Computerlinguistik und Künstliche Intelligenz

Modellierung wirtschaftlicher Prozesse mit Neuronalen Netzen

MAGISTERARBEIT
ZUR ERLANGUNG DES
MAGISTER ARTIUMS

im Fachbereich
Sprach- und Literaturwissenschaft
der Universität Osnabrück

vorgelegt von:

Maria Ananieva
aus Odessa (Ukraine) (Geburtsort)
30 (Jahreszahl)

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS ..... **vi**

TABELLENVERZEICHNIS ..... **viii**

EINLEITUNG ..... **ix**

1 ZIELSETZUNG **1**

1.1 MOTIVATION 1
1.2 DEFINITIONEN 2
1.3 PROBLEMSTELLUNG 4

2 GRUNDLAGEN NEURONALER NETZE **5**

2.1 THEORETISCHE GRUNDLAGEN 5
2.1.1 Bestandteile *5*
2.1.2 Standardtopologien *6*
2.1.3 Approximationseigenschaften *7*
2.1.4 Lernen *7*
2.1.5 Sensitivitätsanalyse *7*
2.2 VERWENDUNG IN DER BETRIEBSWIRTSCHAFT 7
2.2.1 Komplexität ökonomischer Modelle *7*
2.2.2 Anwendungsgebiete *7*
2.2.3 Vorgehensweise beim Einsatz Neuronaler Netze *7*

3 GRUNDLAGEN DER LINEAREN REGRESSION **7**

3.1 EINFÜHRUNG 7
3.2 BEURTEILUNG EINES LINEAREN MODELLS 7
3.2.1 Beurteilung des i*–ten Regressoren* $x^{(i)}$ *7*
3.2.2 Beurteilung des gesamten Modells *7*
3.3 ERSTELLUNG EINES LINEAREN MODELLS 7
3.3.1 Berücksichtigung der zeitlich verlagerten Auswirkungen *7*
3.3.2 forward– , backward– *und* stepwise–solution *–Algorithmen* *7*

***4 EX–POST–*ANALYSE** **7**

4.1 FEHLERMAßE 7
4.2 THEIL-SCHER UNGLEICHHEITSKOEFFIZIENT 7

5 ENTWURF EINES MODELLIERUNGS– UND PROGNOSESYSTEMS **7**

5.1 PROBLEMMERKMALE 7

5.2 BENUTZERMODELL 7
5.3 PHASEN DES PROGNOSEPROZESSES 7
5.3.1 Erstellung eines linearen Modells 7
5.3.2 Erstellung eines neuronalen Modells 7
5.3.3 Validierung eines Modells und Erstellung der Prognose 7

6 PROGNOSESYSTEM DER UNIVERSITÄT OSNABRÜCK 7

6.1 FUNKTIONALITÄT UND BESTANDTEILE 7
6.1.1 Simulation Neuronaler Netze 7
6.1.2 Sensitivitätsanalyse Neuronaler Netze 7
6.1.3 Dateiformate 7
6.2 ERWEITERUNG 7
6.2.1 C++ 7
6.2.2 Perl 7

7 EMPIRISCHE UNTERSUCHUNGEN 7

7.1 EMPIRISCHE DATENBASIS 7
7.1.1 Prognostizierte Zeitreihen 7
7.1.2 Exogene Zeitreihen 7
7.1.3 Periodizität 7
7.1.4 Konvertierung und Organisation der Daten 7
7.2 EINSATZ DER LINEAREN REGRESSION 7
7.2.1 Korrelationsanalyse 7
7.2.2 Erstellung der linearen Modelle 7
7.2.3 ex–ante–*Beurteilung der linearen Modelle* 7
7.2.4 ex–post–*Beurteilung der linearen Modelle* 7
7.3 EINSATZ NEURONALER NETZE 7
7.3.1 Sukzessive Optimierung 7
*7.3.2 Optimierung mit der „*best–first*"–Suche* 7
7.3.3 Optimierung durch Selektion 7
7.4 VERGLEICHSANALYSE 7

8 ZUSAMMENFASSUNG UND AUSBLICK 7

8.1 DATEN– VS. PROGNOSEQUALITÄT 7
8.2 VERGLEICH NEURONALER NETZE UND LINEARER REGRESSION 7
8.3 KI METHODEN ALS INSTRUMENT DER KONNEKTIONISTISCHEN MODELLBILDUNG 7
8.3.1 Heuristik 7
8.3.2 Selektion 7

ANHANG 7

ANLAGE A: KLASSEN DER BIBLIOTHEK „`REGRESSION`" 7

- CORRELATION 7
- MODEL 7

- PROGNOSE 7
- READER 7
- SYNCHRONIZER 7
- REGRESSION 7
- TIMESERIES 7

ANLAGE B: C++–MODULE I

- MATCORLIST 7
- MATCORRELATION 7
- MATERROR 7
- MATMODEL 7
- MATREGRESSION 7

ANLAGE C: PERL–MODULE 7

- AGGREGATE.PL 7
- CONVERTFORREADER.PL 7
- MATCORRELATION.PL 7
- MATERROR.PL 7
- MATMODEL.PL 7
- MATREGRESSION.PL 7
- OPTIMIZETOP.PL 7
- OPTIMIZEINPUT.PL 7
- PLOT_COR.PL 7
- PLOT_SELECTION.PL 7
- SCALE_FOR_SNNS.PL 7
- SEARCHNN.PL 7
- SELECTMODELNN.PL 7

ANLAGE D: EMPIRISCHE DATEN 7

ENDOGENE UMSATZZEITREIHEN 7

EXOGENE WIRTSCHAFTLICHE ZEITREIHEN VOM STATISTISCHEN BUNDESAMT 7

Statistik der Arbeitslosigkeit ab 1960 7

Statistik der Baufertigstellungen ab 1952 7

Statistik der Wirtschaftsrechnungen ausgewählter privater Haushalte ab 1950 7

Statistik der Bausparkassen ab 1951 7

Statistik der saisonbereinigte Wirtschaftszahlen der Deutschen Bundesbank ab 1960 7

Statistik der Indizes des Auftragseingangs und des Umsatzes des verarbeitende Gewerbe ab 1977 7

Statistik vom Preisindex für den Wareneingang des produzierenden Gewerbes (ab 1985) 7

Statistik der Außenhandel ab 1950 7

ANLAGE E: PROGNOSENGRAPHIKEN 7

ANLAGE F: BEISPIEL EINES SELEKTIONSABLAUFS 7

LITERATURVERZEICHNIS 7

VERPFLICHTUNGSERKLÄRUNG 7

Abbildungsverzeichnis

Abbildung 1: Ein künstliches Neuron ... 5
Abbildung 2: Ein *Multilayer Perceptron* (*MLP*) Netz ... 7
Abbildung 3: Ein Jordan–Netz ... 7
Abbildung 4: Ein Elman–Netz ... 7
Abbildung 5: Definition eines universalen Approximierers ... 7
Abbildung 6: Sensitivitätsanalyse, generischer Algorithmus ... 7
Abbildung 7: Dimensionen der Komplexität ... 7
Abbildung 8: Vorgehensweise bei der Entwicklung Neuronaler Netze ... 7
Abbildung 9: Methode der kleinsten Quadrate (*oridinary least squares*) ... 7
Abbildung 10: Evaluation der Regressionsfunktion $\mathbf{y=b_0+b_1x}$ ... 7
Abbildung 11: *forward–solution*–Algorithmus der Erstellung eines linearen Modells ... 7
Abbildung 12: Modell– und Prognoseerstellung beim erweiterten Benutzereinfluß ... 7
Abbildung 13 : Modell– und Prognoseerstellung ohne den Benutzereinfluß ... 7
Abbildung 14: Optimierung des Netzinputs, Algorithmus ... 7
Abbildung 15: Direkte Nachbarn eines Knotens ... 7
Abbildung 16: Modifizierung der Dimension *X* (Netzinput), Algorithmus ... 7
Abbildung 17: Konzepte der Umgebung des Knotens 0 in mehrdimensionalem Raum ... 7
Abbildung 18: Optimierung neuronaler Modelle mit der „*best–first*"-Suche, Algorithmus ... 7
Abbildung 19: Evolutionszyklus ... 7
Abbildung 20: Das Prognosesystem ... 7
Abbildung 21: Erweiterung des Prognosesystems ... 7
Abbildung 22: Klassenbaum von `MatClass` ... 7
Abbildung 23: Architektur von `matcorrelation` ... 7
Abbildung 24: Architektur von `matmodel` ... 7
Abbildung 25: Architektur von `matregression` ... 7
Abbildung 26: Architektur von `materror` ... 7
Abbildung 27: Architektur eines Perl–Adapters ... 7
Abbildung 28: Beispiel einer Datei mit der Modellbeschreibung („`komps`") ... 7

Abbildung 29: Ausschnitt der Ausgabe von `optimizeTop.pl` 7

Abbildung 30: Ausschnitt der Ausgabe von `optimizeInput.pl` 7

Abbildung 31: Schema der Untersuchungen 7

Abbildung 32: Verlauf der Kurve des Umsatzes nach Material Buche 7

Abbildung 33: Visualisierung der Korrelation mit `plot_cor.pl` 7

Abbildung 34: Vergleich von Methoden neuronaler Modellbildung (Durchschnittswerte) . 7

Abbildung 35: Vergleich der Prognosemethoden: Neuronale Netze, Lineare Regression und „naiv“. 7

Abbildung 36: Training und Prognose von „Mittel“ und „Frankreich“ 7

Abbildung 37: Ausschnitt aus der Ausgabedatei von `selectModelNN.pl` 7

Abbildung 38: Durchschnittliche Fitneß bei der Evolution der Modelle für „hell“ 7

Tabellenverzeichnis

Tabelle 1: Dimensionen des Suchraumes ... 7

Tabelle 2: Exogene Zeitreihen ... 7

Tabelle 3: Hierarchie des Projektes BCAS ... 7

Tabelle 4: Ergebnisse der Korrelationsanalyse ... 7

Tabelle 5: *ex–ante*–Analyse der linearen Modelle ... 7

Tabelle 6: *ex–post*–Analyse der linearen Modelle ... 7

Tabelle 7: Sukzessive Erstellung neuronaler Modelle, Zwischenergebnisse ... 7

Tabelle 8: Statistiken zur Optimierung des Netzinput ... 7

Tabelle 9: Sukzessive Erstellung neuronaler Modelle: Testfehler und ex–post–Analyse ... 7

Tabelle 10: Neuronale Modellbildung mit der „*best–first*"–Suche: Testfehler und ex–post–Analyse ... 7

Tabelle 11: Vergleich der „*best–first*"–Suche bei verschiedenen Umgebungskonzepten ... 7

Tabelle 13: Selektionsablauf bei verschiedenen Endogenen ... 7

Tabelle 14: Neuronale Modellbildung durch Selektion: Testfehler und *ex–post*–Analyse ... 7

Tabelle 15: Vergleich von Erstellungsmethoden konnektionistischer Modelle: MAPE, RMSE, *Theil*–scher Koeffizient ... 7

Tabelle 16: Vergleich von Neuronalen Netzen und Linearer Regression als Prognose-methoden ... 7

Tabelle 17: Öffentliche Methoden der Klasse Correlation ... 7

Tabelle 18: Öffentliche Methoden der Klasse Model ... 7

Tabelle 19: Öffentliche Methoden der Klasse Prognose ... 7

Tabelle 20: Öffentliche Methoden der Klasse Reader ... 7

Tabelle 21: Öffentliche Methoden der Klasse Synchronizer ... 7

Tabelle 22: Öffentliche Methoden der Klasse Regression ... 7

Tabelle 23: Öffentliche Methoden der Klasse Timeseries ... 7

Einleitung

Neuronale Netze bekommen im Rahmen der wirtschaftswissenschaftlichen Fragestellungen wie Prognose–, Klassifikations– und Optimierungsaufgaben ein wachsendes Interesse. Die dadurch zunehmende Zahl der Benutzer braucht eine intelligente Unterstützung, die sowohl durch die Verwendung moderner Prinzipien des *Software Engineering* als auch durch Automatisierung von Prozessen geleistet werden kann.

Im Rahmen der Arbeit wurde ein Prognosesystem mit einer erweiterten Unterstützung des Benutzers durch benutzerdefinierte Automatisierung der Modellbildung und Prognose wirtschaftlicher Prozesse entworfen, und folgende Bestandteile des Systems wurden implementiert:

- Auswahl der Prognosemethode (Lineare Regression oder Neuronale Netze);
- Ermittlung maximaler Prognoselänge und maximaler Exogenenzahl;
- Auswahl der Exogenen aus einer großen Anzahl von Zeitreihen unter der Berücksichtigung der verzögerten Auswirkungen;
- Erstellung und Beurteilung eines Modells;
- Optimierung des neuronalen Modells mit Methoden der *Künstlichen Intelligenz* (*KI*).

Nach der Implementation der verwendeten Algorithmen auf Basis des Prognosesystems der Universität Osnabrück [Thi98], wurde eine Serie empirischer Untersuchungen durchgeführt, wobei mehrere reale Umsatzzeitreihen mit den erstellten Instrumenten prognostiziert wurden. Diese Experimente zeigten, daß kurze und stark schwankende Umsatzzeitreihen in der Regel schlecht vorhersagbar sind, dennoch erzielen neuronale Modelle bei der Prognose solcher Zeitreihen im Durchschnitt bessere Ergebnisse als lineare. Der Vergleich des „*best–first*"–Suchalgorithmus und des Selektionsverfahrens zur Erstellung neuronaler Modelle zeigte, daß sie vergleichbareErgebnisse erzielen.

Nach detaillierterer Beschreibung des Hintergrundes und der Fragestellung der Arbeit im Kapitel 1 folgen Kapitel 2 und 3, die kurze Einführungen in die Grundlagen Neuronaler Netze bzw. der Linearen Regression enthalten. Im Kapitel 4 werden Methoden der *ex–post*–Analyse der Prognosegüte erläutert. Entwurf des Modellierungs– und Prognosesystems und im Rahmen der Arbeit realisierte Implementationen werden in Kapiteln 5 und 6 beschrieben. Kapitel 7 enthält Beschreibungen und Bewertungen der Daten und der empirischen Untersuchungen. Die Gliederung dieses Kapitels entspricht dem Gang der Experimente. Zum Schluß werden im Kapitel 8 Ergebnisse und Überlegungen kurz zusammengefaßt.

Im Anhang befinden sich Spezifikationen der implementierten Programme und Klassen, die Information über benutzte empirische Daten und Prognoseprahpiken.

1 Zielsetzung

1.1 Motivation

Die Verwendung von Computern in allen Bereichen des gesellschaftlichen Lebens wird immer breiter. Das führt zu neuen Problemen: Einerseits verringert sich das Niveau des Expertenwissens eines durchschnittlichen Benutzers, und andererseits wächst die Anzahl und Mächtigkeit (und damit auch die Komplexität) moderner Computeranwendungen exponentiell. Beim riesigen Informationsüberfluß eines Benutzers ist die Unterstützung durch ein traditionelles *textsensitives* Hilfssystem nicht mehr ausreichend, notwendigerweise gehören zur Konzipierung moderner Computeranwendungen auch folgende AnsäExotze:

- **Ergonomischer Ansatz**: Eine intuitiv strukturierte, benutzerfreundliche Oberfläche;
- **Didaktischer Ansatz**: Das Einbetten in die Anwendung eines lehrenden Hilfesystems (*ITS*, *Intelligent Tutoring Systems*), das Expertenwissen mit didaktisch begründeten Mitteln an Benutzer transferiert;
- **Problemorientierter Ansatz**: Automatisierung sowohl einfacher, sich wiederholender Prozesse als auch komplizierter Vorgänge.

All dieses betrifft im Einzelfall auch Anwendungen Neuronaler Netze in der Wirtschaftswissenschaft, die in den letzten Jahren beinahe explosionsartig zunehmendes Interesse gewinnen, insbesondere für wirtschaftliche Prognose– und Klassifikations–aufgaben in Finanz– , Marketing– und Produktionsbereichen [Ko95].

Der Gegenstand dieser Arbeit kann dem letzten Punkt der oberen Aufzählung zugeordnet werden: Es wird versucht, einige Teile des Prozesses der Modellierung vs. Analyse wirtschaftlicher Prozesse zu automatisieren.

Ökonomische Analyse ist ein komplizierter mehrstufiger Prozeß, der sowohl Expertenwissen als auch „Fingerarbeit“ verlangt. Besonders sinnvoll wäre die Unterstützung eines Benutzers bei der Modellerstellung im Fall vieler potentieller Exogener (externer Einflußgrößen) und der Berücksichtigung verzögerter Zusammenhänge. Diese Phase der wirtschaftlichen Analyse wird üblicherweise von Hand aufgrund einer Korrelationsanalyse oder des Expertenwissens gemacht und besteht aus sich mehrmals wiederholender Prozesse der Erstellung, Parametrisierung, Beurteilung und Vergleich verschiedener Modelle[1]. Die Systematisierung und Implementation zumindest eines Teiles dieses Vorganges wäre vom Vorteil. Bei linearen Modellen können bereits existierende Verfahren der Bestimmung der besten Exogenen implementiert und in die

[1] Genaue Definitionen der Modelle siehe in Abschnitt 1.2.

Anwendung eingeschlossen werden. Bei neuronalen Modellen, wo es im Unterschied zu linearen Modellen keine Kennzahlen zur Beurteilung der Exogenen gibt, wäre der Einsatz der auf der *Sensitivitätsanalyse*[2] Neuronaler Netze basierenden heuristischen Suche denkbar.

Die meist zeitkritische Phase bei der *konnektionistischen* Modellierung ist die *Netzkonfigurierung*. Das Ziel dieses Vorganges ist es, ein Netz zu finden, das bestmögliche Ergebnisse bei der Lösung der Aufgabe erreicht. Während dieser Phase sollen optimale Einstellungen mehrerer Parameter des Neuronalen Netzes gefunden werden. Wie es in [Bra97], Abschnitt 2.1.4 erläutert wird, ist dieses Optimierungsproblem bei mehrschichtigen Neuronalen Netzen nicht effizient lösbar (NP–vollständig). Aus diesem Grund wird dafür meistens eine rudimentäre „*trial–and–test*" Methode verwendet, obwohl hier der Einsatz von Heuristiken und somit auch von Suchalgorithmen oder von *Evolutionären* Verfahren, wie z. B. *Selektion*, gerechtfertigt ist.

Dabei ist anzunehmen, daß solche zu optimierenden Parameter konnektionistischer Modelle wie Parameter des dazugehörigen Neuronalen Netzes und die Exogenen (der Netzinput) zusammenhängen. Deswegen werden diese Modellteile nicht wie üblich sukzessiv (zuerst die Bestimmung der Exogenen und dann die Verbesserung des Netzes), sondern während des gleichen Optimierungsprozesses bestimmt.

Ergebnisse der empirischen Untersuchungen sowie Module, die im Rahmen dieser Arbeit entwickelt wurden, werden zukünftig im Projekt „Strategisches Prognose– und Planungssystem SPS MBI" der Softwarefirma „MBI Software Company" verwendet.

1.2 Definitionen

Im Zentrum der Arbeit steht der Begriff eines Modells. Die allgemeine Definition eines mathematischen Modells lautet: „Ein **mathematisches Modell** ist eine vereinfachende und idealisierende mathematische Beschreibung eines meist realen Objektes. Diese Beschreibung soll ein in sich vollständiges widerspruchsfreies System von Gleichungen oder Relationen sein. Strukturen des Objektes entsprechen dabei mathematischen Strukturen. Zweck und Form der Beschreibung werden dabei vom Modell–Konstrukteur vorgegeben." [Behn83].

Für die Erläuterung der Problemstellung und des Hintergrundes der Arbeit werden folgende Definitionen benutzt:

[2] Darunter wird partialanalytische Betrachtung der Signifikanz von einzelnen Inputzeitreihen des Neuronalen Netzes betrachtet (genauer siehe Abschnitt 6.1.2).

Def. 1.1 Eine **Zeitreihe** x sei eine Folge von Messungen einer von der Zeit abhängigen Variable $x(t)$ an aufeinanderfolgenden Zeipunkten $t_0, t_1, ..., t_n$ mit gleichen Perioden. Üblicherweise wird dabei eine größere Anzahl von Messungen vorausgesetzt. Eine Zeitreihe wird in Form eines reellen Vektors dargestellt:

$$x=(x(t_0), x(t_1), ..., x(t_n)) .$$

Def. 1.2 Ein **Indikator** x^i sei die um i Zeiteinheiten nach rechts (in die Zukunft) verschobene Zeitreihe x:

$$x^i=(x_{t+i},x_{t+i+1},...,x_{T+i}) , \ 0 \leq i ,$$

x wird Basiszeitreihe des Indikators x^i genannt und ist selbst ein Indikator:

$$x=x^0.$$

Def. 1.3 Eine **Endogene** sei eine zu erklärende (und zu prognostizierende) Zeitreihe. Im Kontext linearer Modelle wird die Endogene oft *Regressand* und im Kontext konnektionistischer Modelle *Output(–zeitreihe)* genannt.

Def. 1.4 Eine **Exogene** oder eine **Einflußgröße** sei ein Indikator, der die Endogene bewirkt und sie dadurch erklärt. Im Kontext linearer Modelle wird eine Exogene häufig *Regressor* und im Kontext der konnektionistischen Modelle *Input(–zeitreihe)* genannt.

Im Kontext der hier betrachteten Prognoseaufgaben wird eine spezifischere als die o. g. Definition eines Modells verwendet.

Def. 1.5 Ein **Modell** sei eine Zusammensetzung folgender Parameter:

1. eine Endogene;
2. maximale Exogenenzahl K;
3. eine bis K Exogene;
4. maximale Prognoselänge.
5. Modellierungs– und Prognosemethode (Neuronale Netze oder Lineare Regression), anhand deren zwischen *linearen* und *neuronalen* (oder *konnektionistischen*) Modellen unterschieden wird;
6. Im Fall der neuronalen Modelle Parameter des verwendeten Neuronalen Netzes[3];

[3] Außer der Gewichte, die während des Training ermittelt werden.

7. mathematische Beschreibung der Zusammenhänge zwischen der Endogenen und den Exogenen: Bei linearen Modellen erfolgt es durch die Ermittlung der linearen Koeffizienten des verwendeten linearen Systems und bei neuronalen Modellen durch eine eindeutige Beschreibung des verwendeten trainierten Neuronalen Netzes.

1.3 Problemstellung

Das Thema der Arbeit kann in folgende Fragestellungen unterteilt werden:

1. Automatische Erstellung und Beurteilung eines Modells für eine vorgegebene Endogene mit dem Ziel ihrer Prognose.

Während der Modellerstellung sollen die in der Def. 1.1 aufgezählten Modellkomponenten bestimmt werden, insbesondere die ersten 5 ggf. 6 Parameter, die den 7. Parameter bestimmen und als ein „Skeleton" des Modells betrachtet werden können. Die Existenz einer großen Anzahl der potentiellen Exogenen muß in Betracht gezogen werden. Für die Beurteilung eines Modells muß eine *ex–ante*–Analyse der Güte sowohl des ganzen Modells als auch der einzelnen Exogenen durchgeführt werden, ggf. auch eine *ex–post*–Analyse der Prognose.

2. Verwendung der Methoden der KI für die konnektionistische Modellbildung

Für die Erstellung der neuronalen Modelle sollen Methoden der Künstlichen Intelligenz (Such– und *Evolutionäre* Algorithmen) eingesetzt werden. Anhand der Ergebnisse wird ihre Verwendbarkeit beurteilt.

3. Erweiterung des Prognosesystems der Universität Osnabrück um eine objektorientierte Implementation der Linearen Regression in C++.

Das System soll um folgende Funktionalitäten erweitert werden:

- Erstellung und Beurteilung eines linearen Modells.
- Erstellung der Prognose mit der Linearen Regression;

4. Verifizierung der Ergebnisse auf empirischen Umsatzdaten.

Im Rahmen der Arbeit müssen neun empirische wöchentlich erfaßte Umsatzzeitreihen automatisch prognostiziert werden. Spezifisch an diesen Zeitreihen ist ihre niedrige Quantität (ca. 200 Werte) und die Existenz der starken Schwankungen und Ausreißer. Im Rahmen der Untersuchungen sind folgende Fragestellungen zu beantworten:

- Beurteilung der Prognostizierbarkeit dieser Zeitreihen;
- Vergleich Neuronaler Netze und Linearer Regression als Prognosemethoden;
- Beurteilung der Verwendbarkeit von Methoden der KI für konnektionistische Modellbildung.

2 Grundlagen Neuronaler Netze

In diesem Kapitel werden sowohl einige Elemente der Theorie Neuronaler Netze als auch praktische Anwendungen Neuronaler Netze in der Wirtschaft erläutert.

