signal Detection in Non-Gaussian Noise*, in: Advances in Neural Information Processing Systems (NIPS 1989), Touretzky D. [Ed.], Vol. 1, Morgan-Kaufmann, 1989, pp. 124-132.

[Lip87] Lippmann R. P.: *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, Vol. 4, 1987, pp. 4-22.

[Luc72] Lucas R. E.: *Expectations and the Neutrality of Money*, in: Journal of Economic Theory, Vol. 4, 1972, pp. 103-124.

[LCM01] Lux T. , Chen S. H. and Marchesi M.: *Testing for Nonlinear Structure in an Artificial Financial Market*, in: Journal of Economic Behavior and Organization, vol. 46, 2001, pp. 327-342.

[LM01] Lux T. and Marchesi M.: *Volatility clustering in financial markets: A micro-simulation of interacting agents*, in: Journal of Theoretical and Applied Finance, No. 3, 2001, pp. 675-702.

[LM99] Lux T. and Marchesi M.: *Scaling and Criticality in a Stochastic Multi-Agent Model of a Financial Market*, in: Nature, Vol. 397, 1999, pp. 498-500.

[Lux98] Lux T.: *The Socio-Economic Dynamics of Speculative Markets: Interacting Agents, Chaos, and the Fat Tails of Return Distributions*, in: Journal of Economic Behavior and Organization, vol. 33, 1998, pp. 143-165.

[Lux95] Lux T.: *Herd Behaviour, Bubbles and Crashes*, in: The Economic Journal, vol. 105, 1995, pp. 881-896.

[Mac98] MacPhail E.: *The Evolution of Consciousness*, Oxford University Press, New York, 1998.

[Mal81] Malkiel B.: *A Random Walk Down Wall Street*, 2nd edition, Norton, New York, 1981.

[vdMS] Malsburg von der C. and Schneider W.: *A Neural Cocktail–Party Processor*, in: Biological Cybernetics, Vol. 54, 1986, pp. 29-40.

[MC01] Mandic D. P. and Chambers J. A.: *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*, Wiley Series in Adaptive and Learning Systems for Signal Processing, Communications and Control, S. Haykin [Ed.], John Wiley & Sons, Chichester 2001.

[Mar52] Markowitz H. M.: *Portfolio Selection*, in: Journal of Finance, Vol. 7, 1952, pp. 77-91.

[Marsh95] Marshall J. A.: *Motion Perception: Self-Organization*, in: M. A. Arbib [Ed.]: *The Handbook of Brain Theory and Neural Networks*, MIT-Press, 1995, pp. 589-91.

[MP43] McCulloch W.S. and Pitts W.: *A logical calculus of the ideas immanent in nervous activity*, Bull. Math., Biophysics No. 5, 1943, pp. 115-33.

[MJ99] Medsker L. R. and Jain L. C.: *Recurrent Neural Networks: Design and Application*, CRC Press international series on comp. intelligence, No. I, ISBN 0-8493-7181-3, 1999.

[Meh95] Mehta M.: *Foreign Exchange Markets*, in: A. P. Refenes: Neural Networks in the Capital Markets, John Wiley & Sons, Chichester, 1995, pp. 177-98.

[Met00] Metzinger Th.: *Neural Correlates of Consciousness*, Empirical and Conceptual Questions, MIT Press, Cambridge, Ma., London 2000.

[MoD89] Moody J. and Darken Ch. J.: *Fast learning in networks of locally-tuned processing units*, in: Neural Computation, 1989, pp. 281-294.

[Mur91] Murphy, J. J.: *Intermarket Technical Analysis*, New York, 1991.

[Mut61] Muth J.F.: *Rational Expectations and the Theory of Price Movements*, 1961.

[MiPa69] Minsky, M.L., Papert, S.A.: *Perceptrons*, Cambridge, MA, MIT Press, first edition, 1969; expanded edition, 1988.

[NP90] Narendra K. S. and Parthasarathy K.*Identification and control of dynamical systems using neural networks*, in: IEEE Transactions on Neural Networks, Vol. 1, 1990, pp. 4-27.

[Neu86] Neumann J.: *Papers of John von Neumann on Computing and Computer Theory*, Eds. W. Aspray and A. Burks, Cambridge, MA, MIT Press, 1986.

[NZ98] Neuneier R. and Zimmermann H. G.: *How to Train Neural Networks*, in: *Neural Networks: Tricks of the Trade*, Springer, Berlin, 1998, pp. 373-423.

[OSY94] Ott E., Sauer T. and Yorke J. A.: *Coping with Chaos*, Wiley, 1994.

[PCFS80] Packard N., Crutchfield N. H., Farmer J. D. and Shaw R. S.: *Geometry from a time series*, Phys. Rev. Lett., Vol. 45, 1980, pp. 712-716.

[PAHL94] Palmer R. G., Arthur W. B., Holland J. H., LeBaron B. and Taylor P.: *Artificial Economic Life: A Simple Model of a Stockmarket*, in: Physica D, Vol. 75, 1994, pp. 264-274.

[PAHL98] Palmer R. G., Arthur W. B., Holland J. H., LeBaron B.:*An Artificial Stock Market*, in: Artificial Life and Robotics, Vol. 3, 1998, pp. 27-31.

[PS93] Park J. and Sandberg, I.: *Approximation and Radial-Basis-function Networks*, in: Neural Computation, Vol. 5, 1993, pp. 305-316.

[Pat87] Patinkin D.: *Walras Law*, in: The New Palgrave, A Dictionary of Economics, Macmillan, London, 1987, Vol. 4, pp. 864-868.

[Per01] Perlovsky L.I.: *Neural Networks and intellect: using model-based concepts.* Oxford University Press, New York, 2001.

[Pearl01] Pearlmatter B.: *Gradient Calculations for Dynamic Recurrent Neural Networks*, in: A Field Guide to Dynamical Recurrent Networks, Eds. Kolen, J.F.; Kremer, St.; IEEE Press, 2001, pp. 179-206.

[Pearl95] Pearlmatter B.: *Gradient Calculations for Dynamic Recurrent Neural Networks: A survey*, in: IEEE Transactions on Neural Networks, Vol. 6, Nr. 5, 1995, pp. 1212-1228.

[Perr93] Perrone, M.: *Improving regression estimation: averaging methods for variance reduction with extensions to general convex measure optimization*, PhD thesis, Brown University, Physics Dept., 1993.

[PDP00] Poddig Th., Dichtl H. and Petersmeier K.: *Statistik, Okonometrie, Optimierung: Grundlagen und Anwendungen in Finanzanalyse und Portfoliomanagement*, Uhlenbruch, Bad Soden/Ts., 2000

[Pod99] Poddig Th.: *Handbuch Kursprognose*, Uhlenbruch, Bad Soden/Ts, 1999.

[PG90] Poggio T. and Girosi F.: *Networks for approximation and learning.* in: Proceedings IEEE, Vol. 78, Nr. 9, 1990, pp. 1481-1497.

[Pol91] Polak, E.: *Computational Methods for Optimization*, Academic Press, New York, 1991.

[Pow85] Powell M. J. D.: *Radial basis functions for multivariable interpolation: A review.* IMA Conference on Algorithms for the Approximation of Functions and Data, RMCS, Shrivenham, England, 1985, pp. 143-167.

[Ref95] Refenes A.-P.: *Testing Strategies and Metrics*, in: A. P. Refenes: Neural Networks in the Capital Markets, John Wiley & Sons, Chichester, 1995, pp. 67-76.

[RHW86] Rumelhart D. E., Hinton G. E. and Williams R. J.: *Learning Internal Representations by Error Propagation*, in: D.E. Rumelhart, J. L. McClelland, et al., Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Vol. 1: Foundations, Cambridge: M.I.T. Press, pp. 318-362, 1986.

[RSMH86] Rumelhart D. E., Smolensky P., McClelland J. L. and Hinton G. E.: *Schemata and Sequential Thought Processes in PDP Models*, in: D.E. Rumelhart, J. L. McClelland, et al., Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Vol. 2: Psychological and Biological Models, Cambridge: M.I.T. Press, pp. 7-58, 1986.