2.1 Theoretische Grundlagen

2.1.1 Bestandteile

Def. 2.1 Ein (künstliches) **Neuron** j [4] ist ein Tripel $(a_j(t), f_{act}, f_{out})$ mit

- *Aktivierungszustand* $a_j(t)$, der den Grad der Aktivierung der Zelle angibt (t sei Zeitparameter);
- *Aktivierungsfunktion* f_{act}. Sie gibt an, wie sich ein neuer Aktivierungszustand $a_j(t+1)$ des Neurons aus der alten Aktivierung $a(t)$ und der *Netzeingabe* (*net input*) $net_j(t)$ ergibt, meist nach der allgemeinen Formel

  $$a_j(t+1) = f_{act}(a_j(t), net_j(t), \Theta_j),$$

  wobei Θ_j der Schwellenwert des Neurons j ist und f_{act} die Aktivierungsfunktion, die aus den angegebenen Parametern die neue Aktivierung berechnet. Die Funktion $net_j(t)$ ist für alle Neuronen des Netzes gleich definiert und wird im nächsten Abschnitt erläutert.
- *Ausgabefunktion* f_{out}. Der Output der Zelle j wird durch eine sogenannte Ausgabefunktion aus der Aktivierung der Zelle bestimmt.

  $$o_j = f_{out}(a_j(t)).$$

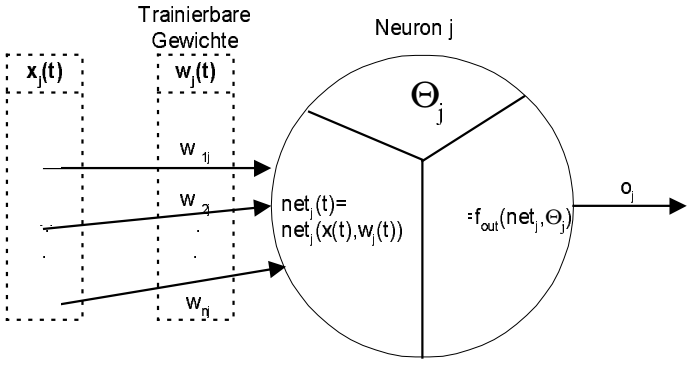


Abbildung 1: Ein künstliches Neuron

[4] Definition erfolgt nach [Zel94]. Jedes Neuron des Netzes wird für die Bequemlichkeit indiziert.

Als Ausgabefunktion f_{out} dienen üblicherweise Sigmoidfunktionen, am häufigsten:

- die Logistische Funktion $f(x) = \frac{1}{1+e^{-Tx}}$,
- der Tangens Hyperbolicus $\tanh(x) = \frac{e^{Tx} - e^{-Tx}}{e^{Tx} + e^{-Tx}}$.

wobei die reelle Zahl ***T*** *Temperaturkonstante* genannt wird.

Ein *Neuronales Netz* besteht nach [Zel94] im allgemeinen aus folgenden Komponenten:

- *Zellen* (*Neuronen, Prozessorelementen, units*);
- *Verbindungsnetzwerk* der Zellen. Ein Neuronales Netz kann als gewichteter gerichteter Graph angesehen werden, wobei die Kanten die gewichteten Verbindungen zwischen den Neuronen darstellen. Das Gewicht der Verbindung von Zelle ***i*** nach Zelle ***j*** wird üblicherweise durch w_{ij} bezeichnet. Die Matrix der Verbindungen aller Zellen (*Gewichtsmatrix*) wird mit ***W*** bezeichnet.
- *Propagierungsfunktion*. Sie gibt an, wie sich die Netzeingabe eines Neurons aus den Ausgaben der anderen Neuronen und den Verbindungsgewichten berechnet. Die Netzeingabe $net_j(t)$ berechnet sich am häufigsten nach

  $$net_j(t) = \sum_i o_i(t) w_{ij}.$$

  Seltener werden andere Funktionen verwendet wie z. B. die quadratische Funktion $net_j(t)=\Sigma o_i^2 w_{ij}$ oder auch beliebig komplexe Funktionen $net_j(t)= \Sigma\Phi(w_{ij},o_i)$.
- *Lernregel*. Die Lernregel ist ein Algorithmus, gemäß dem das Neuronale Netz lernt, für eine angegebene Eingabe eine gewünschte Ausgabe zu produzieren. Das Lernen erfolgt in Neuronalen Netzen meist durch die Modifikation der Stärke der Verbindungen als Ergebnis der wiederholten Präsentation von Trainingsmustern. An meisten Lernregeln wird dabei versucht, den Fehler zwischen erwarteter und tatsächlicher Ausgaben für alle Trainingsmuster zu minimieren.

2.1.2 Standardtopologien

Def. 2.1 Falls der Verbindungsgraph des Netzes keine Zyklen enthält, wird das Netz *vorwärtsgerichtet* genannt, enthält er Zyklen, so wird das Netz *rekurrent* oder *partiell rekurrent* genannt.

Am häufigsten werden in der Praxis Mehrschichtsnetzwerke von Neuronen verwendet. Dieser Netztyp wird oft als *Multilayer Perceptron (MLP)* bezeichnet. Falls die Information in diesem Netzt stets in eine Richtung fließt, wird es *vorwärtsgerichtet* oder *feedforward* genannt. Dabei können auch Verbindungen zwischen nicht benachbarten Schichten (*shortcuts)* verwendet werden. Neuronen solcher Netze können nach ihrer Position im Netz klassifiziert werden: Die Neuronen der Inputschicht (*Eingabeneuronen*) nehmen lediglich die Informationen aus der Umwelt auf. In jeder weiteren Schicht wird das Signal von

vorheriger Schicht verarbeitet, wobei Neuronen einer Schicht parallel und die Schichten dagegen sequentiell aktiviert werden. Häufig nennt man die Neuronen interner Schichten *verdeckte* Neuronen. Die Neuronen der letzten Schicht werden *Ausgabeneuronen* genannt.

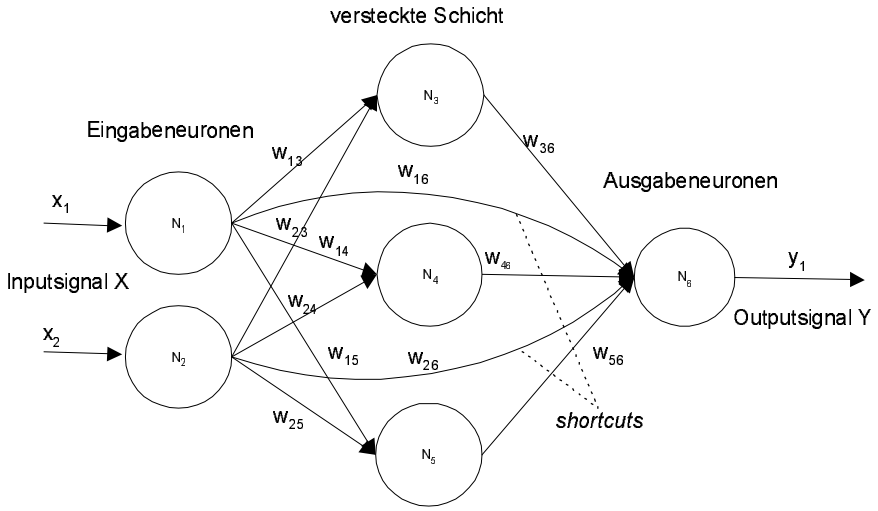


Abbildung 2: Ein Multilayer Perceptron (MLP) Netz

Der Ausgabevektor bestehend aus den Ausgaben dieser Neuronen gilt dann als der Output des gesamten Netzwerks.

Ein spezieller Fall der Inputdaten sind Zeitreihen. Eine Zeitreihe ist eine Folge von Mustern, die isoliert nicht betrachtet werden können, weil bei ihnen die zeitliche Reihenfolge eine wichtige Rolle spielt. In diesem Fall ist nicht nur ein Muster selbst relevant, sondern auch seine Position in der gesamten Mustersequenz. Wird eine Zeitreihe mit einem Neuronalen Netz verarbeitet, so hängt die Ausgabe des Netzes allgemein nicht nur von der aktuellen Eingabe, sondern auch von den vorangegangenen Eingaben ab. Für die Erkennung und die Klassifikation zeitveränderlicher Muster und für die Prognose von Zeitreihen sind *partiell rekurrente* Neuronale Netze besonders geeignet[5]. Diese Netze sind von vorwärtsgerichteten Netzen abgeleitet und besitzen zusätzlich Rückkopplungs–schleifen. Ein partiell rekurrentes Netz enthält nämlich spezielle *Kontextzellen*, mit denen ein Speichermechanismus (Gedächtnis) realisiert wird. Bei den sogenannten *Jordan–Netzen* (siehe Abbildung 3) erhalten die Kontextzellen Eingaben von Ausgabezellen; bei *Elman–Netzen* (siehe Abbildung 4) – von verdeckten Zellen. Die Ausgabe wird dann in das übrige Netz weitergeleitet.

Partiell rekurrente Netze ordnen sich zwischen vorwärtsgerichteten und vollständig rekurrenten Netzen an und haben den großen Vorteil, daß sie mit geringfügig modifizierten Lernverfahren für *feedforward*–Netze trainiert werden können, die effizienter sind, als Lernverfahren für beliebig rekurrente Netze.

[5] siehe [Zel94], Abschnitt 11 ab Seite 137.

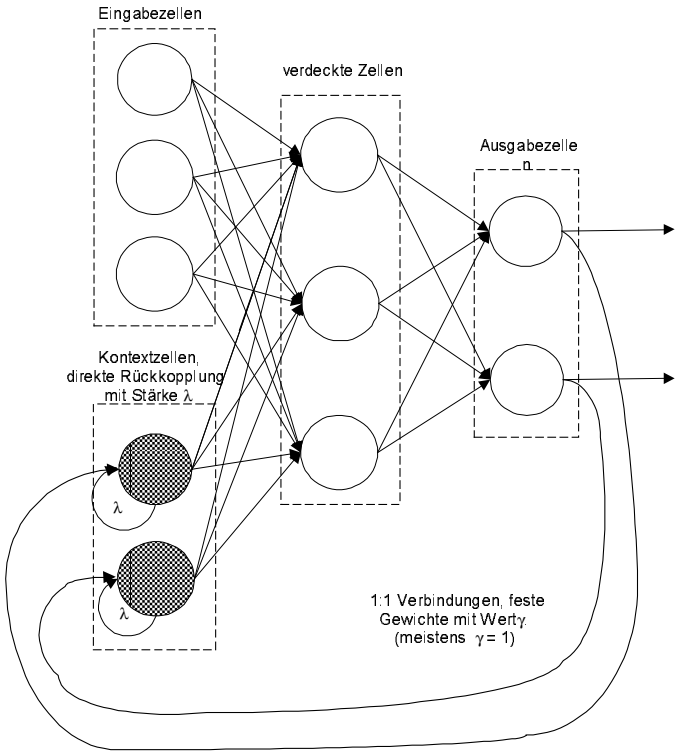


Abbildung 3: Ein Jordan–Netz

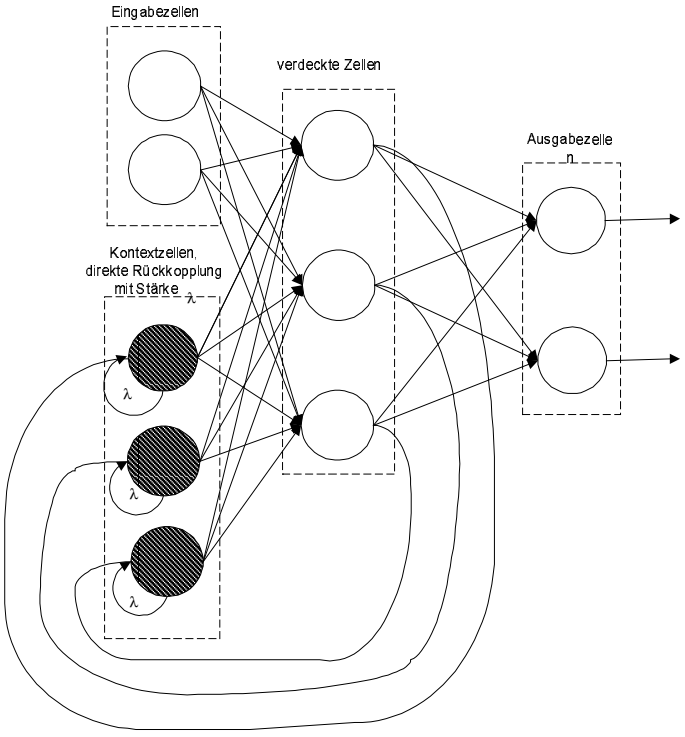


Abbildung 4: Ein Elman–Netz

2.1.3 Approximationseigenschaften

Es stellt sich die Frage nach den Approximationsfähigkeiten eines Neuronalen Netzes [Rüd96]. Die Überlegungen von Hornik[6] u. a. zeigen, daß jede stetige Funktion $\mathbf{f:[0.1]^n \rightarrow R^m}$ durch ein dreischichtiges Neuronales *feedforward*–Netz realisiert werden kann. Abbildung 5 stellt ein solchees Netwerk dar. Das an den Eingabeknoten angelegte Tupel $\mathbf{x=(x_1, ..., x_n)}$ führt über die Kantengewichte $\mathbf{w_{ij}^{(1)}}$ aus $\mathbf{R^n}$ bei den Knoten $\mathbf{K_m}$ der internen Schicht zu der Aktivität:

$$h_j = \sum_{i=1}^{n} w_{ij}^{(1)} x_i - \Theta_j,\ j = 1,...,\ m, i = 1,...,\ n\ ,$$

mit Θ_j als knotenspezifischem Schwellenwert.

S(x) sei eine monoton steigende Funktion, die die Aktivierungszustände $\mathbf{h_j}$ der Knoten $\mathbf{K_j}$ der internen Schicht in das Intervall [0,1] abbildet und für die gilt:

$\lim_{h_j \rightarrow -\infty} S(h_j) = 0$, $\lim_{h_j \rightarrow +\infty} S(h_j) = 1$. S wird als *squashing*–Funktion bezeichnet.

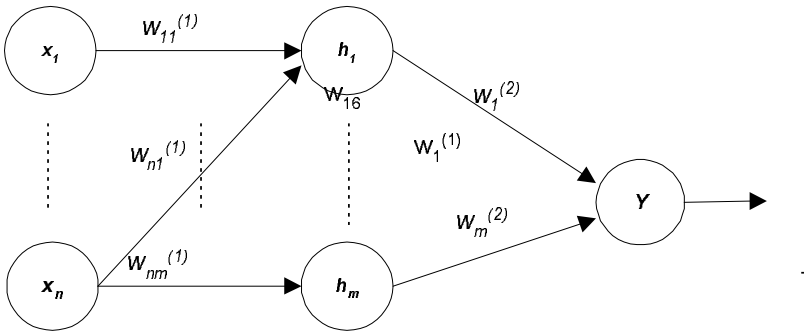


Abbildung 5: Definition eines universalen Approximierers

Zu gegebener Eingabe **x** ermittelt sich dann die Ausgabe **y** des betrachteten Neuronalen Netzes durch Bildung der gewichteten Summe:

$$y = \sum_{j=1}^{m} w_j^{(2)} S(h_j)\ .$$ Eine derartige Ausgabefunktion heißt *Sigmafunktion*.

Der Satz von Hornik u.a. besagt, daß zu jeder stetigen Funktion g, $g : C \subseteq \Re^n \rightarrow \Re$, C kompakt, ein Neuronales Netz obiger Struktur existiert, das $\boldsymbol{g}$ beliebig genau approximiert[4].

Da **S(x)** als vorgegeben gedacht ist, sind im konkreten Fall einer Approximationsaufgabe die Anzahl m der internen Knoten sowie die Gewichtungsparameter $\mathbf{w_{ij}^{(1)}}$, $\mathbf{w_{ij}^{(2)}}$ und die Schwellenwerte $\mathbf{\Theta_1, \Theta_2, ..., \Theta_m}$ zu bestimmen.

---

[6] Den Genauen Beweis siehe (nach der Angabe in [Zel94]) Hornik, K.; Stinchombe, M.; White, H. (1989) Multilayer feedforward networks are universal approximators. In: Neural Networks, 2, 1989, S. 359-366.

Die vorstehenden Überlegungen sagen etwas über die theoretischen Leistungsfähigkeiten Neuronaler Netze. Sie geben jedoch keine genaueren Hinweise darüber, wie die Parameter des Netzes aussehen müssen. Es handelt sich um eine reine Existenzaussage.

Dennoch folgt aus den Approximationstheoremen eine starke Motivation für die Verwendung Neuronaler Netze als Prognoseinstrument empirischer Daten.

2.1.4 Lernen

Eine wichtige Eigenschaft Neuronaler Netze ist die Tatsache, daß die Einstellung der Gewichte nicht *ad hoc* vorgenommen, sondern in einer Lernphase entsprechend dem Lernalgorithmus des Netzes gefunden wird. Dies geschieht in der Regel durch die Präsentation von Ein– und Ausgabebeispielen der zu approximierenden Funktion und durch einen wiederholten Korrekturschritt, in dem die Gewichte verändert werden. Wie die Gewichte korrigiert werden, ist vom verwendeten Lernalgorithmus abhängig. Während des Lernprozesses ist das Netz in eine geschlossene Rückkopplungsschleife eingebettet, die im Idealfall zur Lösung des Lernproblems führt. Lernalgorithmen werden in *überwachtes* („*Lernen mit einem Lehrer*“) und *unüberwachtes* Lernen unterteilt.

Überwachtes Lernen bezeichnet eine Methode mit folgenden Grundzügen: Für ein vorgegebenes Problem gibt es eine Menge von Paaren (Eingabe, *soll*–Ausgabe). Die Eingaben werden in ein Neuronales Netz eingegeben, das zum Beginn des Trainings mit zufällig gewählten Gewichten belegt ist. Die *ist*–Ausgabe des Netzes wird nach jeder Eingabe beobachtet. Falls die *ist*–Ausgabe sich von der *soll*–Ausgabe unterscheidet, werden die Gewichte des Netzes entsprechend dem Lernalgorithmus korrigiert. Dieses Verfahren wird angewendet, falls die zu einer Eingabe gehörende Ausgabe immer bekannt ist. Der populärste Lernalgorithmus für überwachtes Lernen ist *Backpropagation*, der die Suche des Minimums der Fehlerfunktion eines Neuronalen Netzes mit dem Verfahren des Gradientenabstiegs realisiert. Die überwachten Lernverfahren werden weiter in zwei Klassen eingeteilt:

Man spricht vom *Verstärkungslernen*, wenn bei jedem Test von einem Eingabe–Ausgabe–Paar nur beobachtet wird, ob das Netz eine korrekte Ausgabe liefert oder nicht und nur anhand dieses boolschen Wertes werden die Gewichte verändert.

Man hat ein *korrigierendes* Lernen, wenn darüber hinaus noch eine Fehlergröße ermittelt wird und die Änderungen der Gewichte so berechnet werden, daß der Fehler möglichst in einem Schritt behoben wird.

Unüberwachtes Lernen dagegen ist notwendig, wenn die zu einer Eingabe gehörende Ausgabe unbekannt ist (zum Beispiel bei selbstorganisierenden Karten von Kohonen[7]).

Ziel des Lernvorganges ist es jedoch nicht, die präsentierte Daten perfekt abzubilden, sondern eine optimale *Generalisierungsfähigkeit* des Netzes zu erreichen, d. h., es soll mit

[7] Siehe [Zel94], Kapitel 15.

unbekannten Daten, die zum Training nicht zur Verfügung stehen, gute Ergebnisse erzielen [CoM96]. Um eine Überanpassung (*overfitting, overlearning*) des Neuronalen Netzes beim Lernen zu vermeiden, wird die Methode der *Kreuzvalidierung* benutzt, bei der vorhandene Daten in zwei disjunkte Datenbasen aufgeteilt werden:

- Die *Trainingsdaten* dienen dem eigentlichen Lernvorgang, d.h. der Anpassung der Verbindungsgewichte.
- Die *Testdaten* werden nach jeder Lernepoche, d. h. nach einem einmaligen Durchlauf aller Datensätze, zur Berechnung des vom Netz produzierten Testfehler herangezogen.

Das Steigen des Testfehlers über einen längeren Zeitraum zeigt, daß die Generalisierungsfähigkeit des Netzes sinkt und daß der Lernvorgang beendet werden sollte.

In der Regel werden die Testdaten aus der Gesamtdatenbasis zufällig ausgewählt, um die Datenrepräsentativität möglichst wenig zu beeinflussen. Eine Garantie für eine optimale Lerndauer vermag das Kreuzvalidierungsverfahren jedoch nicht zu bieten, weil Einflüsse der Testdatensätze, insbesondere bei kleinen Datenmengen, nicht auszuschließen sind. Es bietet sich daher an, zusätzlich zu den Testdaten noch eine sogenannte Prognosedatenbasis (Validierungsmenge) zu extrahieren und diese dann nach Abschluß des Trainings zur nochmaligen Beurteilung der Generalisierungsfähigkeit des Netzes heranzuziehen.

2.1.5 Sensitivitätsanalyse

Neuronale Netze werden oft kritisiert, weil sie trotz der Bildung der Modell–zusammenhänge keine erklärende Funktion erfüllen können und als *Black–Box* betrachtet werden. Es gibt z. B. im Unterschied zu den linearen Modellen keine *t*–Statistiken, die Signifikanz einzelner Einflußgrößen bestimmen. Es ist aber möglich die Signifikanz der Inputzeitreihen eines Neuronalen Netzes durch eine partialanalytische Betrachtung zu beurteilen.

Für die Beschreibung des Algorithmus braucht man den Begriff des *gemittelten Inputvektors*: Bei mehrdimensionalem Input kann die Menge aller Muster als eine Matrix vorgestellt werden. Die Zeilen dieser Matrix sind vorverarbeitete und synchronisierte Zeitreihen, die Spalten – Inputvektoren, die in einer bestimmten Reihenfolge dem Netz präsentiert werden. Der Spaltenvektor, bestehend aus den Mittelwerten der Zeilen dieser Matrix, wird das *gemittelte Inputvektor* genannt.

Bei der Sensitivitätsanalyse bildet man einen gemittelten Inputvektor, gibt ihn ins trainierte Netz ein und berechnet den Output. Dann „wackelt“ man auf eine bestimmte Weise sukzessiv an allen Komponenten dieses Vektors und beobachtet die Veränderung des Netzoutput. Je mehr er sich beim „Wackeln“ der *i*–ten Komponente ändert, desto signifikanter ist die *i*–te Inputzeitreihe für das Modell.

Der Algorithmus wird genauer in Abbildung 6 beschrieben:

1. Bestimme den gemittelten Inputvektor
2. Trainiere das Netz und halte die Gewichte fest;
3. Berechne Output o des Netzes für gemitteltes Input–Vektor X;
4. Für jede Komponente x_i von X tue:
   5. Addiere ε_i * zu x_i und berechne Output o^+_i und Differenz zwischen o und o^+_i
   6. Subtrahiere ε_i von x_i und berechne Output o^-_i und Differenz zwischen o und o^-_i
   7. Setze x_i auf sein Mittelwert zurück

Abbildung 6: Sensitivitätsanalyse, generischer Algorithmus

2.2 Verwendung in der Betriebswirtschaft

Betriebswirtschaftliche Anwendungsmöglichkeiten Neuronaler Netze erlangen ein zunehmendes Interesse. Das ist zurückzuführen auf die Fähigkeiten von Neuronalen Netzen wie Lernfähigkeit, Fehlertoleranz und Mustererkennung. Auch aus der Sicht der Komplexität wirtschaftlicher Prozesse zeigen sich Neuronale Netze als ein geeignetes Instrument ökonomischer Analyse.

2.2.1 Komplexität ökonomischer Modelle[8]

Wie in Abbildung 7 gezeigt wird, hat die Komplexität eines Modells zwei wichtigen Dimensionen: Grad der Nichtlinearität und Zahl der Variablen.

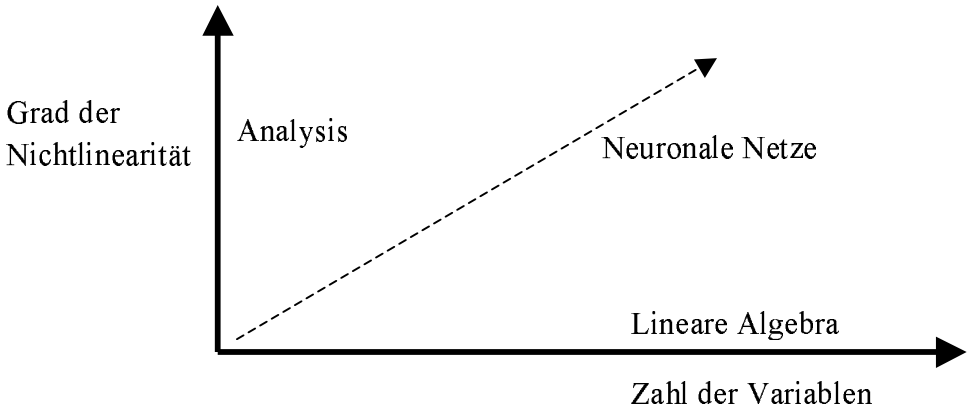


Abbildung 7: Dimensionen der Komplexität[9]

Da einerseits in ökonomischen Prozessen sehr viele Variablen in Wechselwirkung zueinander stehen und andererseits „nicht zu erwarten ist, daß die Ökonomie ihrer Natur

* Als ε_i wird üblicherweise die Standardabweichung der i-ten Zeile von Muster—Matrix benutzt.

[8] Genauer siehe [RZi94], Seiten 12-14.

[9] Quelle siehe Seite 13 in [RZi94].

nach ein lineares Phänomen darstellt“[10], ist die Ökonomie gerade ein Gebiet, wo Hochdimensionalität und Nichtlinearität der Modelle eng miteinander verwoben sind.

Um die neuronalen Abbildungen aus der Sicht der Komplexität klarer vorzustellen, wird ein dreischichtiges Neuronales Netz[11] betrachtet. Jeder Knoten des Neuronalen Netzes realisiert eine primitive nichtlineare Funktion, die „meistens nicht über Summation von Information und eine eindimensionale nichtlineare Funktion“[12] hinausgeht. Aus dieser Sicht stellt sich das Neuronale Netz als eine nichtlineare Funktion F dar, die eine Komposition dieser primitiven nichtlinearen Funktionen ist :

$$F(x) = \text{sigmoid}(W_1\ \text{sigmoid}(W_0 x - \Theta_0) - \Theta_1) = y. \qquad (*)$$

(Dabei wird die Überlagerung der Inputgrößen der einzelnen Schichten durch die Matrizen W_0 und W_1 ausgeführt, die Schwellenwerte sind in den Vektoren Θ_0, Θ_1 schichtweise zusammengefaßt, und die nichtlinearen Sigmoidfunktionen sollen komponentenweise auf die in den Klammern stehenden Vektoren angewandt werden.)

Wie in (*) zu sehen ist, ermöglichen Neuronale Netze einen hochdimensionalen nichtlinearen Modellbau durch eine Hintereinanderschaltung der Methoden, die für die Achsen des o. g. Komplexitätsdiagramms typisch sind: lineare Abbildungen bzw. komponentenweise angewandte eindimensionale, beschränkte, nichtlineare Funktionen.

Diese Eigenschaft Neuronaler Netze, komplexe Modelle in numerisch handhabbarer Weise darzustellen, eignet sie für die Modellierung wirtschaftlicher Prozesse.

2.2.2 Anwendungsgebiete[13]

In der betriebswirtschaftlichen Entscheidungstheorie werden zu lösende Probleme üblicherweise in wohl– und schlechtstrukturierte Probleme eingeteilt. Ein Problem wird dann als *wohlstrukturiert* bezeichnet, wenn

- der Wirkungszusammenhang zwischen Handlungen und ihren Folgen bekannt ist;
- den relevanten Problemmerkmalen eindeutige Werte zugeordnet werden können;
- eine operationale Zielfunktion sowie ein effizientes Lösungsverfahren zur Ermittlung der optimalen Lösung existiert.

Liegen hingegen Probleme vor, auf die keines der o. g. Merkmale zutrifft, dann ist das Problem *wohlstrukturiert* und konventionelle Verfahren wie statistische Methoden, kombinatorische Optimierung oder auch Heuristiken sind besser geeignet, weil diese mit deutlich geringerem Aufwand zu einer Lösung führen. Neuronale Netze ihrerseits sind bei

---

[10] [RZi94], Seite 14.

[11] Wie im Abschnitt 2.1.3 erläutert wurde, ist ein solches Netz hinreichend für die Approximation einer beliebigen nichtlinearen Funktion.

[12] Siehe [Roj96], Seite 25.

[13] Nach [CoM96], Seiten 1-11.

schlechtstrukturierten Problemen der Modellierung dynamischer Marktprozesse besser geeignet. Diese Probleme lassen sich weiter unterteilen:

- Probleme, bei denen der Zusammenhang zwischen den gewünschten Merkmals–ausprägungen und dem zugehörigen Niveau der unabhängigen Variablen unbekannt ist (*wirkungsdefekte* Probleme);
- Probleme, für die keine effiziente Lösungsmethode bekannt ist und die formal nicht mit vertretbarem Aufwand lösbar sind (*lösungsdefekte* Probleme).

Entsprechend lassen sich folgende Ausprägungen Neuronaler Netze unterscheiden:

- *funktionsoptimierende* Neuronaler Netze – bei wirkungsdefekten Problemen;
- *funktionsbildende* Neuronale Netze – bei lösungsdefekten Problemen.

Beim funktionsbildenden Ansatz Neuronaler Netze wird versucht, einen Zusammenhang zwischen Eingabe– und Ausgabewerten aufzustellen, so daß das trainierte Neuronale Netz schließlich als universelle Funktion interpretierbar ist, die Eingabe– auf Ausgabewerte abzubilden vermag. Dieser Vorgang wird auch als *Mustererkennung* bezeichnet.

Demgegenüber ist bei funktionsoptimierenden Neuronalen Netzen der funktionale Zusammenhang zwischen Input und Output eines Netzes bekannt, d. h., es werden lediglich die Ausgabedaten gesucht, die diesem Zusammenhang am ehesten gerecht werden. Hierzu ist es notwendig, das Problem in einer Energiefunktion abzubilden, deren Minimum mit der optimalen Lösung des Problems korrespondiert. Dadurch wird ein *wirkungsdefektes* Problem in ein *lösungsdefektes* transformiert.

Die Anwendungen Neuronaler Netze für Prognoseaufgaben, die offensichtlich als wirkungsdefekte Probleme bezeichnet werden können, basieren auf einem vom Netz gebildeten Modell des zu prognostizierenden Systems, mit dem das Netz in die Lage versetzt werden soll, aus den Vergangenheitsdaten Zukunftswerte zu generieren. Mit dem so gewonnenen Systemmodell können ferner durch die Variationen der Eingabemuster „Was–wäre–Wenn–Simulationen“ durchgeführt werden.

2.2.3 Vorgehensweise beim Einsatz Neuronaler Netze

Ziel der Netzentwicklung ist es, ein Netz zu finden, das mit unbekannten Daten bestmögliche Ergebnisse erreicht. Es kann aber vorkommen, daß keines der getesteten Neuronalen Netze akzeptable Ergebnisse liefert. Dann ist der Ansatz Neuronaler Netze nicht sinnvoll. Dabei kann leider nicht definitiv bestimmt werden, ob es keine Regelmäßigkeit in den Daten gibt oder ein passendes Netz nicht gefunden wurde.

Die Schritte des Einsatzes vom Neuronalen Netz für die Lösung eines Problems werden durch Abbildung 8 graphisch dargestellt, die wichtigsten werden in folgenden Unterkapiteln genauer erläutert.

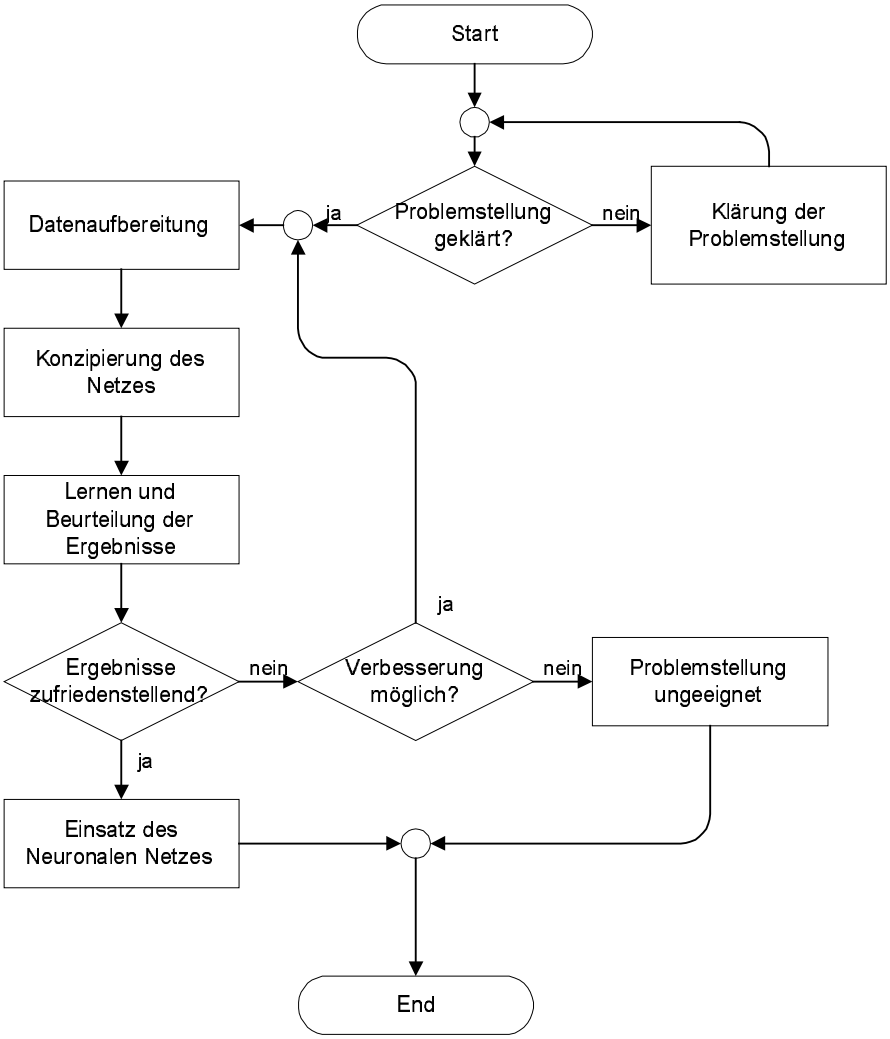


Abbildung 8: Vorgehensweise bei der Entwicklung Neuronaler Netze

2.2.3.1 Klärung der Problemstellung

Während der Klärung der Problemstellung entscheidet man, ob Neuronale Netze für die vorliegende Aufgabe in Frage kommen. Der Einsatz Neuronaler Netze ist nicht sinnvoll, wenn:

- das Problem wohlstrukturiert ist, da dieses dann mit herkömmlichen Ansätzen gelöst werden kann;
- ausschließlich zufälligen oder chaotischen Zusammenhänge vorliegen, d.h. die Daten keine Regelmäßigkeiten aufweisen. Dies ist aber selten *ad hoc* bekannt.

2.2.3.2 Datenaufbereitung

In diesem Schritt sind die vorhandenen Daten so vorzubereiten, daß sie als Input– und Outputdaten für Neuronale Netze dienen können, wobei grundsätzlich folgende Aspekte berücksichtigt werden müssen:

- Erfüllen die Daten qualitative und quantitative Anforderungen?
- Welche Daten sind im Hinblick auf die Problemstellung geeignet und werden dem Neuronalen Netz zugeführt?
- Wie sollen die Daten dem Netz präsentiert werden?
- Welche Daten werden als Trainings–, Test– und Validierungsdaten verwendet?

Hinsichtlich der Quantität gilt üblicherweise die Regel: je mehr desto besser, jedoch nicht immer. Beispielsweise können veraltete Daten bei wirtschaftlichen Zeitreihen störend sein, weil dadurch die Wahrscheinlichkeit der Widersprüche (gleiche Eingaben – unterschiedliche Ausgabe) steigt. Es gibt keine strengen Regeln bezüglich des Umfangs der Daten. Oft wird das Zehnfache der Anzahl der Verbindungsgewichte als erforderlicher Mindestumfang der Datenbasis empfohlen ([CoM96], Seite 8).

Die Qualität (Validität und Konsistenz) der Daten ist die Grundlage der im Neuronalen Netz gebildeten Funktionen. Werden Daten verwendet, die mit hohem Fehleranteil behaftet sind, dann ist davon auszugehen, daß ein Neuronales Netz die Fehler in dieser Form auch reproduziert: *„garbage in – garbage out"*. Demgegenüber beeinflussen vereinzelte falsche Datensätze, bedingt durch die Neuronalen Netzen inhärente Fehlertoleranz, das Lernergebnis kaum.

2.2.3.3 Vorverarbeitung der Zeitreihen

Während der Vorverarbeitung der Daten werden folgende Operationen durchgeführt:

- Datencodierung (optional);
- Datenskalierung oder Datennormierung;
- Trendbereiningung (optional);
- Saisonbereinigung (optional).

Bei der Datencodierung geht es um die Frage, in welcher Form dem Neuronalen Netz Informationen zur Verfügung gestellt und von ihm ausgegeben werden sollen, da nur reelle Zahlen verarbeitet werden können. Manchmal müssen die Daten für eine bessere Informationsdarstellung zerlegt werden. Z. B. ist es sinnvoll, eine in *rad* gemessene Zeitreihe $\{x_1, ..., x_n\}$, die Messungen eines Winkels enthält, durch zwei neue: $(\sin(x_1), ..., \sin(x_n))$ und $(\cos(x_1), ..., \cos(x_n))$ zu ersetzen, um die Kontinuierlichkeit der Daten besser darzustellen.

Für Erläuterung einiger Vorverarbeitungsmethoden werden folgende Definitionen benutzt:

- $x=(x_1, x_2, ..., x_t, ...,x_n)$ ursprüngliche Zeitreihe;

- $z=(z_1, z_2, \ldots, z_t, \ldots, z_n)$ transformierte Zeitreihe;
- aver(x) Mittelwert von x;
- std(x) Standardabweichung von x;
- min(x), max(x) globale Extremwerte von x über Trainingsraum.

Die Formel für die Skalierung ist dann

$$z_t = (x_t - \text{aver}(x)) / \text{std}(x)$$

und die Formel für die Normierung:

$$z_t = (x_t - \min(x)) / (\max(x) - \min(x)) .$$

Diese Normierung hat allerdings den Nachteil, daß die Transformation der Datenmenge durch die Ausreißer bestimmt wird. So können wegen der wenigen Extremwerte die anderen Werte der Datenmenge fast zu einer Konstante zusammengedrückt werden. Die Skalierung im Gegensatz sorgt dafür, daß die Mehrheit der Werte in einem mittleren Bereich gehalten wird, während einige Extremwerte sehr wohl nach unten oder nach oben ausreißen können.

Falls Zeitreihen als Inputdaten für das Neuronale Netz benutzt werden, ist es manchmal notwendig, eine Trend– und Saisonbereinigung durchzuführen. Sonst kann es, wenn eine trendbehaftete Zeitreihe zum Bau von Prognosemodellen verwendet wird, gerade bei Neuronalen Netzen wegen des *Sättigungseffektes* (siehe [RZi94], Seite 22) zu völlig verzerrten Ergebnissen kommen. Für eine Trendbereinigung der Rohdaten sorgen die Differenzbildungen:

$z_t = (x_t - x_{t-n})/n$ erste Ableitung

$z_t = (x_t - 2x_{t-n} + x_{t-2n})/n^2$ zweite Ableitung

$z_t = (x_t - 3x_{t-n} + 3x_{t-2n} - x_{t-3n})/n^3$ dritte Ableitung

2.2.3.4 Konzipierung des Netzes

Im Rahmen der Konzipierung des Netzes wird ein geeigneter Netztyp ausgewählt und konfiguriert. Obwohl dies von grundlegender Bedeutung für die Leistungsfähigkeiten des Neuronalen Netzes ist, liegen hierzu bislang noch keine konkreten Erkenntnisse vor. Zur Auswahl des Netztyps lassen sich zur Zeit nur Aussagen hinsichtlich einer grundsätzlichen Eignung bestimmter Netztypen für einzelne Anwendungsfelder tätigen (siehe [CoM96], Seite 10). Für den Fall des überwachten Lernens gilt das *backpropagation*–Verfahren als weitgehend universell einsetzbar. Im Rahmen der Netzkonfiguration sind die noch offenen Parameter des Neuronenmodells, der Netzarchitektur und des Lernverfahrens festzulegen

2.2.3.5 Testen und Beurteilung

Im Anschluß daran werden beim Lernvorgang dem Netz Trainingsdaten zugeführt und die Verbindungsgewichte anhand der Lernregel angepaßt. Nach dem Ende des Lernens wird die Güte der Ergebnisse beurteilt. Hierzu dienen unterschiedliche statistische Kennzahlen,

sowohl während des Lernens (mittlere quadratische Fehler an Testdaten) als auch unterschiedliche *ex–post*–Beurteilungen an Validierungsdaten nach dem Ende der Lernphase. Ebenso kann eine graphische Gegenüberstellung von gewünschtem und tatsächlichem Netzoutput bei der qualitativen Bewertungen manchmal sogar hilfreicher als statistische Kennzahlen sein.

Falls die Ergebnisse nicht zufriedenstellend sind, werden weitere Tests mehrmals durchgeführt mit verschiedenen Einstellungen bezüglich:

- Topologie des Netzes;
- Lernparameter (Lernalgorithmus, Lernrate, Abbruchkriterium);
- Neuronenparameter (*net*–Funktion, Aktivierungsfunktion);
- Modell (andere o. zusätzliche Inputzeitreihen, Verschiebung der Inputzeitreihen);
- Vorverarbeitungsmethoden der Zeitreihen;

3 Grundlagen der Linearen Regression

Lineare Regression ist eine Methode der Untersuchung und der funktionalen Beschreibung der linearen Beziehungen zwischen zwei oder mehr Variablen, die zusammen ein Modell bilden. Üblicherweise wird jede dieser Variablen durch einen Vektor der empirischen Werte dargestellt. Dabei geht eine Variable als die abhängige Größe oder der *Regressand* in die Untersuchung ein; die übrigen gelten als unabhängige oder *Regressoren*. Gibt es nur einen Regressor, so spricht man über die *einfache* Lineare Regression, sonst über die *multiple* Lineare Regression. Es gibt unterschiedliche Methoden der Zusammensetzung und der Beurteilung der Güte eines linearen Modells. Liefert die lineare Regressionsanalyse des Modells gute Ergebnisse, mit anderen Worten, existieren starke lineare Zusammenhänge zwischen Komponenten des Modells, so kann man aus bekannten Regressoren den Regressand prognostizieren.

3.1 Einführung

Für die weiteren Erläuterungen werden folgende Größen definiert:

n	die Anzahl der Beobachtungen;
m	die Anzahl der Exogenen;
$df_1 = m$	Freiheitsgrade der 1. Art;
$df_2 = n - m - 1$	Freiheitsgrade der 2. Art;
$df_3 = n - 1$	Freiheitsgrade der 3. Art;
$\mathrm{y}=(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n)$	ein Regressand oder eine Endogene (Vektor der Dimension n);
$\hat{\mathrm{y}}=(\hat{\mathrm{y}}_1, \hat{\mathrm{y}}_2, \ldots, \hat{\mathrm{y}}_n)$	der geschätzte Regressand (Vektor der Dimension n);
$\bar{y}=\frac{1}{n}\sum_{i=1}^{n} y_i$	der Mittelwert von y;
$x^{(0)}=(1,\ldots,1)$	der „*dummy*“– oder *Scheinregressor* (Vektor der Dimension n);
$x^{(i)}, i = 1,\ldots,n$	der i–te Regressor oder die i–te Exogene (Vektor der Dimension n);
b_0	das konstante Glied;
$b=(b_0, \ldots, b_m)$	der Vektor der linearen oder Regressions–Koeffizienten;
$e=\mathrm{y}-\hat{\mathrm{y}}$	der Vektor der Residualgrößen der Dimension n;
$X = \begin{pmatrix} x^{(0)} \\ \\ x^{(m)} \end{pmatrix}$	die Matrix der Regressoren der Dimension $(m+1, n)$.

Multiple lineare Regressionsmodelle lassen sich in der Form

$$y = bX + e$$

darstellen.

Der Ansatz zur Berechnung modellspezifischer Schätzwerte lautet dann:

$$\hat{y} = bX .$$

Der Vektor der linearen Koeffizienten b ist nach der ***klassischen Methode der kleinsten Quadrate*** (*ordinary least squares, OLS*) zu ermitteln[14]. Nach der Methode der kleinsten Quadrate ist

$$\|e\|^2 = \sum e_i^2 = \sum (y_i - \hat{y}_i)^2$$

zu minimieren. Die Lösungsmethode – die partielle Differentiation und die Entwicklung eines Normalgleichungssystems – führt zu

$X^T y^T = (X^T X) b^T = 0$, wobei unter 0 ein $(n,1)$–Nullvektor verstanden wird.

Aus dem Normalgleichungssystem ergibt sich für den Vektor b der Regressionsparameter:

$$b^T = (X^T X)^{-1} X^T y^T$$

und damit, da b bekannt ist, die Möglichkeit zur Berechnung von

$\hat{y}^T = Xb^T$ und $e = y - \hat{y}$.

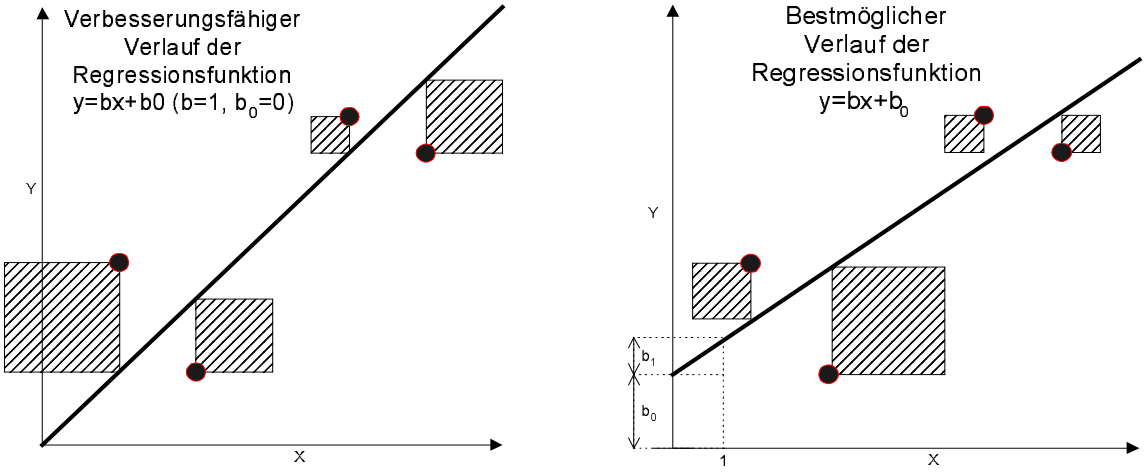


Abbildung 9: Methode der kleinsten Quadrate (oridinary least squares)

Die Methode der kleinsten Quadraten läßt sich für den Fall der einfachen linearen Regression gut illustrieren (siehe Abbildung 9), da jedes Paar (x_i, y_i) als ein Punkt der Fläche aufgefaßt werden kann.

[14] Siehe [Web90], S.80-82.

3.2 Beurteilung eines linearen Modells

Für die Beurteilung der Güte sowohl des gesamten linearen Modells als auch der einzelnen Regressoren werden mehrere Kennzahlen verwendet, unter anderem:

Für den Regressoren $\mathbf{x^{(i)}}$, mit i=0, 2, ... m :

- Standardfehler des linearen Koeffizientes b_i r_i
- Korrelationskoeffizient R_i
- Bestimmtheitsmaß (oder Determinationskoeffizient) R^2_i
- Semipartieller quadratischer Korrelationskoeffizient $R^2_{i.12...i\text{-}1}$
- t–Test der Signifikanz des linearen Koeffizientes b_i t_i

Für das gesamte Modell:

- Multiples Bestimmtheitsmaß R^2
- Multiples verbessertes (*adjusted*) Bestimmtheitsmaß $\overline{R}^2$
- Multipler Korrelationskoeffizient R
- F–Test. F

In folgenden Abschnitten werden die o. g. Kennzahlen kurz beschrieben. Zu weiteren Erläuterungen zu diesem Thema siehe Bücher über mathematische Statistik wie [Edw84], [Web90] u. a.

3.2.1 Beurteilung des *i*–ten Regressoren $x^{(i)}$

- **Standardfehler** des Regressoren $x^{(i)}$ ist:

$r_i = \sqrt{MS_{res} c_{ii}}$ mit der mittleren quadratischen Abweichung $MS_{res} = \frac{SS_{unerklärt}}{df_2}$

und dem diagonalen Element c_{ii} der Matrix $(X^T X)^{-1}$, das b_i entspricht.

- **Korrelationskoeffizient** zwischen $x^{(i)}$ und y ist die Kovarianz $c_{x(i)y}$ von $x^{(i)}$ und y, geteilt durch das Produkt der Standardabweichungen s von Variablen $x^{(i)}$ und y.

$$R_i = \frac{c_{x^{(i)}y}}{s_{x^{(i)}} s_y} = \frac{\sum_j (x_j^{(i)} - \overline{x})(y_j - \overline{y})}{\sqrt{\sum_j (x_j^{(i)} - \overline{x})^2} \sqrt{\sum_j (y_j - \overline{y})^2}}, \; j = 1, \ldots, \; n$$

Entsprechend dem Korrelationskoeffizient kann die „Indikatorqualität“ nach Bravias–Pearson[15] klassifiziert werden:

---

[15] Siehe [Web90], Seite 206.

$\lvert R_i \rvert \geq 0{,}8$	sehr guter Indikator;
$0{,}8 > \lvert R_i \rvert \geq 0{,}6$	guter Indikator;
$0{,}6 > \lvert R_i \rvert \geq 0{,}4$	durchschnittlicher Indikator;
$0{,}4 > \lvert R_i \rvert \geq 0{,}0$	schlechter/kein Indikator.

- Das **Bestimmtheitsmaß** R_i^2 von $x^{(i)}$ ist einfach das Quadrat des Korrelations–koeffizienten R_i
- Der **semipartielle quadratische Korrelationskoeffizient** $R^2_{i.12\ldots i\text{-}1}$ zeigt, um wieviel das Bestimmtheitsmaß des Modells steigt nach dem der Regressor x_i ins Modell eingefügt wird. Für $R^2_{i.12\ldots i\text{-}1}$ gelten die Formeln:

  $$R^2_{i.12\ldots i\text{-}1} = t_i^2 \frac{1-R^2}{df_2} \quad \text{und}$$

  $$R^2 = R_1^2 + R_{2.1}^2 + \ldots + R^2_{n.12\ldots n-1}$$

  Die letzte Formel ist nach Allen L. Edwards „perfectly general“[16] und läßt den Beitrag des neu eingefügten Regressoren als die Differenz zwischen dem alten und dem neuen Bestimmtheitsmaß des Modells berechnen.