[Rum86] Rumelhart D. E. et al.: *Parallel Distributed Processing: Explorations in The Microstructure of Cognition*, two volumes, Cambridge: M.I.T. Press, 1986.

[RN95] Russell St. and Norvig P.: *Artificial Intelligence: A Modern Approach*, Prentice Hall Series in Artificial Intelligence, Prentice Hall, 1995.

[Ros58] Rosenblatt F.: *The perceptron: A probabilistic model for information storage and organization in the brain*, in: Psychological Review, No. 65, 1958, pp. 386-408.

[SYC91] Sauer T., Yorke J. A. and Casdagli M.: *Embedology*, J. Stat. Phys., Vol. 65, 1991, pp. 579-616.

[San99] Santiso J.: *Analysts Analyzed: a socio-economic approach to financial and emerging markets*, in: International Political Science Review, Vol. 20, No. 3, July 1999, pp. 307-31.

[SvR96] Schmidt-von Rhein Andreas: *Die Moderne Portfoliotheorie im praktischen Wertpapiermanagement, Eine theoretische und empirische Analyse aus Sicht privater Kapitalanleger*, Reihe: Portfoliomanagement, Steiner M. [Ed.], No. 4, Uhlenbruch Verlag, Bad Soden / Ts., 1996.

[Sha64] Sharpe, F.:*A Simplified Model for Portfolio Analysis*, Management Science, Vol. 9, 1963, pp. 277-293.

[Shl00] Shleifer A.: *Inefficient Markets, An Introduction to Behavioral Finance*, Clarendon Lectures in Economics, Oxford University Press, New York, 2000.

[Shi97] Shiller R. J.: *Human Behavior and the Efficiency of the Financial System*, in: Handbook of Macroeconomics, Vol. 1, 1997, pp. 1305-40.

[SC00] Sima J. and Cervenka J.: *Neural Knowledge Processing in Expert Systems*, in: I. Cloete and J. M. Zurada [Eds.]: *Knowledge-Based Neurocomputing*, MIT Press, Cambridge, Massachusets, 2000, pp. 419-66.

[Smi93] Smith M.: *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold, New York, 1993.

[SJ98] Sornette D. and Johansen A.: *A hierarchical model of financial crashes*, in: Physica A, Vol. 261, 1998, pp. 581-598.

[Spe00] Sperduti A.: *A Tutorial on Neurocomputing of Structures*, in: I. Cloete and J. M. Zurada [Eds.]: *Knowledge-Based Neurocomputing*, MIT Press, Cambridge, Massachusets, 2000, pp. 118-52.

[Ste97] Steiglitz K. and O'Callaghan L. I.: *Microsimulation of Markets and Endogenous Price Bubbles*, 3rd International Conference on Computing in Economics and Finance, Stanford, Paper is available at *http://bucky.stanford.edu/cef97/papers/Steiglitz/paper.ps*, 1997, p. 1-4.

[Tak81] Takens F.: *Detecting strange attractors in turbulence*, in: D. A. Rand and L. S. Young, editors, *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics, Vol. 898, Springer, 1981, pp. 366-381.

[Tes00] Tesfatsion L.:*Agent-Based Computational Economics: A Brief Guide to the Literature*, Discussion Paper, Economics Dpt., Iowa State University, Jan. 2000, Reader's Guide to the Social Sciences, Fitzroy-Dearborn, London, 2000.

[Tom95] Tomassini, M.: *A survey of genetic algorithms.* in: D. Stauffer, editor, Annual Reviews of Computational Physics, Vol. III, World Scientific, 1995, pp. 87-118.

[TL80] Tong H. and Lim K. S.: *Threshold Autoregression, Limit Cycles and Cyclical Data*, in: Journal of The Royal Statistical Society, Series B, Vol. 4, 1980, pp. 245-292.

[Top91] Topol R.: *Bubbles and volatility of stock prices: effect of mimetic contagion*, in: Economic Journal, Vol. 101, 1991, pp. 786-800.