- ***t*–Test der Signifikanz für den linearen Koeffizient b_i**

  Alle Signifikanztests (sowohl für einzelne Regressoren als auch für ein Modell) haben das gleiche Prinzip: Die auf den Beobachtungswerten basierende Statistik wird mit einer Standardstatistik verglichen, die eine bekannte Wahrscheinlichkeitsverteilung hat. Aufgrund dieses Vergleichs kann mit einer bestimmten Wahrscheinlichkeit die Aussage über die Signifikanz des Regressoren bzw. des Modells gemacht werden. Für den Fall der einfachen linearen Regression kann die Signifikanzanalyse des Regressionskoeffizienten b_1 visuell illustriert werden, wie es in Abbildung 10 gezeigt wird.

  Der *t*–Test dient zur Untersuchung der Signifikanz einzelner Regressoren sowie zur Ermittlung von Konfidenzintervallen für Individual– und Mittelwerte der abhängigen Variablen. Die Testgröße läßt sich nach der Formel

  $$t(b_i, df_2) = \frac{b_i}{r_i}, i = 1, \ldots, m$$

[16] Siehe [Edw84], Seite 129.

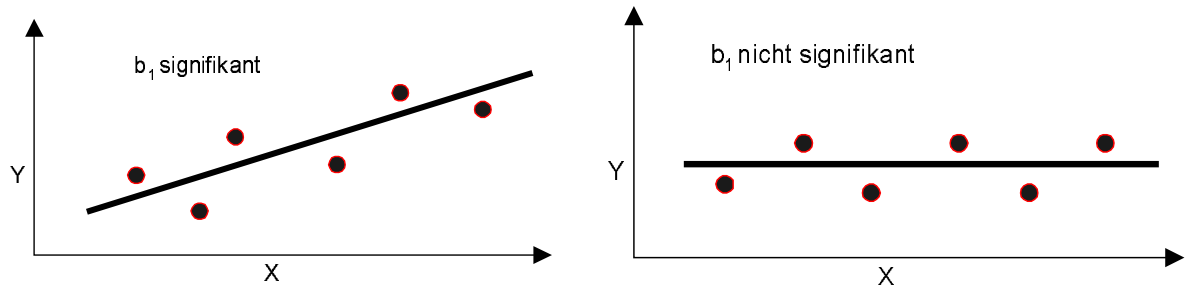


Abbildung 10: Evaluation der Regressionsfunktion $\mathbf{y=b_0+b_1x}$

berechnen. Danach wird die berechnete Größe mit dem tabellarischen Wert von der statistischen *t*–Verteilung[17] $t_{tabellarisch}(\alpha/2,\ df_2)$ verglichen. Falls

$$|t(b_i)|>t(\alpha/2,df_2)$$

zutrifft, gilt der berechnete Regressionskoeffizient b_i mit der Wahrscheinlichkeit $1\text{-}\alpha$ als wesentlich verschieden von Null; er sollte mithin bei der endgültigen Festlegung des Regressionsmodells mitberücksichtigt werden[18].

3.2.2 Beurteilung des gesamten Modells

- Das **Bestimmtheitsmaß** R^2 wird im Laufe der *Abweichungsanalyse* eines linearen Modells berechnet. Die Abweichungsanalyse läßt sich nach der folgenden Formel ausgestalten:

$$(y_i - \bar{y}) = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y}) .$$

Für die aus den Einzelabweichungen zu berechnenden Quadratsummen gilt[19]:

$$\sum_i (y_i - \bar{y})^2 = \sum_i (y_i - \hat{y}_i)^2 + \sum_i (\hat{y}_i - \bar{y})^2, i = 1,\ldots, n$$

oder kompakt

$$SS_{total} = SS_{unerklärt} + SS_{erklärt}.$$

Der Bestimmtheitsmaß R^2 ist dann die Relation der erklärten Abweichungen $SS_{erklärt}$ zu den Gesamtabweichungen SS_{total}:

$$R^2 = \frac{SS_{erklärt}}{SS_{total}} = 1 - \frac{SS_{unerklärt}}{SS_{total}}.$$

Der Korrelationskoeffizient R und das Bestimmtheitsmaß R^2 sind durch eine einfache Relation verbunden:

---

[17] Auch *student*-Verteilung.

[18] Abraham und Ledolter (1983), S. 26; nach der Eingabe in [Web90].

[19] Siehe z.B. [Web90], Seite 83.

$$R = \sqrt{R^2}\ .$$

Oft verwendet man den sogenannten *verbesserten* (*„**adjusted**“*) Korrelationskoeffizient $\overline{R}^2$, wobei die Relativierung von R^2 bezüglich der Freiheitsgrade berücksichtigt wird:

$$\overline{R}^2 = R^2 - (1 - R^2)\frac{n-1}{n-m-1} = R^2 - (1 - R^2)\frac{df_3}{df_2}\ .$$

Diese Änderung führt dazu, daß $\overline{R}^2$ im Gegensatz zu R^2 auch abnehmen kann, wenn eine zusätzliche Variable mitberücksichtigt wird.

- Der **F–Test** wird oft für die Signifikanzanalyse der linearen Modelle verwendet. Beim F–Test vergleicht man die Verteilung der echten *y*–Werte rund um die geschätzte Linie mit der bekannten statistischen F–Verteilung und kann dann mit einem bestimmten Signifikanzniveau α (üblicherweise α = 0.1, 0.05 oder 0.01) die Signifikanz des Modells mit dem ausgewählten Signifikanzniveau tabellarisch bestimmen.

  Die F–Statistik ist nach der Formel

  $$F(df_1, df_2) = \frac{R^2 / df_1}{(1-R^2)/df_2}$$

  zu berechnen, die sich auch in der Form

  $$F(df_1, df_2) = \frac{SS_{erklärt} / df_1}{SS_{unerklärt} / df_2}$$

  schreiben läßt. Danach wird der tabellarische Wert $F_\alpha(df1, df2)$ für ein bestimmtes α in der Tabelle der *F*–Verteilung gefunden. Falls der berechnete *F*–Wert größer als F_α ist, kann das verwendete Modell mit der Wahrscheinlichkeit 1-α als signifikant betrachtet werden.

  Die Aussagefähigkeit des F–Tests wird allerdings in der Literatur zurückhaltend beurteilt[20].

3.3 Erstellung eines linearen Modells

3.3.1 Berücksichtigung der zeitlich verlagerten Auswirkungen

Mit der Linearen Regression können keine in der Zeit verschobene Zusammenhänge zwischen Zeitreihen entdeckt werden. Bei wirtschaftlichen Prozessen spielen aber solche Zusammenhänge eine große Rolle, weil gerade in diesem Bereich Auswirkungen einer Einflußgröße auf die Endogene oft erst später zu sehen sind. Um dieses mitzuberücksichtigen, werden im Modell in die Zukunft verschobene Zeitreihen, sog.

[20] Siehe [Web90], Seite 88[Web90].

Indikatoren[21], als Regressoren benutzt. Eine von mehreren Methoden zur Auswahl der optimalen Verschiebung basiert auf der Korrelationsanalyse:

Die $k \geq 0$ und $K \geq k$ seien untere bzw. obere Grenzen von betrachteten Verschiebungen der Indikatoren x^i. $R^{(i)}$ sei der Korrelationskoeffizient zwischen diesem Regressoren und dem Regressand. Die optimale Verschiebung i kann dann aus der folgenden Formel bestimmt werden:

$$i_{opt} = \max_{R^{(i)}} (i), k \leq i \leq K \ .$$

Die Qualität jedes Regressoren $x^{(i)}$ kann zusätzlich nach *Bravias–Pearson*[22] klassifiziert werden.

Zwei alternative Vorgehensweise bei der Betrachtung von zeitlich verschobenen Zusammenhängen lassen sich unterscheiden:

(1) Zuerst werden unter den ursprünglichen Zeitreihen die Regressoren für das Modell ausgewählt und dann ihre optimalen Verschiebungen („lag–Struktur“ des Modells) ermittelt;

(2) Zuerst werden aus der Menge der Basiszeitreihen, die als Einflußgrößen in Frage kommen, eine größere Menge der Indikatoren erstellt, wobei aus einer Zeitreihe mehrere Indikatoren erzeugt werden[23]. Bei der Modellerstellung werden diese Indikatoren als vollständige Zeitreihen betrachtet: Es wird nur ihre Güte beurteilt ohne auf ihre Herkunft zu achten. In Folge dessen kann es allerdings vorkommen, daß dieselbe, jeweils unterschiedlich verschobene Zeitreihe im Modell mehrmals vorkommt.

In dieser Arbeit wird die zweite Vorgehensweise bevorzugt, weil sie systematischer zu sein scheint. Deswegen fängt jeder Modellierungsprozeß mit der Erstellung der Indikatoren an, und die Auswahl der optimalen „lag–Struktur“ des Modells ist implizit im globalen Wettbewerb zwischen den Indikatoren (verschobenen Zeitreihen) um einen Platz als Regressor enthalten.

3.3.2 forward–, backward– und *stepwise–solution* – Algorithmen

Charakteristisch für Lineare Regression ist eine ständige Verbesserung der geschätzten Werte des Regressandes durch die Einfügung weiterer Regressoren ins Modell, was durch die Erhöhung der Freiheitsgrade df_1 geschieht.

Dennoch ist die Berücksichtigung einer großen Zahl von Regressoren nicht sinnvoll, weil die nötigen Berechnungen an Matrizen schnell aufwendig werden. Ein neuer Regressor

---

[21] Siehe Def. 1.2

[22] Siehe Seite 21.

[23] Üblicherweise werden aus einer Zeitreihe x Indikatoren $x^{(k)}$, $x^{(k+1)}$, ..., $x^{(K)}$ erzeugt (k bzw. K sind die angegebenen minimale bzw. maximale betrachtete Verschiebung).

kann darüber hinaus keinen wesentlichen Beitrag liefern: falls er z. B. eine niedrige Korrelation mit dem Regressand hat oder wenn er mit übrigen Regressoren des Modells stark korreliert. Außerdem wird die Anzahl der Regressoren durch die Anzahl der berücksichtigten Werte begrenzt. Deswegen benutzt man bei mehreren potentiellen Regressoren unterschiedliche Algorithmen zur Auswahl einer begrenzten Menge der besten Exogenen. In [Edw84] werden drei solche Algorithmen beschrieben: *forward solution*, *backward solution* und *stepwise solution*. Im Rahmen der Arbeit wird der *forward–solution*–Algorithmus verwendet, der in Abbildung 11 schematisch abgebildet ist. Bei diesem Verfahren wird schrittweise aus einer Menge unbenutzter Indikatoren die signifikanteste Exogene ausgewählt und ins Modell eingefügt. Dieses Verfahren hat allerdings den Nachteil, daß eine Einflußgröße, die frühzeitig ausgewählt wurde, nach dem Einfügen von weiteren Einflußgrößen nicht mehr so viel wie vorher zu dem Bestimmt–heitsmaß R^2 des Modells — und damit in das Ergebnis des F–Testes — beitragen kann.

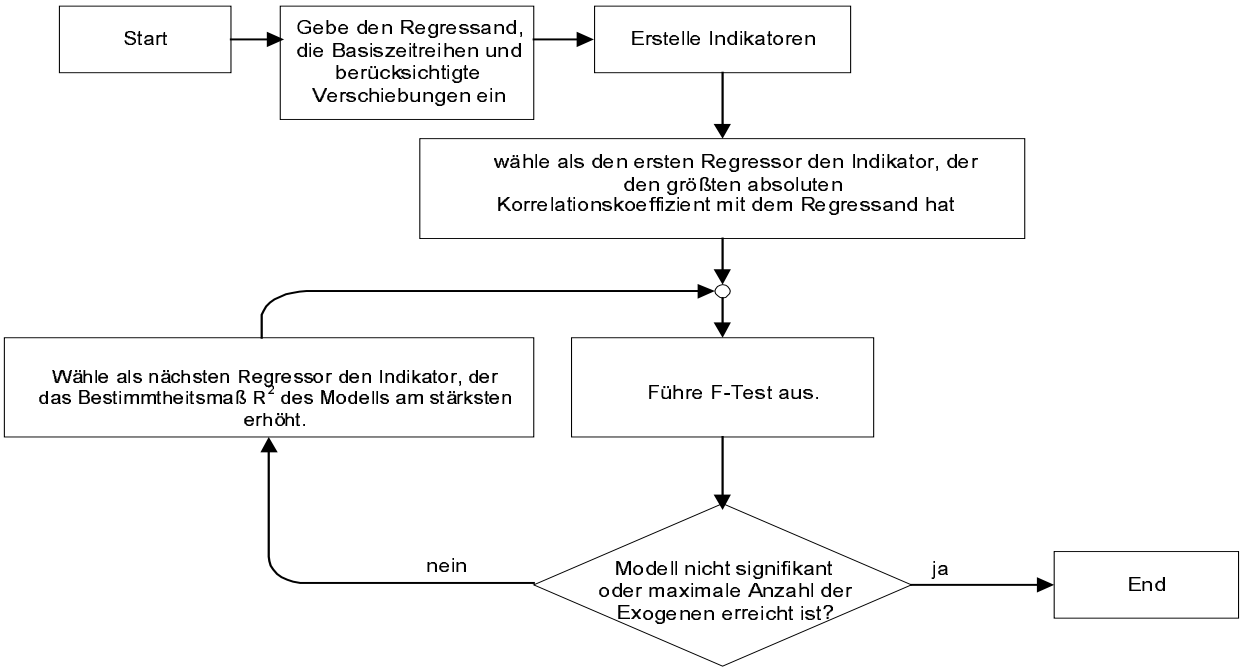


Abbildung 11: forward–solution–Algorithmus der Erstellung eines linearen Modells

Der *backward–solution*–Algorithmus ist die Umkehrung vom *forward solution*: Am Anfang werden alle potentiellen Regressoren ins Modell aufgenommen, und dann werden die unrelevantesten (die das Bestimmtheitsmaß des Modells am wenigsten erhöhen) sukzessiv ausgeschlossen. Dieses Verfahren hat den Nachteil, daß bei einer großen Anzahl der Regressoren im Modell der Rechenaufwand sehr hoch ist. Deswegen wäre *forward solution* bei großen Mengen der in Frage kommenden Regressoren besser geeignet.

Ein komplizierteres aber zuverlässigeres Verfahren ist *stepwise solution*. Das ist eine Kombination der o. g. Algorithmen: Nachdem mit dem *forward solution* die maximale

Anzahl der Regressoren ausgewählt wurde, wird der F–Test für die weitere Optimierung des Modells benutzt. In jedem weiteren Schritt des Verfahrens wird der unrelevanteste, mit dem *backward solution* ermittelte Regressor durch den besten unbenutzten, mit dem *forward solution* ermittelten Indikator ersetzt. Dieser Vorgang wird wiederholt, bis das verwendete Abbruchkriterium erfüllt ist: Z. B. die Menge der noch nicht überprüften Indikatoren ist erschöpft, oder das Modell wird nicht (wesentlich) besser. S*tepwise solution* ist viel aufwendiger als *forward solution*, deswegen wurde das letztere Verfahren anfänglich für die Implementation gewählt. Es wäre aber sinnvoll, zukünftig auch diesen Algorithmus zu implementieren und einem Benutzer für die Erstellung der Regressionsmodelle zur Verfügung zu stellen.

4 *ex–post*–Analyse [24]

Unter der *ex–post*–Analyse wird eine Beurteilung der Prognosequalität durch den Vergleich der prognostizierten mit den tatsächlichen Werten verstanden. Dafür werden unterschiedliche Fehlermaße benutzt. Die am häufigsten benutzten Kennzahlen werden unten aufgezählt.

4.1 Fehlermaße

Alle Fehlermaße basieren auf der Definition des Prognosefehlers e_t zum Zeitpunkt t aus dem Prognoseraum (T+1,T+2,...,T+l):

$$e_t = y_t - \hat{y}_t \quad , \quad t = T+1,...,T+l \, .$$

- Mittlerer Prognosefehler *ME*

$$ME = \frac{1}{l} \sum_{t=T+1}^{T+l} e_t$$

- Mittlerer absoluter Prognosefehler *MAE*

$$MAE = \frac{1}{l} \sum_{t=T+1}^{T+l} |e_t|$$

- Mittlerer relativer Prognosefehler *MRE*

$$MRE = \frac{1}{l} \sum_{t=T+1}^{T+l} \frac{|e_t|}{y_t}$$

- Mittlerer absoluter prozentualer Prognosefehler *MAPE*

$$MAPE = \frac{100}{l} \sum_{t=T+1}^{T+l} \left| \frac{e_t}{y_t} \right|$$

- Summe der quadratischer Prognosefehler *SSE*

$$SSE = \frac{1}{l} \sum_{t=T+1}^{T+l} e_t^2$$

- Mittlerer quadratischer Prognosefehler *MSE*

$$MSE = \frac{SSE}{l} = \frac{1}{l} \sum_{t=T+1}^{T+l} e_t^2$$

- Wurzel des mittleren quadratischen Prognosefehler *RMSE*

---

[24] Als Quelle wurde die Zusammenfassung aus [Thi98] (Seiten 63-66) benutzt.

$$RMSE = \sqrt{MSE}$$

4.2 *Theil*-scher Ungleichheitskoeffizient

Wenn die Qualität der unterschiedlichen Prognosen für eine und dieselbe prognostizierte Zeitreihe verglichen wird, ist der *RMSE* aussagefähiger, weil diese Kennzahl den durchschnittlichen „Abstand" zwischen tatsächlichen und prognostizierten Werten in Einheiten der Zeitreihe darstellt. Um die Höhe des Prognosefehlers anhand *RMSE* zu beurteilen, muß man allerdings zumindest den Mittelwert und die Standardabweichung der prognostizierten Zeitreihe kennen. *MAPE* ist wegen seiner Relativität für den Vergleich der Prognostizierbarkeit verschiedener Zeitreihen geeigneter.

Für die Entscheidung, ob die Prognose „gut" oder „schlecht" ist, soll eine untere Schranke der Prognosequalität gewählt werden. In der Regel wird dafür die sog. „naive" Prognose benutzt, bei der „der letzte Beobachtungswert als Prognosewert dient". Dieser Wert sei o.E.d.A. gleich y_T. Der *Theil'scher Ungleichheitskoeffizient* U stellt eine Normierung der „aktuellen" Prognose bezüglich der „naiven" dar und kann dann aus der folgenden Formel bestimmt werden:

$$U = \sqrt{\frac{\sum_{t=T+1}^{T+l} (y_t - \hat{y}_t)^2}{\sum_{t=T+1}^{T+l} (y_t - \hat{y}_T)^2}} = \frac{RMSE_{aktuell}}{RMSE_{naiv}}.$$

Leicht zu sehen, daß wenn $U < 1$ ist, ist die aktuelle Prognose besser als die naive und die Prognosemethode eingesetzt werden kann. Dabei ist zu beachten, daß der Hauptnachteil des Ungleichheitskoeffizienten darin liegt, daß er die Prognosefehler zwischen U=0 und U=1 in der Regel nicht linear abbildet (siehe [Thi98], Seite 66).

5 Entwurf eines Modellierungs– und Prognosesystems

Im Rahmen der Arbeit wurde das Konzept eines Systems zur automatischen Prognoseerstellung mit einer erweiterten Benutzerunterstützung entworfen; seine Hauptmodule wurden implementiert und empirisch verifiziert. Die Erstellung eines Modells für die untersuchte Zeitreihe und ihre Prognose werden dabei als Teile eines gemeinsamen Prozesses betrachtet. Dieser Abschnitt enthält Beschreibungen der sowohl beim Hauptprozeß als auch in den einzelnen Modulen verwendeten Algorithmen.

5.1 Problemmerkmale

Beim Entwurf des Algorithmus der Modellerstellung wurden folgende relevante Merkmale berücksichtigt:

- Der unterschiedliche Umfang des Expertenwissens bei verschiedenen Benutzern im Bereich Neuronaler Netze und ökonomischer Modellierung.
- Eine hohe Komplexität des Problems, die sich wegen der vielen festzulegenden Parameter des Modells ergibt, insbesondere aufgrund der:
    - vielen vorhandenen Zeitreihen, die als Exogene in Frage kommen, aber nicht alle ins Modell eingefügt werden können;
    - Berücksichtigung der zeitlich verlagerten Zusammenhänge;
    - vielen zu bestimmenden Parameter der neuronalen Modelle, die mit keinem analytischen Verfahren ermittelt werden können.

5.2 Benutzermodell

Es bietet sich an, verschiedene Ebenen der Automatisierung des Prognoseprozesses dem Benutzer zur Verfügung zu stellen. Ein Benutzer kann dann eine Unterstützungsebene bevorzugen, die seinem Vorbereitungsniveau am besten entspricht. Am flexibelsten kann es mit einem „Fragebogen“ realisiert werden, womit ein Benutzer angibt, welche Teile des Prozesses er dem Computer übergibt und welche beeinflussen will. Alternativ oder zusätzlich kann das Niveau des Expertenwissens eines Benutzers abgefragt werden. Anhand der Antworten bildet und speichert das System ein *Benutzermodell* und paßt sich an dieses Modell an. Beispiele der verschiedenen Automatisierungsniveaus werden in den Abbildung 12 und Abbildung 13 angezeigt. Wie zu sehen ist, betreffen diese Veränderungen nur die logische Verbindungsstruktur zwischen wiederverwendbaren Hauptblöcken des Prozesses und die Zulassung eines Benutzereinflusses an einigen Stellen des Prognoseprozesses.

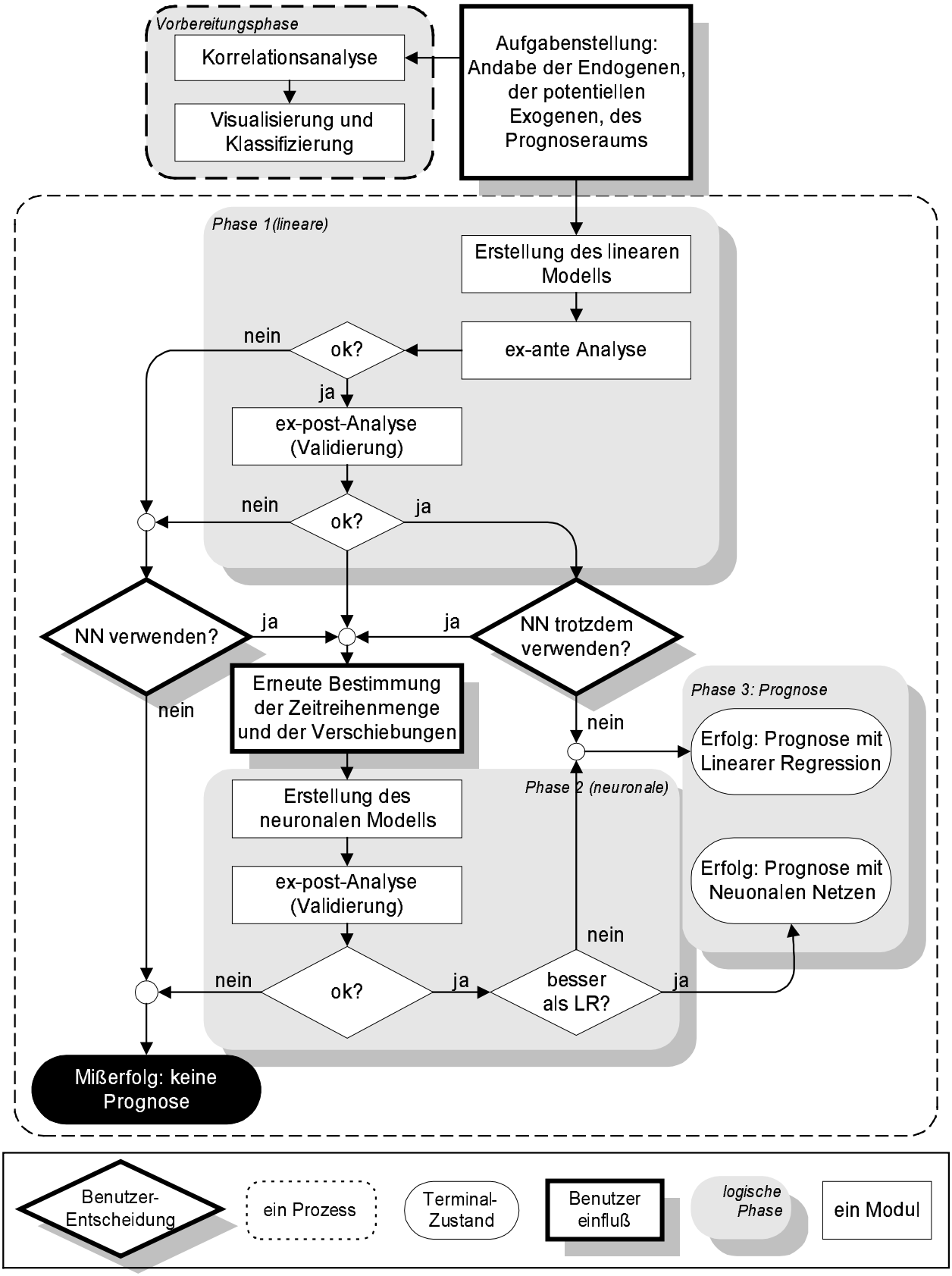


Abbildung 12: Modell– und Prognoseerstellung beim erweiterten Benutzereinfluß

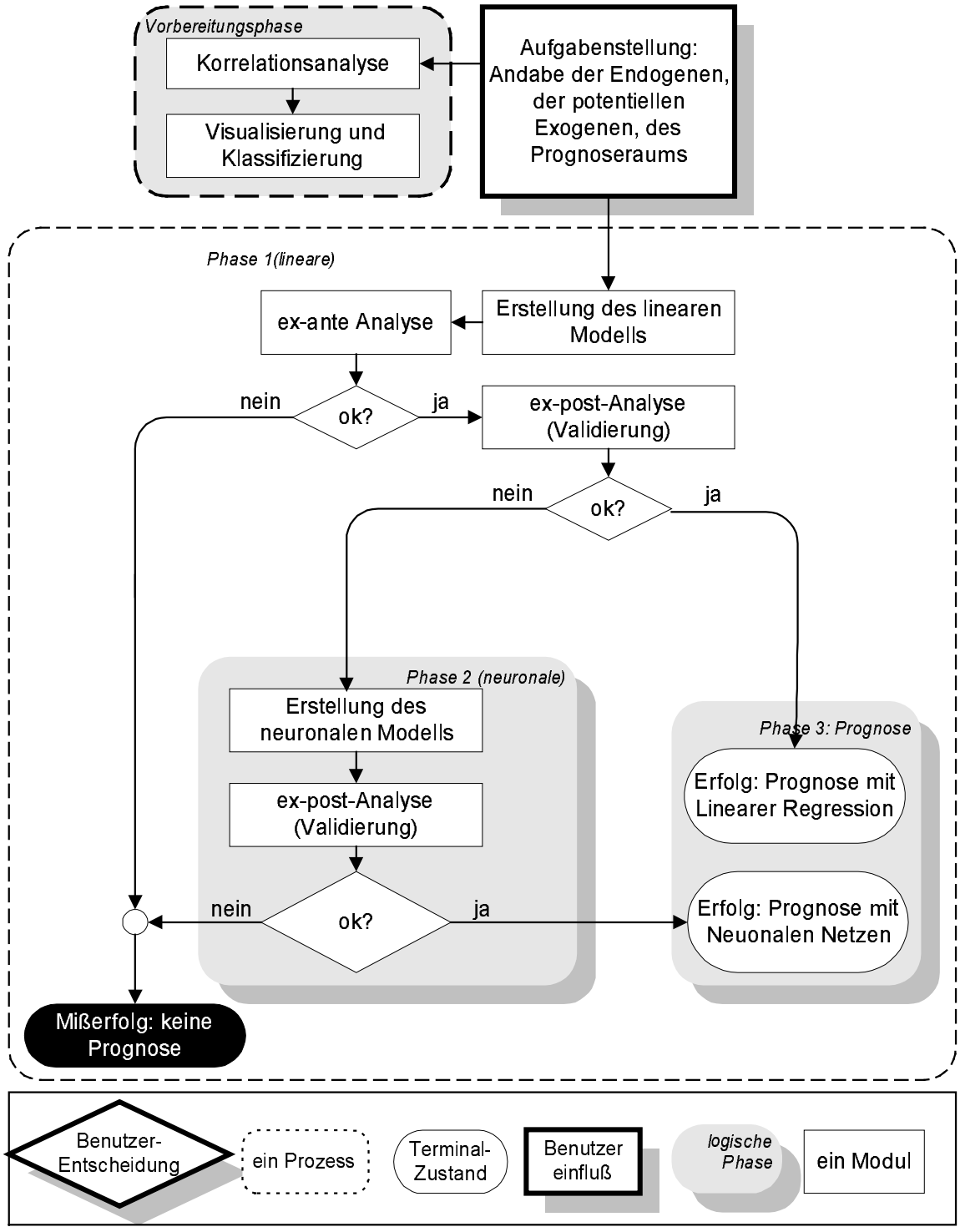


Abbildung 13 : Modell– und Prognoseerstellung ohne den Benutzereinfluß

5.3 Phasen des Prognoseprozesses

Für das Starten des Prognoseprozesses braucht das System eine Aufgabenstellung. Dafür muß ein Benutzer mindestens den Namen der prognostizierten Zeitreihe eingeben.

Optional können die zusätzlichen Modellierungs– und Prognoseparameter angegeben werden, darunter

- erwünschte Anzahl der Einflußgrößen;
- Start der Prognose;
- erwünschte Prognoselänge;
- Menge der zu prüfenden potentiellen Exogenen;
- Grenzen der berücksichtigten zeitlichen Verschiebungen von Zusammenhängen;
- Einstellungen Neuronaler Netze.

Einige angegebene Informationen können inkonsistent oder unzulässig sein. In diesem Fall werden sie vom System korrigiert, und der Benutzer wird benachrichtigt. Danach fängt der Modellierungs– und Prognoseprozeß an. Entsprechend der Abbildung 13 kann er in drei Hauptphasen eingeteilt werden:

(1) Erstellung, Beurteilung und Validierung eines linearen Modells;

(2) ggf. Erstellung und Validierung eines neuronalen Modells;

(3) ggf. (bei Erfolg) Erstellung der Prognose.

Der Prozeß enthält die in folgenden Unterkapiteln beschriebenen prozeduralen Hauptblöcke:

(1) Validierung des Modells und Erstellung der Prognose;

(2) Erstellung neuronales Modells;

(3) Erstellung lineares Modells;

5.3.1 Erstellung eines linearen Modells

In diesem Schritt werden die optimale Anzahl der Indikatoren im linearen Modell und die maximale Prognoselänge bestimmt:

- Die maximale Prognoselänge ist um 1 größer als die Exogenenzahl (Quelle: [Edw84], S. 126).
- Bezüglich der optimalen Anzahl der Indikatoren wird in [Edw84] (S. 99) empfohlen: „In general, the number of observations n in a multiply correlation or regression problem should be considerably larger than the number of X Variables k. No hard rules are available, but we would suggest that n/k should be equal to or greater than 10."

  Die Länge des Validierungszeitraums ist (wie es im vorherigen Abschnitt beschrieben wurde) gleich der maximalen Prognoselänge, die sich aus der o. g. Regel ergibt. Wenn k die Anzahl der Exogenen ist, dann ist die maximale Prognoselänge gleich k+1. Nach dem Ausschließen der Validierungsmenge bleiben von n Beobachtungen n-k-1 übrig. Damit ergibt sich für die Bestimmung der Exogenenzahl k_{max} im Modell:

$$(n-k-1)/k \geq 10 \quad \Rightarrow \quad n-k-1 \geq 10k \quad \Rightarrow \quad k \leq (n\text{-}1)/11 \text{ und damit auch}$$

$$k_{max} \leq (n\text{-}1)/11 .$$

- Die k_{max} besten Regressoren werden mit *forward solution* (s. Abschnitt 3.3.2) aus einer meistens großen Indikatorenmenge gewählt. Zeitlich verlagerte Zusammenhänge (s. Abschnitt 3.3.1) und Autoregression (durch die Betrachtung der Endogenen als einer der exogenen Basiszeitreihen) werden dabei berücksichtigt.

5.3.2 Erstellung eines neuronalen Modells

Während der Erstellung eines konnektionistischen Modells werden sowohl der Input als auch einige Einstellungen des Neuronalen Netzes bestimmt. Da es kein analytisches Verfahren dafür gibt, wurden im Rahmen der Arbeit einige KI–Ansätze zur Problemlösung ausgearbeitet und verglichen. Das vollständige Problem der Parametrisierung Neuronaler Netze ist wegen seiner hohen Dimensionalität sehr komplex. Deswegen wurde dieses Problem dadurch reduziert, daß:

- die Anzahl der Schichten in Netzen bei der Erstellung neuronaler Modelle konstant gehalten werden: Es werden nur dreischichtige Netze betrachtet, weil die, wie im Abschnitt 2.1.3 begründet wurde, hinreichend sind, um eine nichtlineare Funktion beliebig gut zu approximieren.
- die Lernparameter werden nicht optimiert. Als voreingestellte Einstellungen dieser Lernparameter gelten die in der Dokumentation zu SNNS empfohlenen Werte, die allerdings vom Benutzer geändert werden können.

Der Prozeß der Modellerstellung kann in zwei Schritte eingeteilt werden:

(1) Beginne mit alten Modellindikatoren und einem beliebigen Netz. Trainiere das Netz und ermittle das Optimierungskriterium (z. B. Testfehler);

(2) Optimiere Netztopologie und Netzinput bzgl. dieses Kriteriums.

Dafür wurden folgende Verfahren ausgearbeitet und verglichen:

- Sukzessive Modellerstellung;
- Parallele Modellerstellung mit der „*best–first*"–Suche;
- Parallele Modellerstellung mit dem Selektionsalgorithmus.

Die zwei letzteren Ansätze gehören zu den Methoden der KI.

Im einzelnen werden diese Verfahren in den nächsten Unterkapiteln beschrieben.

5.3.2.1 Ausgangsmodell

Jedes o. g. Verfahren der Erstellung neuronaler Modelle benutzt als Ausgangspunkt:

- Ein dreischichtiges *feedforward*–Netz mit dem *backpropagation*–Lernverfahren und einer möglichst kleinen Anzahl der versteckten Neuronen. Als Aktivierungsfunktion wird Tangens Hyperbolicus benutzt und als Anfangsgewichte werden kleine zufällige

Zahlen (zwischen –0.05 und 0.05) genommen, um lineare Abhängigkeiten möglichst gut abzudecken[25].

- Die während der „linearen" Phase ermittelten Exogenen, Prognoselänge und optimale Anzahl der Exogenen. Das ist folgendermaßen begründet:
  - Wie in [RZi94], Abschnitt 2.3 angegeben wird, decken Neuronale Netze bei den oben beschriebenen Einstellungen der Lernparameter beide nichtlineare und lineare Zusammenhänge ab. Deswegen ist zu erwarten, daß wenn zwei Modelle sich nur in der Prognosemethode unterscheiden, liefert ein neuronales Modell in der Regel bessere Ergebnisse als ein lineares.
  - Das *forward–solution*–Verfahren garantiert mit einer hohen Wahrscheinlichkeit die Erstellung eines bestmöglichen linearen Modells. Damit kann seine Güte als eine sinnvolle untere Schranke für die Güte des neuronalen Modells dienen, die auf der Basis dieses linearen Modells erstellen wird.
  - Es ist zu erwarten, daß wenn wirtschaftliche Zeitreihen zusammenhängen, sie mit einer hohen Wahrscheinlichkeit sowohl lineare als auch nichtlineare Zusammenhänge aufweisen. Mit anderen Worten ist die Existenz der nichtlinearen Abhängigkeiten zwischen den korrelierten Zeitreihen wahrscheinlicher als zwischen den nichtkorrelierten. Zimmerman in [RZi94] schreibt darüber: „Hat man sehr viele Inputs" und „stellt man bei einer Korellationsanalyse zwischen Target und einigen Inputindikatoren fest, daß praktisch überhaupt kein linearer Zusammenhang besteht, so kann man diese Inputindikatoren fallen lassen."[26]
  - Bei einer großen Anzahl der potentiellen Inputzeitreihen (die sich wegen der Berücksichtigung von verzögerten Zusammenhängen vervielfacht) und bei der Nichtexistenz eines analytischen Verfahrens zur Auswahl von Inputzeitreihen, ist es eine einleuchtende Entscheidung.

5.3.2.2 Sukzessive Erstellung eines neuronalen Modells

Bei diesem Verfahren wird die Anzahl der Inputzeitreihen konstant gehalten. Der Vorgang kann in zwei nacheinanderfolgende Schritte unterteilt werden:

(1) Optimierung der Netzstruktur. In diesem Schritt werden folgende Parameter des Netzes eingestellt:

- Topologie: (*feedforward*–, Elman– oder Jordan–Netz);
- Anzahl der Neuronen in der verdeckten Schicht zwischen ihren optional einstellbaren minimalen und maximalen Grenzen;
- *shortcuts*: jeweils mit und ohne.

[25] Empfohlen in [RZi94], Seite 26.

[26] Siehe [RZi94], Abschnitt 2.3, Seite 26.

Für jede mögliche Kombination der o. g. Parameter wird ein entsprechendes Neuronales Netz erzeugt und mit den alten (während der „linearen Phase“ ermittelten) Inputzeitreihen trainiert. Das „beste“ bzgl. des Optimierungskriteriums (z. B. Testfehler, MAPE oder RMSE) Netz wird festgehalten.

(2) Optimierung des Inputs. Dieser Teil des Optimierungsprozesses erfolgt nach dem folgenden Algorithmus:

1. xWähle als Inputzeitreihen mit *forward solution* ermittelte Exogene;
2. Erstelle eine (evtl. sortierte) Liste ***L*** der unbenutzen Indikatoren;
3. Initialisiere das zu minimierende Kriterium ***e*** mit einer großen Zahl;
4. Bis das Abbruchkriterium nicht erfüllt ist, wiederhole:
    5. Trainiere das Netz und ermittle das neue ***e***;
    6. Falls sich ***e*** verringert,
        Führe eine Sensitivitätsanalyse durch und finde die Inputzeitreihe, die den Output am wenigsten beeinflußt, nenne sie ***current;***
    7. Sonst
        ersetze ***current*** durch die obere Zeitreihe aus ***L***, schließe sie aus ***L*** und nenne sie wiederum ***current***.

Abbildung 14: Optimierung des Netzinputs, Algorithmus

5.3.2.3 Erstellung des neuronalen Modells mit der Methoden der KI

Es stellt sich die Frage zur Effizienz und sogar Zulässigkeit der sukzessiven Optimierung der Modellparameter und über den Einsatz der intelligenteren Algorithmen für die Erstellung der neuronalen Modelle durch eine gleichzeitige Optimierung mehrerer Modellparameter. Im Rahmen dieser Arbeit wurden zwei auf Methoden der KI basierende Optimierungsalgorithmen der neuronalen Modelle implementiert und bei empirischen Untersuchungen verwendet:

- heuristische *„best–first“*–Suche;
- *Evolutionärer* Algorithmus (*Selektion* durch *Mutation*).

Beide Verfahren benutzten den gleichen Startpunkt, dieselbe Repräsentation des Suchraumes und die gleiche heuristische Funktion. In den weiteren Unterkapiteln wird zuerst diese gemeinsame Basis beider Algorithmen beschrieben und dann sie selbst.

5.3.2.3.1 Suchraum

Es werden Einstellungen für folgende Parameter des neuronalen Modells gesucht: **Inputzeitreihen** und **Netzarchitektur**. Netzarchitektur wird ihrerseits durch die Topologie des Netzes **T** (z. B. *Elman–*, *Jordan–* und *feedforward–*Netze, jeweils mit und ohne *schortcuts*), die Anzahl von verdeckten Neuronen **H** und die Anzahl der Inputneuronen **I** beschrieben.

Für eine eindeutige Beschreibung des Netzinputs werden alle potentiellen Indikatoren durchnummeriert. Somit kann der Input eines Netzes mit I Inputneuronen als eine geordnete Menge $X^I = (i_1, i_2, .., i_I)$ eindeutig dargestellt werden, wobei i_k der Index der k–ten Inputzeitreihe ist und $i_1 < i_2 < ... < i_I$ erfüllt ist. Die Reihenfolge der Nummerierung von

Indikatoren ist für die verwendeten Algorithmen relevant, weil in dieser Reihenfolge neue Inputzeitreihen ins Modell eingefügt werden. Deswegen sollen eventuell bessere Exogenen kleinere Indizes haben[27].

Die 4 Dimensionen des resultierenden Suchraumes werden in der Tabelle 1 angezeigt.

Tabelle 1: Dimensionen des Suchraumes

Dimension	Beschreibung	Wertebereich
T	Netztopologie	endlich, einstellbar. Z.B.{e, e_s, j, j_s, f, f_s}[28].
H	Anzahl versteckter Neuronen	$\{H_{min}, H_{min}+1,\ldots, H_{max}\}$, I_{min}, I_{max} einstellbar
I	Anzahl der Inputneuronen	$\{I_{min}, I_{min}+1,\ldots, I_{max}\}$, H_{min}, H_{max} einstellbar
$X^I = i_1 i_2 \ldots i_I$, $i_1<i_2<\ldots< i_I$	Inputzeitreihen	Alle I–elementigen geordneten Untermengen der Menge aller Zeitreihen

- **Größe des Suchraumes**

Wegen der Endlichkeit der Wertebereiche von allen Suchdimensionen, ist der resultierende Suchraum auch endlich. Dennoch kann er aus der praktischen Sicht nicht als geschlossen gelten, wegen eines polynomialen Wachstums des Wertebereichs der 4. Dimension (Inputzeitreihen) mit der Anzahl der betrachteten Indikatoren. Der Grad des Polynomes ist dabei gleich I_{max}. Dieses kann wie folgt gezeigt werden: N sei die Anzahl aller betrachteten Indikatoren und I gehört zu $\{I_{min},\ldots,I_{max}\}$, dann kann die Anzahl aller I–elementigen geordneten Untermengen der Menge aller Indikatoren L von unten wie folgt geschätzt werden:

$$L = \sum_{i=I_{\min}}^{I_{\max}} \binom{N}{i} = \sum_{i=I_{\min}}^{I_{\max}} C_i^N = \sum_{i=I_{\min}}^{I_{\max}} \frac{N!}{i!(N-i)!} > \frac{1}{I_{\max}!} \sum_{i=I_{\min}}^{I_{\max}} N(N-1)\cdots(N-i+1) =$$

$$= \frac{1}{I_{\max}!} \sum_{i=I_{\min}}^{I_{\max}} N(N-1)\cdots(N-i+1) = \frac{1}{I_{\max}!}(N^{I_{\max}} + a_{I_{\max}-1}N^{I_{\max}-1} + \ldots + a_1 N + a_o) = .$$

$$\underset{N\to\infty}{=} \frac{N^{I_{\max}} + o(N^{I_{\max}})}{I_{\max}!} \underset{N\to\infty}{\approx} \frac{N^{I_{\max}}}{I_{\max}!},$$

mit $a_j \in \boldsymbol{Z}$ für alle j=0, …, I_{max}-1.

Beispiel: Es sei I_{min}= 4, I_{max}=10. Wenn wir alle 71 in der Datenbasis[29] vorhandenen Exogenen jeweils unter der Berücksichtigung von 10 Verschiebungen untersuchen (das

[27] Die Indikatoren können z. B. bzgl. der Korrelationskoeffizienten oder der mit dem *forward-solution*-Verfahren erstellte Reihenfolge durchnumeriert werden. Eine zuf. Reihenfolge kann auch benutzt werden.

[28] e – Elman-, j – Jordan-, f – *feedforward*-Netz; Endung „_s“ – *shortcuts* zw. Eingabe- u. Ausgabeschichten.

ergibt 710 Indikatoren), ergibt sich $L \cong 10^{21}$. Wenn wir 20 Verschiebungen betrachten, ist $L \cong 10^{24}$.

- **Zustände vs. Knoten des Suchbaumes**

Ein Knoten des Suchbaums wird durch 4 Koordinaten vollständig und eindeutig beschrieben und entspricht einem bestimmten neuronalen Modell:

$$Z := (T,H,I,X) \qquad \text{mit } X = (X_{k1},X_{k2},..,X_{kI}),\ k_1<k_2<...<k_I$$

- **Direkte Nachbarn eines Knotens**

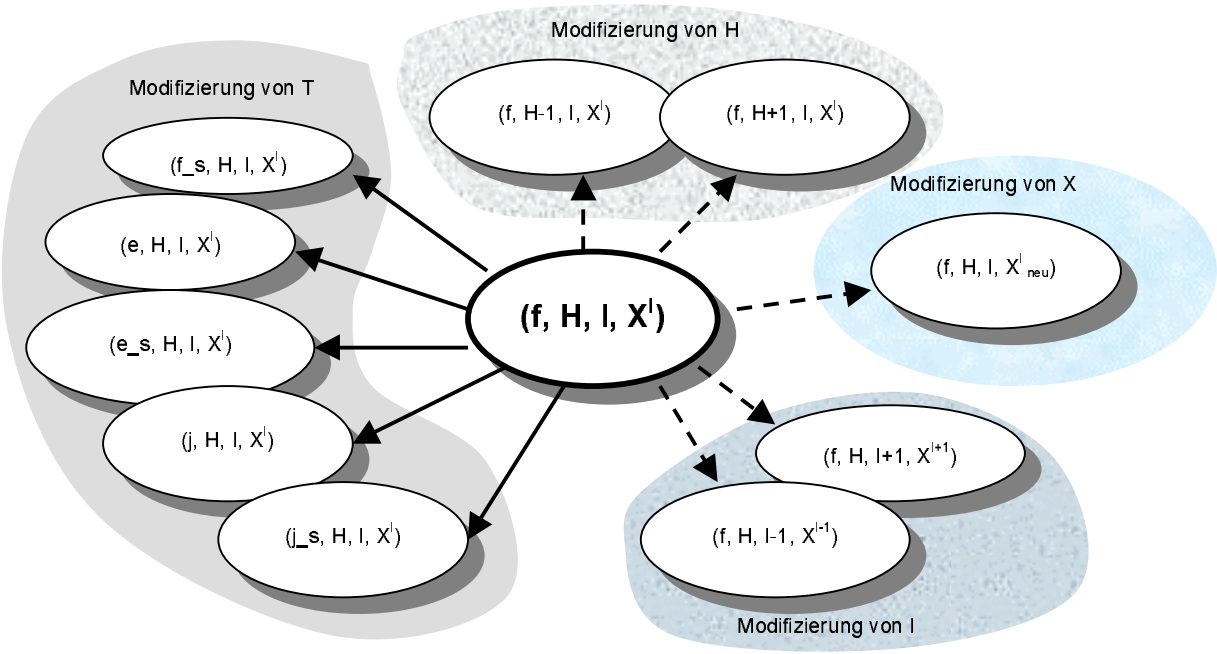


Abbildung 15: Direkte Nachbarn eines Knotens

Alle mit dem Beispielknoten (f, I, H, X) direkt benachbarte Zustände werden in Abbildung 15 gezeigt. Mit punktierten Pfeilen werden die Übergänge gekennzeichnet, die nicht immer vorhanden sind. Modifizierungen von I, H und T sind einleuchtend. Für die Modifizierung von X wird einmalig eine sortierte[30] Liste L aller Indikatoren erstellt und dann der in der Abbildung 16 beschriebene Algorithmus verwendet.

[29] Die empirische Datenbasis der Arbeit wird im 7.1 beschrieben.

[30] : Sortierkriterium läßt sich variieren(z. B. Korrelationskoeffiziente).

1. Trainiere das aktuelle Netz auf dem Input *X* des Ausgangsknotens und ermittle seinen Testfehler.
2. Markiere alle Zeitreihen in *L*, die im Modell bereits vorkommen.
3. Führe die Sensitivitätsanalyse des aktiven Zustandes durch und bestimme die unsignifikanteste Inputzeitreihe *old*.
4. Falls *L* nur markierte Zeitreihen enthält gehe zu 9.
5. Ersetze in *X old* durch die erste nicht markierte Zeitreihe *new* der Liste und markiere sie.
6. Trainiere das Netz mit modifiziertem Input *X* und ermittle seinen Testfehler.
7. Falls der neue Fehler kleiner als der alte ist, liefere den Zustand mit dem modifizierten *X* zurück und gehe zu 9.
8. sonst weise *new* = *old* zu und gehe zum Schritt 4.
9. Abbruch

Abbildung 16: Modifizierung der Dimension X (Netzinput), Algorithmus

- **Umgebung eines Knotens**

Es wird unterschieden zwischen den direkten Nachbarn eines Zustandes, die in der Abbildung 15 gezeigt wurden, und der unterschiedlich interpretierten Menge seiner Nachfolger, oder seiner *Umgebung*, die bei der Suche benutzt wird. Verschiedene Umgebungskonzepte werden für den Fall eines zweidimensionalen Suchraums durch die Abbildung 17 illustriert.

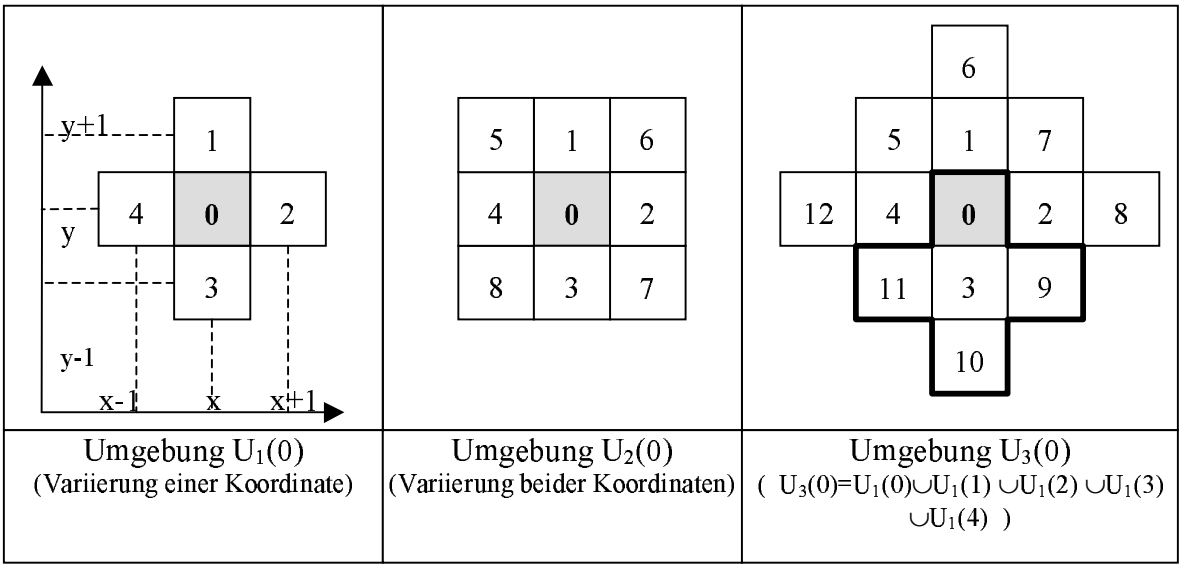


Abbildung 17: Konzepte der Umgebung des Knotens 0 in mehrdimensionalem Raum

Es ist leicht zu sehen, daß aus der Umgebung U_1, die durch das „Wackeln“ jeweils nur einer Koordinate entsteht (wie bei der Definition von direkten Nachbarn eines Knotens in Abbildung 15), diagonal liegende Nachbarknoten unbegründet ausfallen, obwohl sie nicht viel „weiter“ vom Ausgangsknoten als senkrecht liegende entfernt sind. Aus dieser Sicht ist die Umgebung U_2 optimal. Der Vorteil der Umgebung U_3 ist, daß sie einerseits die

Umgebung U_2 enthält und andererseits eine Komposition mehrerer Umgebungen U_1 darstellt und sich dadurch effizient implementieren läßt.