[TNZ97] Tresp V., Neuneier R. and Zimmermann H. G.: *Early Brain Damage*, in: Advances in Neural Information Processing Systems (NIPS 1996), Mozer M., Jordan M. and Petsche Th. [Eds.], MIT Press, 1997, pp. 669-675.

[Vel95] Velmans M.:*Theories of Consciousness*, in: M. A. Arbib [Ed.]: *The Handbook of Brain Theory and Neural Networks*, MIT-Press, 1995, pp. 247-50.

[WHD02] Wan H. A., Hunter A. and Dunne P.: *Autonomous Agent Models of Stock Markets*, in: Artificial Intelligence Review, Vol. 17, No. 2, 2002, pp. 87-128.

[Wei90] Wei W. S.: *Time Series Analysis: Univariate and Multivariate Methods*, Addison-Wesley Publishing Company, N.Y., 1990.

[Weid00] Weidlich, W.: *Sociodynamics: a Systematic Approach to Mathematical Modelling in the Social Sciences*, Harwood Academic Publishers, Overseas Publisher Assoc., Amsterdam, 2000.

[WZ98] Weigend A. S. and Zimmermann H. G.: *Exploiting local relations as soft constraints to improve forecasting*, in: Computational Intelligence in Finance, Vol. 6, No. 1, 1998, pp. 14-23.

[WRH90] Weigend A. S., Rumelhart D. E. and Huberman B. A.: *Generalization by Weight-Elimination with Application to Forecasting*, in: Advances in Neural Information Processing Systems (NIPS 90), Vol. 3, Lippmann R. P., Moody J. and Touretzky D. S. [Eds.], Morgan Kaufmann, Redwood City, CA, 1990, pp. 875-882.

[Wer74] Werbos P. J.: *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD. Thesis, Harvard University, 1974.

[Wer94] Werbos P. J.: *The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting*, Wiley, New York, 1994.

[WidH60] Widrow B., Hoff, M.E.: *Adaptive switching circuits*, in: IRE WESCON Convention Record, part 4, 1960, pp. 96-104.

[Wie48] Wiener N.: *Cybernetics: Or Control and Communication in the Animal and the Machine*, Wiley, New York, 1948.

[WiZi89] Williams R. J. and Zipser D.: *A learning algorithm for continually running fully recurrent neural networks*, in: Neural Computation, Vol. 1, No. 2, 1989, pp. 270-280.

[Yan00] Yang J.: *Price Efficiency and Inventory Control in Two Interdealer markets: An Agent-Based Approach*, in: D. Sallach and Th. Wolsko [Eds.], Proceedings of the Workshop on Simulation of Social Agents, Architectures and Institutions, University of Chicago, 2000, pp. 52-61.

[Yan99] Yang J.: *Heterogeneous Beliefs, Intelligent Agents, and Allocative Efficiency in an Artificial Stock Market*, Society for Computational Economics, Series of Computing in Economics and Finance, No. 612, 1999.

[YH95] Youssefmir M. and Huberman B. A.: *Clustered Volatility in Multiagent Dynamics*, Working Paper, Santa-Fe Institute, 95-05-051, 1995, p. 1-23.

[ZaRe99] Zapranis A. D. and Refenes A. P.: *Principles of Neural Model Identification, Selection and Adequacy: with Applications to Financial Econometrics*, Perspectives in Neuro Computing, Springer, London 1999.

[ZGTN02] Zimmermann H. G., Neuneier R., Grothmann R. and Tietz Ch.: *Market Modeling based on Cognitive Agents*, submitted to the International Conference on Artificial Neural Networks (ICANN), August 2002, University of Madrid, forthcoming.

[ZNG02] Zimmermann H. G., Neuneier R. and Grothmann R.: *Undershooting: Modeling Dynamical Systems by Time Grid Refinements*, in: M. Verleysen [Ed.], Proceedings of the European Symposium on

Artificial Neural Networks (ESANN) 2002, Bruges, Belgium, pp. 395-400.

[ZGT02] Zimmermann H. G., Grothmann R. and Tietz Ch.: *Yield Curve Forecasting by Error Correction Neural Networks and Partial Learning*, in: M. Verleysen [Ed.], Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2002, Bruges, Belgium, pp. 407-12.