5.3.2.3.2 Heuristik

Die Güte jedes Zustandes wird durch den Testfehler des entsprechenden, trainierten Neuronalen Netzes $E(Z)=E(T,H,I,X)$ charakterisiert. Diese Funktion wird als das zu minimierende Optimierungskriterium benutzt. Man betrachte die Menge der Punkte $(T,H,I,X,E,(T,H,I,X))$ des 5–dimensionalen Raums für alle zulässigen Zustände Z. Im Idealfall, wenn $E(T,H,I,X)$ stetig ist, bilden diese Punkte eine Oberfläche in diesem Raum. Für die Anwendbarkeit einer Heuristik E ist aber bereits eine schwächere Bedingung ausreichend, die in [Bra97] „Quasi–Stetigkeit“ genannt wird. Darunter wird verstanden, daß „ähnliche Zustände ähnliche Güte haben“ (s. [Bra97], S. 178). Mit anderen Worten soll die Güte der Nachbarzustände stochastisch um die Güte des Ausgangszustandes stochastisch herum schwanken. Offensichtlich, daß je „quasi–stetiger“ die Heuristik $E(Z)$ ist, desto erfolgreicher wird die Optimierung von Z mit der heuristischen Suche bzgl. $E(Z)$. Sonst, wenn $E(Z)$ diese Eigenschaft nicht besitzt (was *ad hoc* nicht unbedingt bekannt ist), ist sie als Heuristik unbrauchbar und soll ersetzt werden. Leider ist die Definition der „Quasi–Stetigkeit“ zu informell um benutzt zu werden. Es kann lediglich intuitiv angenommen werden, daß E diese Eigenschaft besitzt. Im Gange der Experimente hat sich gezeigt, daß $E(Z)$ in einem bestimmten Maße diese Bedingung erfüllt.

Im Kontext der Selektion wird die Heuristik *Fitneß* genannt und im Laufe des Selektionsprozesses evaluiert (siehe Seite 43).

5.3.2.3.3 „best–first“ Suche

Für die Optimierung des Modells wurde der *best–first*–Suchalgorithmus mit *backtracking* verwendet aus dem Grund, daß damit (im Unterschied zu z. B. *generate–and–test* oder *Hill–Climbing*) das Verhalten der Fehlerfunktion in der Umgebung eines Knotens vollständiger untersucht wird. Aus demselben Grund werden zwei Varianten dieses Algorithmus verwendet, die sich nur bei der Generierung der Nachfolger eines Zustandes (seiner Umgebung) unterscheiden. Bei einem Algorithmus werden als Nachfolger eines Knotens nur seine direkten Nachbarn betrachtet, beim anderen werden auch ihre direkten Nachfolger in die Nachfolgerliste aufgenommen. Dieses entspricht den Konzepten von Umgebung_1 bzw. Umgebung_3 in der Abbildung 17.

Als Startzustand wird $Z_0=(ff, Imin, Hmin, X_0^{Imin})$ mit $X_0^{Imin} = (0, 1, ..., I_{min-1})$ benutzt.

Der Algorithmus ist parametrisierbar:

- *Toleranz* ist eine kleine Konstante, die einerseits eine *unwesentliche* (kleinere als *Toleranz*) Verschlechterung der Heuristik und damit das Verlassen eines lokalen Minimums erlaubt und, andererseits, eine unwesentliche Verbesserung der Fehlerfunktion ignoriert, um den Optimierungsprozeß abbrechen zu können, wenn er zu langsam wird und sich nicht mehr lohnt.

- *Maximale Anzahl der Schritte* zwischen *wesentlichen* Verbesserungen des Modells. Falls sie überschritten wird, wird die Suche abgebrochen.

1. Bewerte den Startzustand $Z_0=(T_1, H_{min}, I_{min}, X^{I_{min}}{}_0)$;
2. Wenn das Abbruchkriterium erfüllt ist gehe zu 10;
3. Nehme Z_0 in den Suchpfad auf;
3. Erstelle die Nachfolgerliste von Z_0:
   - 3.1 Erstelle die *Umgebung*[31] von Z_o
   - 3.2 Schließe die bereits vorhandene im Suchpfad Zustände aus;
   - 3.3 Bewerte die gebliebenen, sortiere sie nach ihren Bewertungen und schließe die *wesentlich* schlechtere als Z_0 aus seiner *Umgebung;*
   - 3.4 Die gebliebenen Umgebungsknoten nehme in die Nachfolgerliste von Z_0 auf;
6. Wenn Nachfolgerliste von Z_0 leer ist, tue:
   - 7. Wenn der Pfad leer ist, gehe zu 10.
   - 8. Sonst
     - 8.1 schließe Z_0 aus den Pfad;
     - 8.2 nenne den letzten Zustand im Pfad Z_0;
     - 8.3 gehe zu 6.
9. Sonst: Nenne den besten Nachfolger Z_0, schließe ihn aus der Nachfolgerliste und gehe zu 3;
10. Abbruch

Abbildung 18: Optimierung neuronaler Modelle mit der „best–first"–Suche, Algorithmus

5.3.2.3.4 Selektion

Ein weiterer Ansatz ist die Verwendung des *Genetischen* (oder *Evolutionären*) Verfahrens [Bra97]. Die Idee, evolutionäre Algorithmen für die Optimierung von Problemen einzusetzen, die sich mit analytischen Methoden nicht effizient lösen lassen, wurde bereits in der sechziger Jahren untersucht und bekommt seit Ende der achtziger Jahre ein wiederbelebtes Interesse.

- **Grundlagen**

Im Kontext der evolutionären Algorithmen wird eine Näherungslösung des Problems (in unserem Fall – ein neuronales Modell) als *Individuum* bezeichnet. Die Individuen werden

[31] Eine Menge der Nachbarknoten entsprechend dem verwendeten Umgebungskonzept (s. Abschnitt 5.3.2.3.1).

schrittweise verbessert. Das Resultat eines Verbesserungsschrittes wird als *Nachkomme* bezeichnet. Für die Erzeugung von Nachkommen wird *Rekombination* (Verwendung mehrerer sog. *Eltern*) oder *Mutation* (Verwendung nur eines Eltern–Individuums) benutzt. Evolutionäre Algorithmen besitzen ein Gedächtnis für erfolgversprechende Lösungsansätze, genannt *Population*. Mehrere Nachkommen, die in einem Iterationsschritt erzeugt werden, werden als Generation bezeichnet.

Ein Iterationszyklus setzt sich aus vier Teilen zusammen (siehe Abbildung 19). Zuerst werden aus der Population die Individuen mit höherer *Fitneß* (oder Güte) als Eltern ausgewählt. Dabei, um beide *Exploration* und *Exploitation* des Suchraums auszunutzen, ist „nicht nur das beste Individuum als Elter auszuwählen, sondern auch mit monoton fallender Wahrscheinlichkeit auch das nächstbeste“[32] (sog. *Ranking* – bzw. *Fitneß Based Selection*)[33]. Dann werden neue Nachkommen erzeugt und evaluiert. Um die Populationsgröße konstant zu halten, wird aus alten Populationsmitgliedern und den Nachkommen (sog. (μ+λ)–*Strategie*) oder ausschließlich aus Nachkommen ((μ,λ)–*Strategie*) die neue Population selektiert.

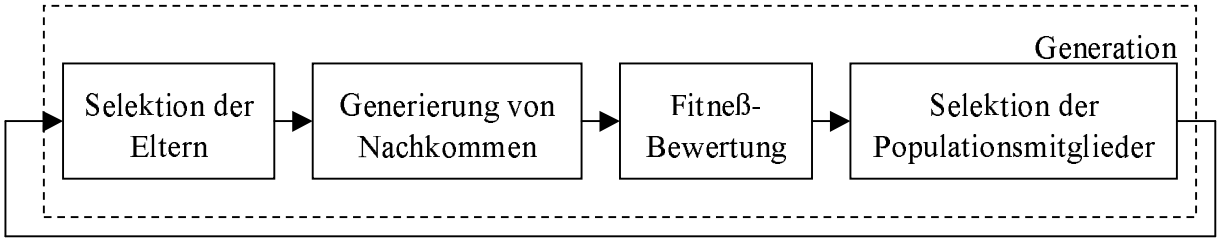


Abbildung 19: Evolutionszyklus

Auch hierbei werden die Individuen mit höherer Fitneß bevorzugt. Die Steuerparameter des Evolutionszyklus können sowohl fest als auch selbst Gegenstand evolutionärer Optimierung sein.

Unter den Populationsmodellen werden einfache Menge, Inselmodell und Nachbarschaftsmodell unterschieden. Bei der einfachen Menge wird die Selektion global durchgeführt. Beim Inselmodell läuft die Evolution auf verschiedenen Inseln getrennt ab, wobei allerdings eine kleine *Migrationsrate* vorausgesetzt werden kann. Beim Nachbarschaftsmodell dürfen nur die Individuen Nachkommen erzeugen, die in ihrer lokalen Nachbarschaft die beste Fitneß haben. Zusätzlich können weitere Phänomene der realen Evolution nachgemacht werden.

- **Spezifizierung des Algorithmus**

Ein Knoten des Suchbaums wird als ein Individuum und Testfehler des entsprechenden Neuronalen Netzes als Fitneß dieses Individuums betrachtet. Dabei, um den *Overfitting*–

[32] [Bra97], S. 178.

[33] Siehe [Bra97], Abschnitt 4.1.4.2.

Effekt zu vermeiden, wird, wie es bei [Bra97][34] empfohlen wird, bei jeder Generation eine neue Testmenge gewählt, und die Fitneß der Populationsmitglieder wird durch die Ermittlung der durchschnittlichen Fitneß korrigiert[35]. Als Nachkommen eines Individuums wurden seine direkten Nachbarn betrachtet (*Umgebung*$_1$ in der Abbildung 17).

Bei der Elternselektion wurde *Ranking Based Selection* benutzt, die auf der Basis eines speziellen Zufallsgenerators implementiert wurde. Die Stärke der Konzentration generierter Zahlen im vorderen Bereich der Reihenfolge ist einstellbar. Ein Elter kann nur einmal für die Generierung von Nachkommen benutzt werden.

Es wurde das einfachste Populationsmodell gewählt, wobei alle Individuen in einem globalen „Wettbewerb“ teilnehmen. Bei der Selektion der Populationsmitglieder wird die $(\mu+\lambda)$–*Strategie* gewählt um die beste Lösungen festzuhalten, wobei Populationsgroße $\mu = 40$ und Anzahl der Nachkommen $\lambda = 10$ festgehalten werden, so wie es im [Bra97] (Seite 201) empfohlen wird.

Die Startpopulation wird aus etwa gleichen Teilgruppen für jede betrachtete Netztopologie zusammengesetzt. Die Menge der Topologien ist einstellbar (z. B. *feedforward*–, Elman– und Jordan–Netze, jeweils mit und ohne *shortcuts*). Die Individuen jeder Gruppe werden zufällig ausgewählt, allerdings werden die Individuen mit kleineren Anzahlen der Gewichte mit einer hohen Wahrscheinlichkeit bevorzugt.

5.3.3 Validierung eines Modells und Erstellung der Prognose

Wie in [CoM96], insbesondere für kleine Datenmengen, empfohlen wird, wurde für die Beurteilung der Modelle eine zusätzliche Validierung verwendet: Eine Untermenge der Daten (üblicherweise eine feste Anzahl der zuletzt bekannten Werte) wird aus dem Trainingsraum ausgeschlossen. Nach der Erstellung und der *ex*–anten Beurteilung eines (sowohl linearen als auch neuronalen) Modells, wird eine auf dem ermittelten Modell basierende Prognose für den Validierungszeitraum erstellt, und eine *ex–post*–Analyse wird durchgeführt. Diese Beurteilung, die *Validierung* genannt wird, gilt als Hauptkriterium bei der Entscheidung über die Verwendbarkeit des Modells: Falls der *Theil*–sche Koeffizient[36] der Prognose unter 1 liegt (und somit die „*naive*“ Prognose besser als die mit dem Modell ermittelte ist), wird das Modell als nicht zufriedenstellend beurteilt.

Als der Umfang der Validierungsmenge wird für beide lineare und neuronale Modelle die empfohlene maximale Prognoselänge (s. nächsten Abschnitt) genommen, um diese Prognosemethoden bei gleichen Rahmenbedingungen zu vergleichen.

Falls die Qualität der Prognose für den Validierungszeitraum zufriedenstellend ist, wird mit demselben Modell die unbekannten Zukunftswerte der Endogene generiert.

---

[34] Siehe [Bra97], Abschnitt 4.1.4.3.

[35] Als Fitneß—Wert berechnet man den durchschnittlichen Generalisierungsfehler über die aktuelle und alle vorherige Test—Mengen.

[36] Siehe Kapitel 4.

6 Prognosesystem der Universität Osnabrück

Für die Durchführung der empirischen Experimente wurde das an der Universität Osnabrück von Frank Thiesing entwickelte Prognosesystem [Thi98] benutzt, das während der Arbeit um einige Funktionalitäten sowie um ein neues Projekt erweitert wurde. Als ein Werkzeug für diese Erweiterungen diente die objektorientierte Klassenbibliothek `MatClass`, die eine effiziente Programmierung der Linearen Regression und Matrix–Syntax erlaubt.

Diese bereits vorhandenen Werkzeuge sowie eigene Programme werden in diesem Abschnitt erläutert.

6.1 Funktionalität und Bestandteile

Das Prognosesystem der Universität Osnabrück stellt Lösungen für folgende Aufgaben bereit:

- Zeitreihenverarbeitung und –speicherung in einheitlichem Format. Es werden viele unterschiedliche Transformationen von Zeitreihen mit der automatischen inversen Transformation unterstützt;
- Zeitrechnung in verschiedenen Zeiteinheiten ab Jahr 1945.
- Gute Visualisierung von Zeitreihen, wobei gleichzeitig mehrere Zeitreihen mit unterschiedlichen Zeiteinheiten beobachtet werden können.
- Prognose von Zeitreihen, hauptsächlich mit Neuronalen Netzen. Als Simulator Neuronaler Netze dient SNNS (s. Abschnitt 6.1.1).
- *ex*–ante Analyse der Modellindikatoren: Korrelationsanalyse für die Entdeckung linearer Abhängigkeiten (allerdings mit GAUSS) und Sensitivitätsanalyse für die Beurteilung der Signifikanz der Indikatoren in einem neuronalen Modell.
- Batch–Betrieb, der das Laufen des Systems ohne Interaktion mit dem Benutzer erlaubt. Dazu gehört ein einheitliches Format der Infodateien (*templates*).
- Prognosefehleranalyse nach unterschiedlichen Fehlermaßen.
- Verwaltung der verschiedenen, hierarchisch organisierten Projekten und Erweiterbarkeit des Systems um neue Projekte.

Verschiedene Programmierwerkzeuge und Techniken, die für o. g. Zwecke benutzt wurden, werden in Abbildung 20 angezeigt. Eine vollständige Beschreibung des Prognosesystems findet man in [Thi98]. In weiteren Unterkapiteln werden kurz die im Kontext dieser Arbeit wichtige Fragmente des Systems beschrieben: Simulation und Sensitivitätsanalyse Neuronaler Netze.

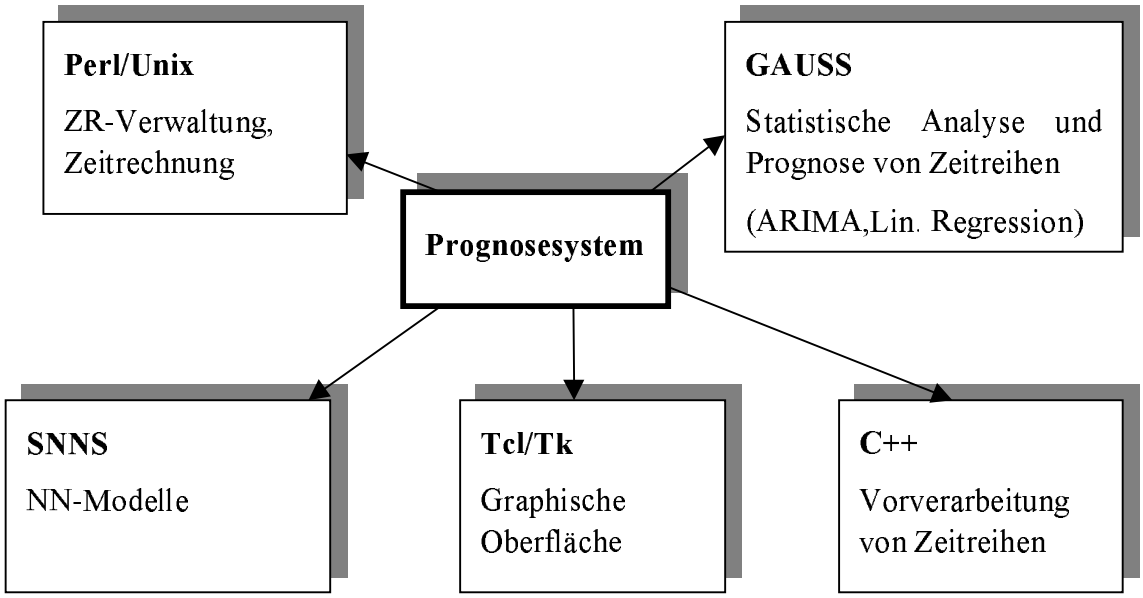


Abbildung 20: Das Prognosesystem

6.1.1 Simulation Neuronaler Netze

Für die Simulation Neuronaler Netze wird im Prognosesystem der **S**tuttgarter **N**euronale **N**etze **S**imulator (**SNNS**) verwendet .

SNNS ist ein effizienter Simulator künstlicher Neuronaler Netze auf Unix–Workstations. Er erlaubt Generierung, Training (mit Tests) und Visualisierung Neuronaler Netze. Entsprechend diesen Aufgaben besteht er aus drei großen, teilweise unabhängigen Teilen:

1. einem Simulatorkern, der Aktionen der Lern– und Arbeitsphase eines Neuronalen Netzes durchführt;
2. dem Compiler, der Neuronale Netze aus einer Netzwerk–Beschreibungssprache generiert;
3. der graphischen Benutzeroberfläche (wird im Prognosesystem nicht benutzt).

Das Prognosesystem übernimmt die Generierung von „*pattern*"–Mengen im SNNS–Format aus den Zeitreihen und versteckt Aufrufe von SNNS–Werkzeugen vor dem Benutzer. Die reiche Funktionalität von SNNS kann jedoch über die Einstellungen in Infodateien vollständig benutzt werden (siehe [Zel95]).

Außerdem wurde SNNS um ein zusätzliches Lernverfahren `BPM_ln_cosh` erweitert, das innerhalb des Lernalgorithmus `BackpropMomentum` anstatt einer quadratischen Zielfunktion die lineare Funktion $E = | t - o |$ benutzt[37]. Diese Fehlerfunktion hat sich bei

[37] Dies ist nicht ganz korrekt. Um die Differenzierbarkeit der Zielfunktion im Null zu erreichen wurde sie durch die folgende ersetzt: $E = ln\,(\,cosh\,(\,o - t\,)\,)$.

der ökonomischen Anwendung Neuronaler Netzen als geeignet gezeigt, weil sie die quadratische Übergewichtung von großen Fehlern, die Ausreißer verursachen, vermindert.

6.1.2 Sensitivitätsanalyse Neuronaler Netze

Es ist möglich mit dem Prognosesystem eine Sensitivitätsanalyse (vs. Signifikanzanalyse) einzelner Inputzeitreihen eines Neuronalen Netzes durchzuführen und ihre Ergebnisse zu visualisieren. Die Sensitivitätsanalyse wird während einer üblichen Prognose durchgeführt, wenn ein Block mit entsprechenden Anweisungen in die Infodatei hinzugefügt wird. Dabei kann der Analysealgorithmus durch mehrere Parameter modifiziert werden. Die am häufigsten verwendete Variante wird im Abschnitt 2.1.5 beschrieben, eine detailliertere Beschreibung befindet sich in [Thi98], Abschnitt 8.10.

Dieses Werkzeug des Prognosesystems wurde bei den implementierten Optimierungsverfahren verwendet.

6.1.3 Dateiformate

Wie es im Abschnitt erwähnt wurde, kann das System im Batch–Betrieb laufen. Viele Parameter, die das System für die Erstellung der Prognose (sowohl mit Linearer Regression als auch mit Neuronalen Netzen) braucht, werden aus den Dateien im speziellen Format extrahiert. Deswegen spielt die Verwendung solcher Dateien im System eine wesentliche Rolle. Weiter werden sie *templates* oder *Infodateien* genannt.

Leider sind die Formate von *templates* für die Prognoseerstellung mit Linearer Regression und mit Neuronalen Netzen noch nicht ganz einheitlich. Dieses Problem läßt sich allerdings mit einem einfachen Konverter, der während der Arbeit geschrieben wurde, leicht lösen.

6.2 Erweiterung

Das vorhandene Prognosesystem bietet eine reiche und bequeme Funktionalität an. Dennoch befindet es sich in einer Entwicklungsphase und zu Beginn der Arbeit hatte es einige verbesserungs– bzw. erweiterungsbedürftige Punkte:

- Lineare Regression stand zwar im System zur Verfügung, wurde aber mit der Verwendung von GAUSS implementiert. Da GAUSS sogar in der Universität nur auf wenigen Rechnern lief, machte es das „lineare“ Teil des Prognosesystems nicht transportabel und ineffizient. Außerdem brauchte der jeweilige Benutzer des Systems extra einen Zugang, um GAUSS zu benutzen. Interaktion zwischen dem Prognosesystem und GAUSS erfolgte nach dem Remote–Einlogen (`rlogin`) des Benutzers auf einem GAUSS–Rechner. Nach dem Beenden der Berechnungen wurden die Ergebnisse zurück für eine weitere Bearbeitung transferiert.

- Eine automatische Zusammensetzung eines linearen Modells wurde nicht implementiert. Ein lineares Modell wurde vom Hand erstellt. Dieser Prozeß konnte aber nach dem in [Edw84] beschriebenen Algorithmus leicht implementiert werden.
- Die Unterstützung des Batch–Betriebs ist zwar eine gute Voraussetzung für die Implementierung der dauerhaften Optimierungsalgorithmen, es wurde aber kein solcher Algorithmus implementiert. Die konnektionistischen Modelle wurden mit der „*trial–and–test*“ Methode erstellt.
- Es gab wenige Werkzeuge für die Aggregation der Zeitreihen (z. B. für die Erstellung einer wöchentlich aggregierten Zeitreihe aus einer täglich erfaßten Zeitreihe oder umgekehrt).

Diese Aufgabenstellung motivierte eine Erweiterung des Systems im Rahmen der Arbeit. Für die Programmierung wurden hauptsächlich die Programmiersprachen Perl und C++ verwendet.

Bei den Implementierungen von C++–Programmen wurden Objekte der entwickelten C++–Klassenbibliothek „`regression`“ benutzt, die auf der bereits vorhandenen Klassenbibliothek `MatClass` (s. Abschnitt 6.2.1.1) basiert. Üblicherweise sind Aufrufe dieser Programme in Perl–Skripte eingebettet, die als eine Schnittstelle zum übrigen Prognosesystem dienen.

Schematisch ist die Interaktion verschiedener Module in der Abbildung 21. dargestellt.

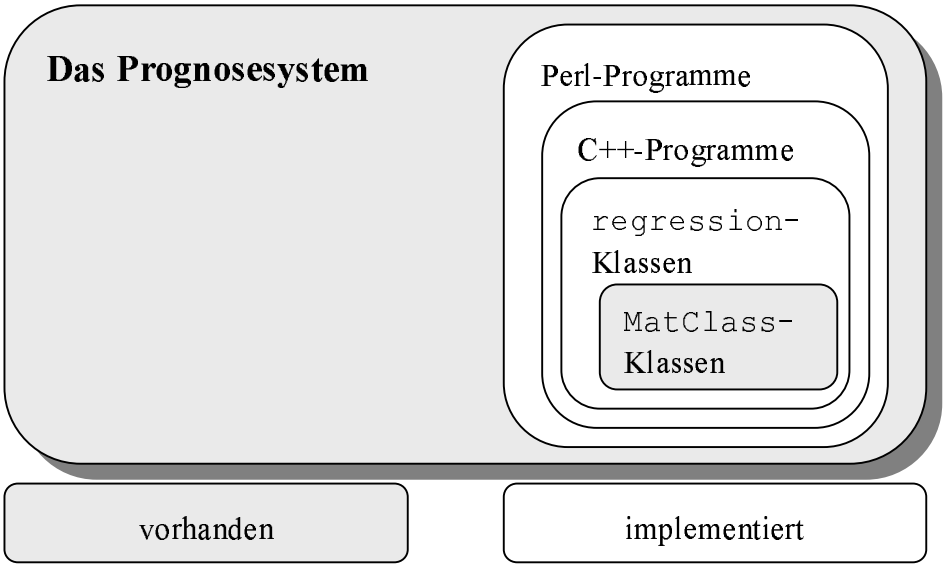


Abbildung 21: Erweiterung des Prognosesystems

In nächsten Unterkapiteln werden die wichtigsten implementierten Werkzeuge beschrieben. Genauere Spezifikationen der Programme (Aufrufe, benötigte Dateien) sowie Deklarationen der öffentlichen Methoden von erstellten Klassen befinden sich in den Anlagen A, B und C des Anhangs.

6.2.1 C++

6.2.1.1 Klassenbibliothek `MatClass` als Entwicklungsbasis

> "The central aim of MatClass project is to go some way in evaluating of the potential of an object–oriented approach to econometric computation using C++."[38]

Die an der Universität Manchester entwickelte nichtkommerzielle Bibliothek `MatClass` enthält eine Sammlung der Klassen für die objektorientierte Entwicklung wirtschaftlicher und statistischer Computeranwendungen.

Der Erfolg von GAUSS zeigt, daß viele Wirtschaftswissenschaftler gerne eigene Programme für die Erstellung und Analyse wirtschaftlicher Modelle entwickeln. Bis heutzutage werden dabei GAUSS und FORTRAN am häufigsten benutzt. Beide Werkzeuge haben eigene Vor– und Nachteile. GAUSS hat eine bequeme Matrix–Syntax, leidet aber am Mangel der Kompatibilität und Effizienz, weil es bereits ein kompiliertes Code enthält und sich nicht auf dem Zielrechner neu kompilieren läßt. FORTRAN–Bibliotheken wie z.B. "`LINPACK`" oder "`Numerical Recipes`", sind kompatibler und effizienter, weil die o. g. Pakete Source–Code enthalten und können auf jedem Rechner mit den dort vorhandenen Bibliotheken zusammen kompiliert werden. FORTRAN stellt aber zu wenig Aggregation– und Kontrollstrukturen zur Verfügung. Weder FORTRAN noch GAUSS bieten Möglichkeiten zur Erstellung eigener erweiterter Datentypen.

Die neuartige objektorientierte C++–Bibliothek MatClass hat o. g. Nachteile nicht. Durch die Benutzung von solchen Prinzipien der Objektorientierung wie Kapselung und Vererbung erhöht sich die Verständlichkeit der Programme und die Effizienz ihrer Entwicklung. Die Bibliothek enthält viele erweiterbare, teilweise abstrakte Klassen und externe Funktionen: sowohl Standardfunktionen für die Anwendung auf Matrizen als auch einige spezielle statistische Funktionen (Generatoren der statistischen Verteilungen u. a.). Überladene C++–Operatoren unterstützen einen bequemen Matrix–Syntax. Die wichtigsten Klassen der Bibliothek werden auf der Abbildung 22 dargestellt.

Alle Klassen stammen von der abstrakten Klasse `matObject` ab, die eine Fehlerbehandlung verwaltet. Unter weiteren wichtigen Klassen sind `matrix` und abstrakte Klassen wie `matFile`, `matDec` und `matOls`.

Die Klasse `matrix` ist eine zentrale Klasse der Bibliothek, weil alle anderen Klassen mit den Objekten dieser Klasse operieren.

Die Klasse `matFile` hat zwei Unterklassen `inFile` und `outFile`, die eine breite Funktionalität beim Einlesen und bei der Formatierung der Ausgabe kapseln. Aufgabe der Klasse `matDec` ist, lineare Gleichungssysteme mit Dekomposition zu lösen. Ihre instanzierbare Unterklassen qrhDec, cholDec und `luDec` unterscheiden sich nur in den numerischen Methoden der Dekomposition und haben gleiche „öffentliche“ Methoden, wie

---

[38] Siehe [Bir93], Seite 11.

ihre Oberklasse. Alle Unterschiede sind in privaten Methoden gekapselt. Dasselbe gilt für die abstrakte Klasse `matOls` und ihre Unterklassen `olsChol`, `olsQrh` und `olsSvd`, die Lineare Regression mit der *Methode der kleinsten Quadrate* (siehe Abschnitt 3.1) lösen.

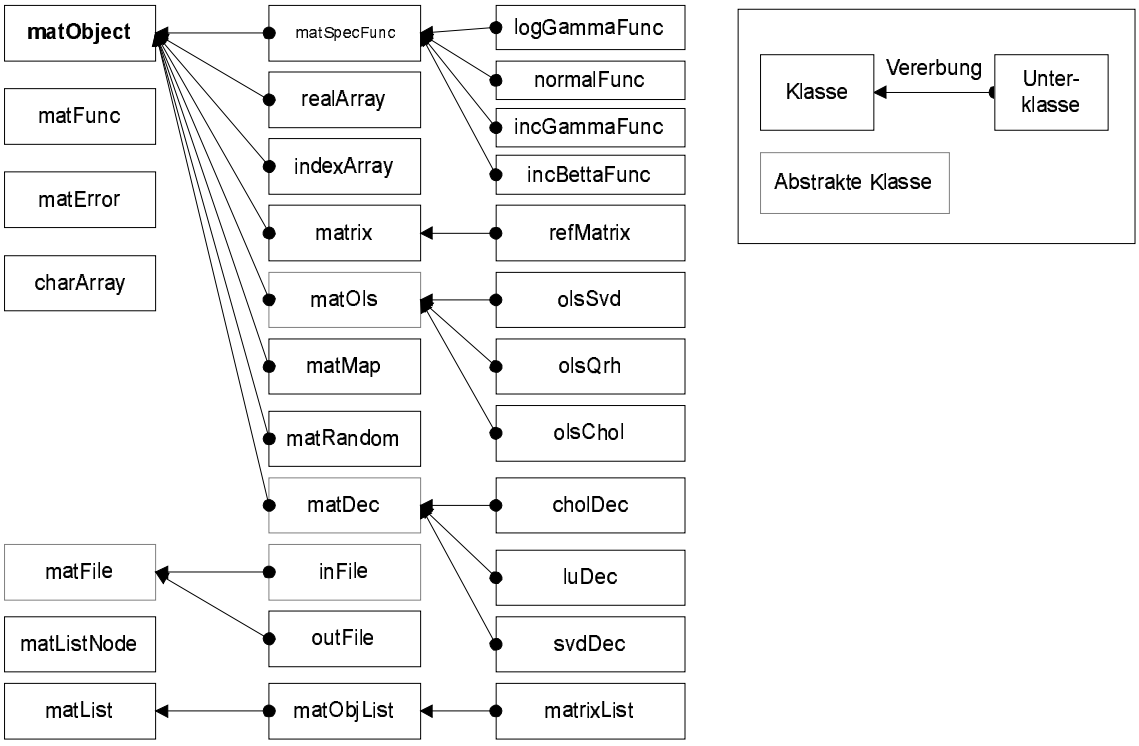


Abbildung 22: Klassenbaum von `MatClass`

Leider, ist die heutige Version von `MatClass` noch nicht ganz fehlerfrei. Allerdings ist es ein gutes Werkzeug für eine objektorientierte und dadurch effiziente Entwicklung von qualitativen wirtschaftswissenschaftlilchen Computeranwendungen.

Eine detaillierte Beschreibung der Bibliothek `MatClass` sowie Beispiele ihrer Verwendung und Überlegungen über die Objektorientierung bei der Erstellung der ökonomischen Applikationen findet man in der Dokumentation zu dieser Bibliothek [Bir93].

6.2.1.2 Implementierte Klassenbibliothek „regression“

Die entwickelte C++–Klassenbibliothek „`regression`“, basiert auf der Klassen–bibliothek `MatClass` und enthält folgende wiederverwendbare Klassen:

- `TimeSeries`
- `Reader`
- `Synchronizer`

- `Regression`
- `Prognose`
- `Model`
- `Correlation`

Alle Klassen stammen von `Object` und benutzen Objekte der Bibliothek „`MatClass`“. Instanzvariablen werden gekapselt und sind ggf. über öffentliche `set/get`–Methoden zugänglich. In folgenden Unterkapiteln werden die Klassen nur generell aus der Sicht ihrer Funktionalität beschrieben. Deklarationen von einzelnen Methoden aller Klassen befinden sich in der Anlage B. Falls ein tieferes Interesse besteht, befinden sich dort sowohl ein Ausdruck der Header–Datei „`regression.hpp`“ mit den Klassendeklarationen als auch eine CD mit Code.

6.2.1.2.1 TimeSeries

Ein Objekt dieser Klasse kapselt eine Zeitreihe, mit allen dazugehörigen Attributen wie Name, Typ (Endogene oder Exogene), Verschiebung, Start, Ende und eigentliche Werte und erlaubt das Setzen und Abfrage von jedem Attribut. Werte sind in einem `matrix`–Objekt enthalten und können sowohl in diesem Container als auch einzeln (für einen bestimmten Zeitpunkt) abgefragt werden. Fehlende Werte werden mit einer vordefinierten Konstante `MISSING` dargestellt. Üblicherweise erzeugt ein `Reader`–Objekt TimeSeries–Objekte entsprechend ihren Deklarationen in der Infodatei, speichert sie in einem Vektor und stellt den anderen Objekten bzw. Prozessen zur Verfügung.

6.2.1.2.2 Regression

`Regression`–Objekte bekommen eine Endogene und Exogenen, erzeugen eine synchronisierte Matrix der Zeitreihen–Werte, fügen eine konstante Exogene (Scheinregressor) hinzu und führen die Lineare Regression aus. Dafür wird ein `matOls`–Objekt der Bibliothek MatClass benutzt, das die Implementierung der Methode der kleinsten Quadrate kapselt. Dabei können Cholesky, QR oder SVD –Dekompositionen je nach der Einstellung verwendet werden. Ergebnisse werden in Instanzvariablen gespeichert und können später mehrmals abgefragt werden, unter anderem:

- lineare Koeffizienten;
- Unterschiedliche Beurteilungen der einzelnen Exogenen wie Standardfehler, *t*–Wert, Signifikanz anhand des *t*–Tests, u. a.;
- Beurteilungen des ganzen Modells wie sein Bestimmtheitsmaß, F–Wert, Signifikanz anhand des F–Tests, u. a.

6.2.1.2.3 Correlation

`Correlation`–Objekte werden mit einer Endogene, einer Exogene und den Grenzen der Verschiebung initialisiert. Sie erstellen aus der exogenen Basiszeitreihe alle zu untersuchenden Indikatoren[39] und führen die Korrelationsanalyse für jeden Indikator durch. Danach kann die Liste aller Korrelationskoeffizienten sowie der maximale Koeffizient und die entsprechende optimale Verschiebung jede Zeit abgefragt werden. Es ist möglich eine Exogene dynamisch zu wechseln, ohne das `Correlation`–Objekt vollständig neu zu initialisieren

6.2.1.2.4 Model

`Model`–Objekte erstellen ein optimales lineares Modell mit dem *forward–solution*–Verfahren. Als Input–Information bekommen sie

- eine Endogene;
- eine Menge potentieller Exogene;
- minimale und maximale Grenzen berücksichtigter Verschiebungen;
- die maximale Anzahl der auszuwählenden Modellindikatoren (optional) und
- die Angaben über Validierungmenge, die von allen Berechnungen ausgeschlossen wird, durch zwei von drei folgenden Parametern: `VAL_PAIRS`, `FIRST_PROG` und `LAST_PROG`. Die Angabe von `LAST_PROG` ist optional, voreingestellt ist das Ende des gemeinsamen Zeitsegments.

Als erstes werden alle potentiellen Modellindikatoren erzeugt. Dann wird anhand der Länge des Trainingsraums (gemeinsames Zeitsegment für die Indikatoren und Endogene), die maximale (vs. optimale) Anzahl der Modellindikatoren und die maximale Prognoselänge berechnet.

6.2.1.2.5 Prognose

`Prognose`–Objekte sagen eine Endogene für den voreingestellten Zeitraum mit einer der drei vordefinierten Methoden vorher: `SIM`, `REAL` oder `SINGLE`. Der Unterschied zwischen den Methoden ist nur im Fall der (partiell) autoregressiven Modelle relevant, wenn die verschobene Endogene auch als eine Exogene vorkommt und als „`UNKNOWN`“ gekennzeichnet wird. Falls die zu prognostizierenden Werte in der Tat bereits bekannt sind (zu einer Validierungsmenge gehören), können drei Varianten der Prognosemethode verwendet werden:

- Im Fall `SIM`–Methode benutzt man in der `UNKNOWN`–Zeitreihe ihre tatsächlichen Werte;

[39] durch die Verschiebung der Basiszeitreihe zwischen den angegebenen Grenzwerten jeweils um eine Zeiteinheit.

- im Fall `REAL` — berechnete Werte sukzessiv;
- `SINGLE` ist ähnlich zu `REAL`, nur mit folgendem Unterschied: Nach der Berechnung jedes Prognosewertes werden die linearen Koeffizienten anhand des verlängerten Trainingsraum neu berechnet.

6.2.1.2.6 Synchronizer

Die Hauptaufgabe eines `Synchronizer`–Objektes ist, die (eventuell bereits verschobenen) Zeitreihen eines Modells über ein gemeinsames Zeitsegment zu synchronisieren. Dabei werden die Werte der synchronisierten Zeitreihen in einer Matrix zurückgegeben, in der jede Zeitreihe durch eine Spalte dargestellt wird. Zusätzlich können mit der Hilfe eines Synchronizer–Objektes Zeilen einer Matrix, die „fehlende“ (s. Abschnitt 6.2.1.2.1) Werte enthalten, entfernt werden.

6.2.1.2.7 Reader

`Reader`–Objekte lesen einheitlich formatierte Information aus einer Infodatei und stellen sie anderen Modulen zur Verfügung. Entsprechend der eingelesenen Information werden `TimeSeries`–Objekte erzeugt und als Instanzvariablen gespeichert. Beim Bedarf kann auch eine simple Vollständigkeitsanalyse der Zeiteinstellungen durchgeführt werden. Die nicht verstandenen sowie die auskommentierten Zeilen der Infodatei werden ignoriert. Das Format der Infodatei ist ähnlich zum üblichen Format der *template*–Dateien des Prognosesystem, hat aber dennoch einige Unterschiede, die das Einlesen und die Interpretation der Information für C++ vereinfachen. Bei weiterer Beschreibung von C++–Modulen werden solche Dateien „Infodateien (im `Reader`–Format)“ genannt.

6.2.1.3 Programme

Alle implementierten Programme sind in der Regel kurz, weil sie meistens nur die folgenden Aufgaben erfüllen sollen:

- das Einlesen der Kommandozeile,
- Instanzierungen benötigter Objekte und Arbeitsverteilung zwischen denen;
- Abfrage der Ergebnisse ihrer Berechnungen und
- formatierte Ausgabe dieser Ergebnisse.

Die Formatierung der Ausgabe erfolgt üblicherweise mit Hilfe der Klasse `outFile`, die unter anderem vier Formate (mit optionalen zusätzlichen Einstellungen) für die Ausgabe der Matrizen zur Verfügung stellt.

Alle Programme brauchen eine umfangreiche Eingabeinformation. Deswegen bekommen sie als ein der Kommandozeileargumente den Namen einer Infodatei, die die meisten Eingabeparameter enthält und von einem `Reader`–Objekt gelesen wird.

Üblicherweise werden kompilierte Module aus den dazugehörigen Perl–Skripten (sog. Adaptern) aufgerufen (s. Abschnitt 6.2.2), sie können aber auch unmittelbar aufgerufen werden. Dafür braucht man eine Datei im „`Reader`–Format“, die entweder vom Hand oder aus einem *template*[40] mit dem Perl–Skript `convertForReader.pl` (s. Abschnitt 6.2.2.3) erstellt werden kann.

Die genauen Spezifikationen der Module befinden sich in der Anlage B des Anhangs.

6.2.1.3.1 matcorrelation

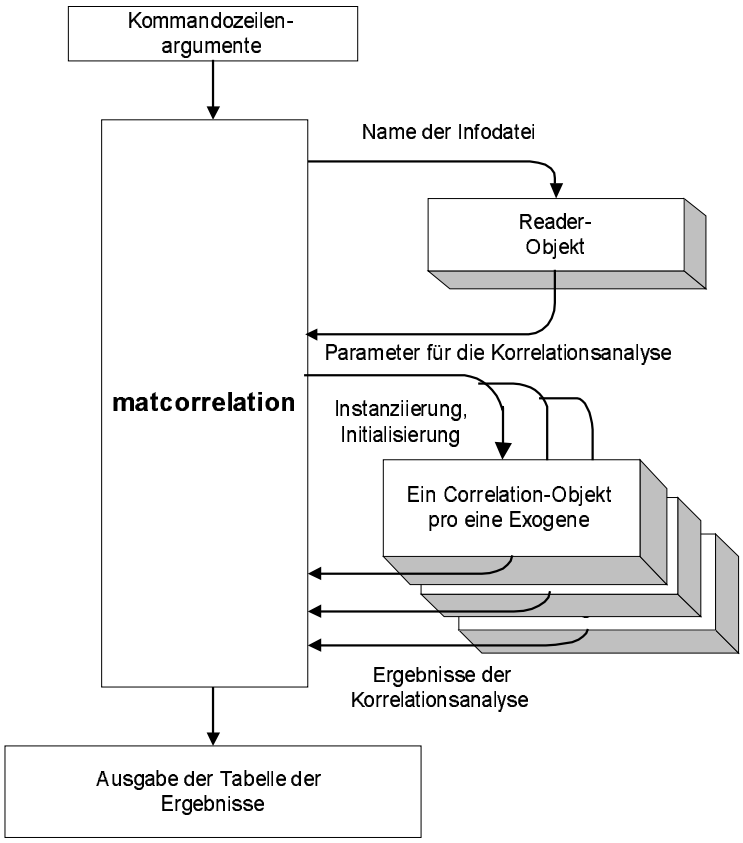


Abbildung 23: Architektur von `matcorrelation`

Als Eingabeparameter bekommt das Programm unter anderem die Endogene, Exogene und minimale/maximale betrachtete Grenzen der Verschiebungen jeder Exogenen k und K. Jede Exogene wird um k, $k+1$ usw. bis K Zeiteinheiten nach rechts (in die Zukunft) verschoben und der Korrelationskoeffizient zwischen ihr und der Endogenen wird jeweils berechnet. Die optimale Verschiebung jeder Exogenen (bei der die stärkste Korrelation beobachtet wird) wird nach *Bravias–Pearson*[41] klassifiziert und mit dem dazugehörigen

[40] Eine Infodatei im üblichen Format des Prognosesystems der Universität Osnabrück.

[41] Siehe Seite 87.

Korrelationskoeffizient ausgegeben. Falls die Option „`-v`“ angegeben wird, wird zusätzlich pro Exogene eine nach Verschiebungen sortierte Tabelle aller Korrelationskeffizienten ausgegeben. Diese Ausgabe kann mit `plot_cor.pl` visualisiert werden.

Siehe auch `matcorrelation.pl` im Abschnitt 6.2.2.1.1

6.2.1.3.2 matcorlist

Die Funktionalität des Programms unterscheidet sich von der Funktionalität von `matcorrelation` nur im Ausgabeformat. In diesem Fall wird eine flache nach den Korrelationskoeffizienten sortierte Liste der Indikatoren erstellt. Diese Liste wird vom Programm `optimizeInputs.pl` benutzt.

Die Programmstruktur stimmt fast überein mit der Struktur von `matcorrelation` mit der Unterschied, daß hier nur ein `Correlation`–Objekt instanziiert wird, das später im Zyklus mit einer neuen Exogene initialisiert wird. Die berechneten Korrelationskoeffizienten werden gleich abgefragt und in einem `matrix`–Objekt gesammelt. Zum Schluß werden die Ergebnisse sortiert und ausgegeben.

6.2.1.3.3 matmodel

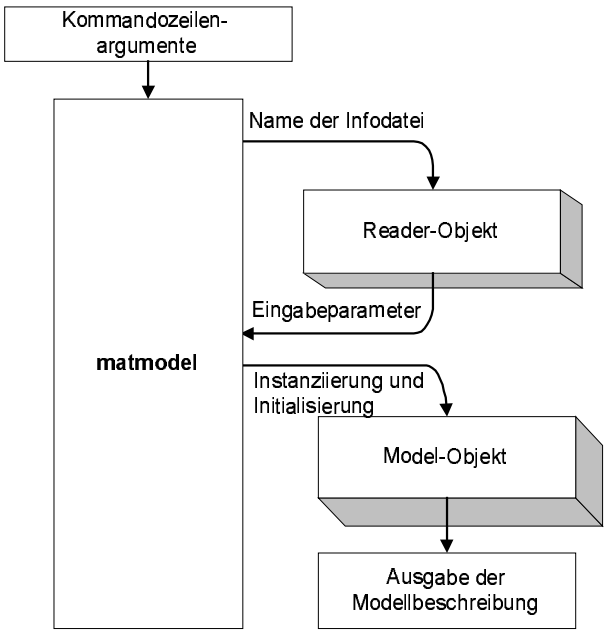


Abbildung 24: Architektur von `matmodel`

Mit Hilfe dieses Moduls wird ein lineares Modell erstellt. Aus einer großen Anzahl der Indikatoren werden die besten mit dem *forward–solution*–Algorithmus bestimmt, wobei auch Autoregression zulässig ist. Modellbeschreibung sowie Beurteilung des Modells und die Vorschläge über die optimale Anzahl der Exogenen und über die maximale

Prognoselänge werden ausgegeben. Genaueres über die Vorgehensweise bei der Modellerstellung siehe in der Beschreibung der Klasse `Model`, Abschnitt 6.2.1.2.4.

Siehe auch `matmodel.pl` im Abschnitt 6.2.2.1.2.

6.2.1.3.4 matregression

Mit diesem Modul wird eine Zeitreihe mit der Linearen Regression prognostiziert. Dabei werden berechnete Werte für den Prognosezeitraum und, optional, für den Trainingszeitraum sowie statistische Kennzahlen der Linearen Regression ausgegeben. Es werden für alle Regressoren Regressionskoeffizienten, Standardfehler, t–Werte und die entsprechenden Signifikanzniveaus berechnet.

Außerdem wird ein F–Test der Signifikanz des Modells durchgeführt, und die Ergebnisse werden ausgegeben.

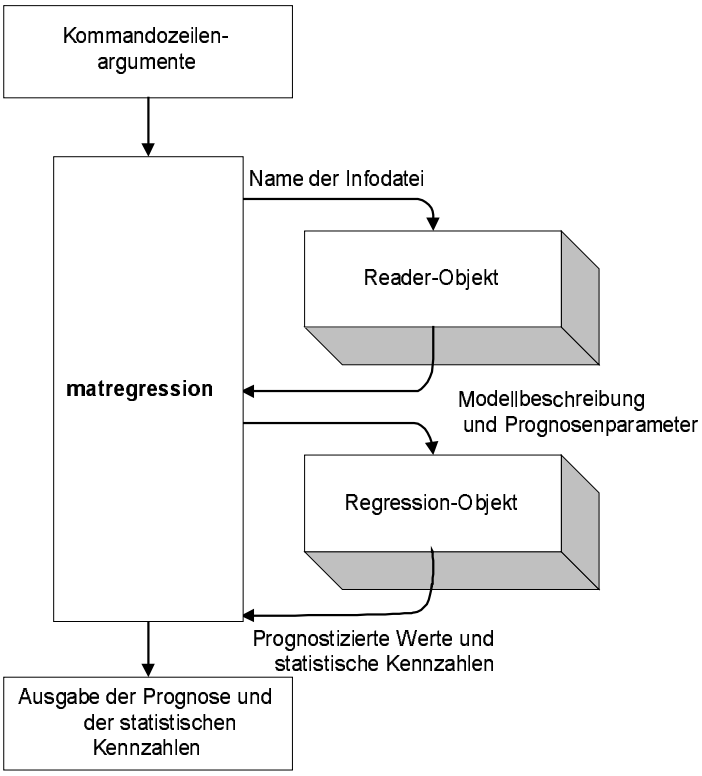


Abbildung 25: Architektur von `matregression`

Siehe auch `matregression.pl` im Abschnitt 6.2.2.1.3.

6.2.1.3.5 materror

Mit diesem Programm kann die *ex–post*–Analyse der Prognose durchgeführt werden. Verschiedene Prognosen können gleichzeitig bewertet werden. In diesem Fall werden nur die Werte der Prognose–Zeitreihen betrachtet, die zum gleichen Zeitsegment gehören, die übrigen Werte werden ignoriert. Die Ergebnisse werden in zwei unterschiedlich sortierten (nach MAPE bzw. nach RMSE[42]) Tabellen zusammengefaßt, um einen Vergleich der

unterschiedlichen Prognosemethoden zu erleichtern. Jede Zeile der Tabelle entspricht einer Vorhersage und jede Spalte einem Fehlermaß. Maximal 8 unterschiedliche Kennzahlen (7 Fehlermaße und *Theil*–scher Koeffizient[42]) können in einer formatierten Tabelle ausgegeben werden. Nach dem Wunsch können einige Kennzahlen ausgeblendet werden. Zusätzlich wird eine „naive" Prognose[43] generiert und ihre Bewertungen werden ausgegeben. In der letzten Tabellenzeile werden durchschnittliche Werte zusammengefaßt.