[ZN01] Zimmermann H. G. and Neuneier R.: *Neural Network Architectures for the Modeling of Dynamical Systems*, in: A Field Guide to Dynamical Recurrent Networks, Eds. Kolen, J.F.; Kremer, St.; IEEE Press, 2001, pp. 311-350.

[ZN00a] Zimmermann H. G. and Neuneier R.: *Combining State Space Reconstruction and Forecasting by Neural Networks*, in: G. Bol, G. Nakhaeizadeh and K. H. Vollmer [Eds.]: *Datamining und Computational Finance*, Ergebnisse des 7. Karlsruher Ökonometrie-Workshops, Wissenschaftliche Beiträge 174, Physica-Verlag, pp. 259-67.

[ZNG01a] Zimmermann H. G., Neuneier R. and Grothmann R.: *Modeling of Dynamical Systems by Error Correction Neural Networks*, in: Modeling and Forecasting Financial Data, Techniques of Nonlinear Dynamics, Eds. Soofi, A. and Cao, L., Kluwer Academic Publishers, 2002, pp. 237-263.

[ZNG01b] Zimmermann H. G., Neuneier R. and Grothmann R.: *An Approach of Multi-Agent FX-Market Modeling based on Cognitive Systems*, in: Dorffner, G., Bischof, H. and Hornik, K. [Eds.]: Proceedings of the International Conference on Artificial Neural Networks (ICANN), Spinger, Lecture Notes in Computer Science, Vol. 2130, Vienna, 2001 pp. 767-774.

[ZNG01c] Zimmermann H. G., Neuneier R. and Grothmann, R.: Active Portfolio-Management based on Error Correction Neural Networks. in: Advances in Neural Information Processing Systems (NIPS 2001), Vancouver, Canada 2001, forthcoming.

[ZNG01d] Zimmermann, H. G., Neuneier, R. and Grothmann, R. (2001) *Multi-Agent Modeling of Multiple FX-Markets by Neural Networks*, in: IEEE Transactions on Neural Networks, Special issue, Vol. 12, No. 4, IEEE Press., 2001, pp. 735-743.

[ZNG00b] Zimmermann H.G., Neuneier R., Grothmann, R.: *Multi Agent Market Modeling of Foreign Exchange Rates*, in: Advances in Complex Systems (ACS), special issue, Hermes Science Publ., Oxford, 2001.

[ZNG00c] Zimmermann H. G., Neuneier R. and Grothmann R.: *Modeling of the German Yield Curve by Error Correction Neural Networks.* in: Leung, Kwong, Chan, Lai-Wan and Meng, Helen, Eds. Proceedings Intelligent Data Engineering and Automated Learning, Hong Kong, 2000, pp. 262-267.

[ZNG00d] Zimmermann H. G., Neuneier R. and Grothmann R.: *Multi-Agent FX-Market Modeling by Neural Networks*, internal discussion paper, SIEMENS AG, Munich, 2000, unpublished.

[ZNG00e] Zimmermann H. G., Neuneier R. and Grothmann R.: *A Cognitive Systems Approach of Multi-Agent FX-Market Modeling*, internal discussion paper, SIEMENS AG, Munich, 2001, unpublished.

[Zim94] Zimmermann H. G.: *Neuronale Netze als Entscheidungskalkühl.* in: Rehkugler, H. and Zimmermann, H. G. (Eds.): Neuronale Netze in der Ökonomie, Grundlagen und wissenschaftliche Anwendungen, Vahlen, Munich, 1994.

[Zim89] Zimmermann H. G.: *Application of the Boltzmann Machine in Economics*, in: Rieder, H.; Gessner, P.; Peyerimhoff, A. and Radermacher F. [Eds.], Methods of Operations Research, Vol. 63, Hain, Ulm, 1989, pp. 491-506.

[ZW97] Zimmermann H. G. and Weigend A. S.: *Representing dynamical systems in feed-forward networks: A six layer architecture*, in: A. S. Weigend, Y. Abu-Mostafa, and A.-P. N. Refenes, editors, *Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM-96)*, World Scientific, Singapore, 1997, p. 289-306.