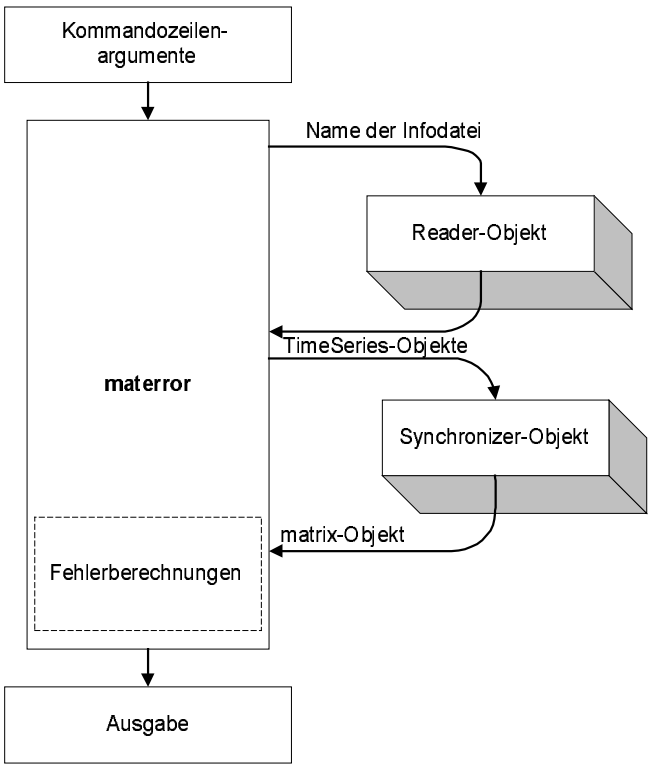


Abbildung 26: Architektur von `materror`

Das Programm benutzt massiv matrix–Objekte der C++–Bibliothek MatClass und von dieser Bibliothek unterstützte arithmetische und transzendente Operationen mit Matrizen sowie Ausgabeformatierungen. In Folge dessen ist das Programm kurz und effizient.

Siehe auch `materror.pl` im Abschnitt 6.2.2.1.4.

6.2.2 Perl

Perl–Programme lassen sich in drei Gruppen aufteilen: Adapter–, Optimierungs– und sonstige Module. Dem entsprechend werden sie in drei folgenden Abschnitten beschrieben.

[42] Alle Kennzahlen wurden im Abschnitt 4 auf der Seite 28 beschrieben.

[43] Siehe Seite 29.

6.2.2.1 Adapter

Die Perl–Module, deren Namen (ohne Erweiterungen) mit den Namen von C++–Programmen (siehe Abschnitt 0) übereinstimmen, kapseln jeweils ein C++–Modul und dienen als „Adapter“ zwischen ihm und dem übrigen Prognosesystem bzw. dem Benutzer. Die Architektur kann wie folgt dargestellt werden:

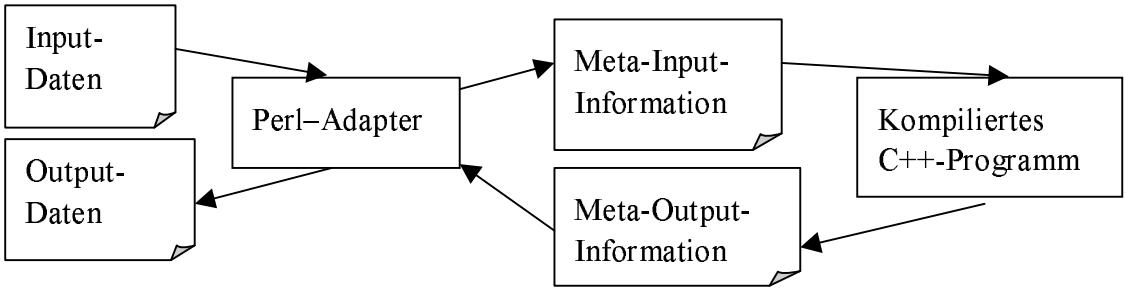


Abbildung 27: Architektur eines Perl–Adapters

Diese Lösung hat folgende Gründe:

- Perl ist eine mächtige Textverarbeitungssprache;
- Ein wesentlicher Teil des Universitätsprognosesystems wurde mit Perl geschrieben und kann somit benutzt werden. Besonders nützlich sind Perl–Funktionen, die Manipulationen mit Zeitreihen erlauben und Zeitberechnungen durchführen;
- Das Prognosesystem befindet sich in einer Entwicklungsphase und einige Details ändern sich, zum Beispiel im Format von Infodateien. Perl–Adapter lassen sich schnell und effizient anpassen, ohne C++–Programme umzuschreiben und neu zu kompilieren.

Da die Fuktionalität von „Adapter“–Modulen mit der Fuktionalität ihrer C++–Kernmodule fast übereinstimmt, verweise ich jeweils auf das Kapitel mit der Beschreibung des entsprechenden C++–Modul.

6.2.2.1.1 matcorrelation.pl

Siehe über `matcorrelation` im Abschnitt 6.2.1.3.1.

6.2.2.1.2 matmodel.pl

Siehe über `matmodel` im Abschnitt 6.2.1.3.3. Die Information über das gefundene Modell wird in das Format der Infodateien des Prognosesystems konvertiert, so daß mit dieser Datei direkt eine Prognose erstellt werden kann.

6.2.2.1.3 matregression.pl

Siehe über `matregression im` Abschnitt 6.2.1.3.4. Das Programm wandelt die simple numerische Ausgabe dieses Moduls in zwei Zeireihen im Sinne des Prognose–systems um. Eine Zeitreihe enthält prognostizierte Werte und die andere (sog. „`TRACE`“–

Zeitreihe) die mit Linearer regression berechneten Werte für den Trainingsraum. Diese Zeitreihen werden in das angegebene Unterprojekt des aktiven Projekts eingefügt.

6.2.2.1.4 materror.pl

Siehe `materror` im Abschnitt 6.2.1.3.4. Das Programm bietet zusätzliche Möglichkeiten bei der Ausgabeformatierungen. Zum Beispiel die Tabellen der Fehler, die nach MAPE bzw. RMSE sortiert werden, können je nach Einstellung sowohl in einer als auch in verschiedenen Dateien gespeichert werden. Auf Wunsch werden in Tabellen außer den Zeitreihennamen auch Kommentare eingetragen.

6.2.2.2 Optimierungsmodule

In diesem Abschnitt werden Implementierungen der Optimierungsalgorithmen beschrieben. Diese Algorithmen sind sehr dauerhaft, weil sie mit dem oft wiederholenden, zeitkritischen Lernen Neuronaler Netze verbunden sind. Deswegen ist bei diesen Programmen gemeinsam eine permanente Sicherung der gefundenen Information und die Nutzung dieser Information beim erneuten Start des Programms, damit kein Neuronales Netz unnötig zweimal trainiert wird. Um das zu erreichen wurden verwendet:

(1) eine eindeutige Kodierung der neuronalen Modelle;

(2) ein einheitlicher Mechanismus für Speicherung/Extrahieren von *hashtables* (*assotiative arrays*).

Nach der Berechnung jeder Kennzahl wird sie sofort in eine entsprechende *hashtable* eingetragen und in der dazugehörigen Datei gesichert. Beim neuen Start werden alle benötigten *hashtables* (eine Tabelle pro eine Kennzahl) aus den Dateien „ausgelesen“. Bei der Validierung eines Modells wird das Training nur dann durchgeführt, wenn der Eintrag für dieses Modell in der entsprechenden *hashtable* fehlt.

Ein weiteres spezifisches Detail dieser Module ist eine dynamische Erstellung der *template*–Dateien (die Einstellungen für das Training und Prognose enthalten) aus einzelnen *template*–Blöcken. Dadurch wird der Speicherplatz gespart, und die Information über vorhandene Dateien bleibt auch bei einer großen Anzahl der Läufe übersichtlich.

Außerdem automatisieren diese Programme solche sich wiederholenden Operationen wie:

- benötigte Skalierungen und Verschiebungen der Zeitreihen;
- Aufräumen von skalierten Zeitreihen, die nicht mehr benötigt werden;
- Aufräumen von erstellten Prognose–Zeitreihen nach ihrer *ex–post*–Bewertung;
- Zusammenfassung des Ablaufs des Prozesses in einer Tabelle;
- Speicherung vom *template* des besten Modells.

6.2.2.2.1 optimizeTop.pl und optimiseInput.pl

Diese zwei Module wurden als Teile eines gesamten Prozesses der sukzessiven Modelloptimierung gedacht (vgl. mit Abschnitt 5.3.2.2). Die Funktionalität der Programme entspricht ihren Namen. Während der Untersuchungsphase der Arbeit wurden sie manuell nacheinander gestartet, nur um Zwischenergebnisse besser analysieren zu können.

- **Gemeinsames**

Im Arbeitsverzeichnis muß die Datei "`komps`" vorhanden sein, die den *template*–Block mit folgenden Beschreibungen des Modells enthält: Deklarationen der Zeitreihen (der Exogenen und Endogenen) und Bestimmung der Validierungs– bzw. Prognosemenge. Als diese Datei kann die Ausgabe von `matmodel.pl` mit der Beschreibung des ermittelten linearen Modells benutzt werden.

```
MEASURE Wochen
PARAM VAL_PAIRS 10
PARAM METHOD SIM
### Modell-Vorschlag von matmodel: ###
OUTPUT $KOMP1 Umsatz Frankreich Land_Frankreich
USE 1 $KOMP1 Umsatz Mittel Farbe_mittel SHIFT 27
USE 1 $KOMP1 Exogena Konjunktur_ab_01.86 ZR2876020 SHIFT 29
USE 1 $KOMP1 Exogena Baufertigungen_ab_01.68 ZR0088061 SHIFT 28
USE 1 $KOMP1 Exogena UmsatzIndex_ab_01.91 ZR4654675 SHIFT 15
USE 1 $KOMP1 Exogena Konjunktur_ab_01.86 ZR2876134 SHIFT 25
USE 1 $KOMP1 Umsatz Dunkel Farbe_dunkel SHIFT 27
USE 1 $KOMP1 Umsatz Kiefer Material_Kiefer SHIFT 14
USE 1 $KOMP1 Exogena UmsatzIndex_ab_01.91 ZR4654120 SHIFT 20
USE 1 $KOMP1 Umsatz Kiefer Material_Kiefer SHIFT 28
USE 1 $KOMP1 Exogena UmsatzIndex_ab_01.91 ZR4654555 SHIFT 14
### Maximale Anzahl der Exogene: ca.  12 ###
### Maximale Prognose-Laenge: ca.     13 ###
```

Abbildung 28: Beispiel einer Datei mit der Modellbeschreibung („`komps`“)

Als das Optimierungskriterium kann der Testfehler des Netzes, MAPE oder RMSE, eingestellt werden. Bei jedem Schritt werden o. g. Kennzahlen und Koeffizient von Theil ausgegeben.

Beide Programme interagieren direkt mit mehreren Systemmodulen (`scale_for_snns.pl`, `clean_prog.pl`, `training.pl`, `materror.pl`, `convertForReader.pl` und `matcorlist`).

- **`optimizeTop.pl`**

Das Programm implementiert den im Abschnitt 5.3.2.2 beschriebenen Algorithmus. Als das Optimierungskriterium kann, wie bei `optimizeInput.pl`, Testfehler, MAPE oder RMSE verwendet werden.

Im Arbeitsverzeichnis des Programms müssen zusätzlich eine bis mehrere Dateien vorhanden sein, die jeweils einen *template*–Block mit Parametern für ein bestimmtes Neuronales Netz enthalten. Die Parameter HIDDEN und SHORTCUT sind irrelevant, weil sie in jedem Schritt ersetzt werden. Die Namen dieser Dateien müssen mit „BP.“ anfangen,

Die Suffixes sind nicht relevant und entsprechen üblicherweise einer bestimmten Netzarchitektur (z. B. „BP.Elman“, „BP.FF“, etc.).

```
**************** START at Sun Sep 13 22:10:34 MET DST 1998 **************
check shortcuts
use test_err for choosing
----------------- ------- --------- ----------- ----------- ------
Topologie         Hidden  TestError MAPE        RMSE
----------------- ------- --------- ----------- ----------- -------
BP.Elman          4       0.207411  25.68       5.615e+04   better
BP.Elman._sk      4       0.234999  35.44       8.517e+04
BP.FF             4       0.21969   32.79       7.394e+04
BP.FF._sk         4       0.218436  35.28       8.066e+04
BP.Jordan         4       0.216099  37.01       8.757e+04
```

Abbildung 29: Ausschnitt der Ausgabe von `optimizeTop.pl`

- **optimizeInput.pl**

Dieses Programm optimiert die Auswahl anhand des im Abschnitt 5.3.2.2 beschriebenen Algorithmus. Im Arbeitsverzeichnis müssen außer „komps“ die Dateien vorhanden sein:

- „BP.head“: Der *template*–Block mit Parametern des verwendeten Neuronalen Netzes (Lernparameter, Architektur).
- „ALL.template“ Datei im Systemformat mit Deklarationen aller Basiszeitreihen für die Erstellung der zu untersuchenden Indikatoren.
- „sense.block“ *template*–Block mit Parametern für die Sensitivitätsanalyse.

In der Kommandozeile werden minimale und maximale Verschiebungen für die Erstellung der Indikatoren aus der in „ALL.template“ beschriebenen Basiszeitreihen angegeben sowie Optimierungskriterium (MAPE, RMSE oder Testfehler).

Alle Indikatoren werden nach Korrelationskoeffizienten mit der Endogene angeordnet außer ersten, die aus der Datei "komps" genommen werden. Dadurch wird eine sortierte Indikatorenliste erstellt, die beim implementierten Algorithmus (s. Abbildung 14) benutzt wird.

```
**************** START at Thu Sep 24 16:20:53 MET DST 1998 **************
10 InputZeitreihen
1410 candidads
use test_err for choosing
---- --------------- --------------- ------- --------------- -------
Step  State                 testErr  MAPE    RMSE        Theil
---- --------------- --------------- ------ ---------- ------ -------
0    0:1:2:3:4:5:6:7:8:9   0.200136  28.39  5.934e+04  1.219
1    0:1:2:3:10:5:6:7:8:9  0.210433  27.59  5.984e+04  1.229
2    0:1:2:3:11:5:6:7:8:9  0.20726   29.41  6.204e+04  1.274
3    0:1:2:3:12:5:6:7:8:9  0.212616  30.51  6.626e+04  1.361
4    0:1:2:3:13:5:6:7:8:9  0.205147  28.4   5.77e+04   1.185
5    0:1:2:3:14:5:6:7:8:9  0.198057  25.81  5.525e+04  1.135    1% better
```

Abbildung 30: Ausschnitt der Ausgabe von `optimizeInput.pl`

6.2.2.2.2 searchNN.pl

Das Programm implementiert den im Abschnitt 5.3.2.3.3 beschriebenen Suchalgorithmus.

Im Arbeitsverzeichnis muß nur die Datei „`komps`“ vorhanden sein, die Beschreibungen der Endogenen, der potentiellen Exogenen und die Bestimmung der Validierungs– bzw. Prognosemenge enthält (siehe Abbildung 28). Die konstanten (nicht optimierten) Parameter Neuronaler Netze werden in den Dateien mit bestimmten Namen enthalten. Diese Dateien können, müssen aber nicht ins Arbeitsverzeichnis kopiert zu werden. Es reicht, den Pfad zu einem Verzeichnis anzugeben, wo sie zu finden sind. Die Namen dieser Dateien sind relevant: „`BP.FF`“ fürs *feedforward*–, „`BP.Elman`“ fürs Elman– und „`BP.Jordan`“ fürs Jordan–Netz. In demselben Verzeichnis muß auch die Datei „`BP.sense`“ mit den Einstellungen für die Sensitivitätsanalyse vorhanden sein. Ob bei der Generierung der Nachfolger nur direkte oder auch indirekte Nachbarn berücksichtigt werden, kann mit Kommandozeilenoptionen eingestellt werden.

Als Optimierungskriterium kann zur Zeit nur der Testfehler des Netzes benutzt werden.

Der Vorgang wird in der Datei „`output`“ protokolliert. Außerdem wird der aktuelle Pfad in der Datei „`path`“ gesichert.

Über alle Eingabeparameter siehe auch Anlage C des Anhangs.

6.2.2.2.3 selectModelNN.pl

Das Programm implementiert den im Abschnitt 5.3.2.3.4 beschriebenen Selektionsalgorithmus. Welche der drei verschiedenen Netztopologien (*feedforward*–, Elman– und Jordan–Netze) an der Selektion teilnehmen, kann eingestellt werden. Unter anderem kann auch die Anzahl der zu selektierenden Eltern angegeben werden. Voreingestellt ist (nach der Empfehlung [Bra97]), daß eine Population aus 40 Individuen besteht und in jedem Evolutionszyklus eine Viertel davon (10 Individuen) als Eltern gewählt werden.

6.2.2.3 *Sonstige Module*

6.2.2.3.1 convertForReader.pl

Das Programm konvertiert eine Infodatei im Format des Prognosesystems in eine Datei im `Reader`–Format (s. Abschnitt 6.2.1.2.7) und wird hauptsächlich von Adapter–Modulen benutzt.

6.2.2.3.2 aggregate.pl

Mit diesem Programm können Zeitreihen des Prognosesystems über neue Zeiteinheiten (von „täglich“ bis zu „jährlich“) aggregiert bzw. aufgeteilt werden. Als Aggregierungsfunktion kann entweder Summieren oder Mittelwertbildung ausgewählt werden. Die resultierende Zeitreihe wird in das Projekt hinzugefügt, wo sich die

ursprüngliche Zeitreihe befindet. Das Programm benutzt die im Prognosesystem bereits vorhandenen Funktionen für Zeitumwandlungen.

6.2.2.3.3 plot_cor.pl

Das Modul visualisiert die Ausgabe vom Programm `matcorrelation.pl` (das mit der Option „-v“ aufgerufen wurde). Dabei können mehrere Fenster erstellt werden. Jedes Fenster enthält eine Graphik mit einer einstellbaren Anzahl der Kurven. Eine Kurve zeigt die Veränderung der (absoluten) Korrelation einer Exogenen mit der Endogene in Abhängigkeit von der Verschiebung dieser Exogenen.

6.2.2.3.4 scale_for_snns.pl

Das Programm bekommt als Kommandozeilenargument den Namen des *template* mit Deklarationen der Inputzeitreihen und automatisiert eine notwendige Vorverarbeitung dieser Zeitreihen. Für die Vorverarbeitung der einzelnen Zeitreihen (Skalierung, Normierung, Verschiebung) wird das bereits vorhandene Programm des Prognosesystems `scale_single.pl` benutzt, das die eigentlichen Berechnungen durchführt und eine neue Zeitreihe erstellt. Die Vorverarbeitung einer Zeitreihe wird nur dann durchgeführt, wenn die resultierende Zeitreihe nicht bereits vorhanden ist.

Außerdem wird das *template* für die Verwendung mit Neuronalen Netzen konvertiert. Das ist notwendig aus den im Abschnitt 6.1.3 beschriebenen Gründen.

Das Modul wird von allen Optimierungsmodulen benutzt.

7 Empirische Untersuchungen

Im nächsten Unterkapitel wird die Vorbereitung für empirische Untersuchungen beschrieben sowie Merkmale der Datenbasis. Danach folgen Beschreibungen der eigentlichen Untersuchungen. Abbildung 31 stellt den Ablauf der Versuche schematisch dar. Jeder numerierte Block dieses Schemas entspricht einer Serie der entsprechenden Experimente mit allen neun Umsatzzeitreihen als Endogenen. In folgenden Unterkapiteln werden diese Versuche genau beschrieben, und ihre Ergebnisse sowie einige Statistiken zu ihren Verläufen werden tabellarisch und graphisch angegeben. Während der Versuche wurde bei der Erstelllung der Indikatoren die minimale Verschiebung immer gleich der Größe des Validierungszeitraums gewählt, um bei der Prognoseerstellung für diesen Zeitraum nur die bereits bekannten Werte zu benutzen.

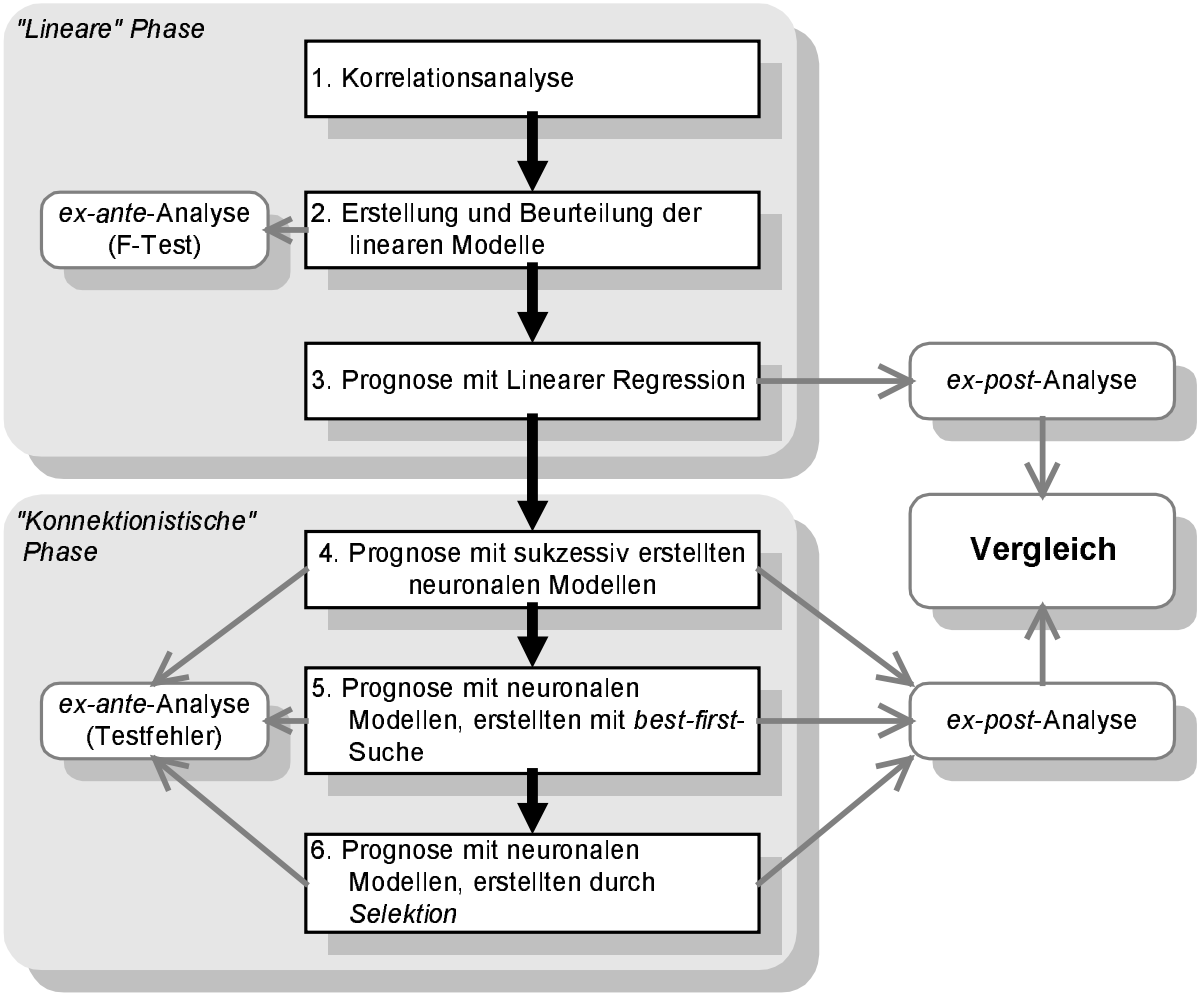


Abbildung 31: Schema der Untersuchungen

7.1 Empirische Datenbasis

7.1.1 Prognostizierte Zeitreihen

Als empirische Datenbasis dienten 9 Umsatzzeitreihen aus der Datenbank eines Küchenmöbelherstellers. Diese Zeitreihen lassen sich in drei Gruppen einteilen:

- Nach Export in verschiedene Länder: **USA**, **Dänemark**, **Frankreich**, **Österreich**
- Nach Material: **Buche**, **Kiefer**
- Nach verschiedenen Farbtönen: **hell**, **mittel**, **dunkel**

Der beobachtete Zeitraum ist von 1.Woche 1994 bis zur 40. Woche 1997. Jede Zeitreihe ist wöchentlich erfaßt und enthält ca. 200 Werte. In der Abbildung 32 ist eine typische Zeitreihe gezeigt (weitere Graphiken befinden sich in der Anlage D des Anhangs). Wie man sieht, sind für diese Umsatzdaten starke Schwankungen und die Existenz von Ausreißern sowie eine niedrige Quantität typisch. Diese Merkmale erschweren Prognoseaufgaben (insbesondere mit Neuronalen Netzen). Dennoch stellen diese Umsatzzeitreihen ein typisches Beispiel der betriebswirtschaftlichen Daten dar und sind dadurch als empirische Datenbasis interessant.

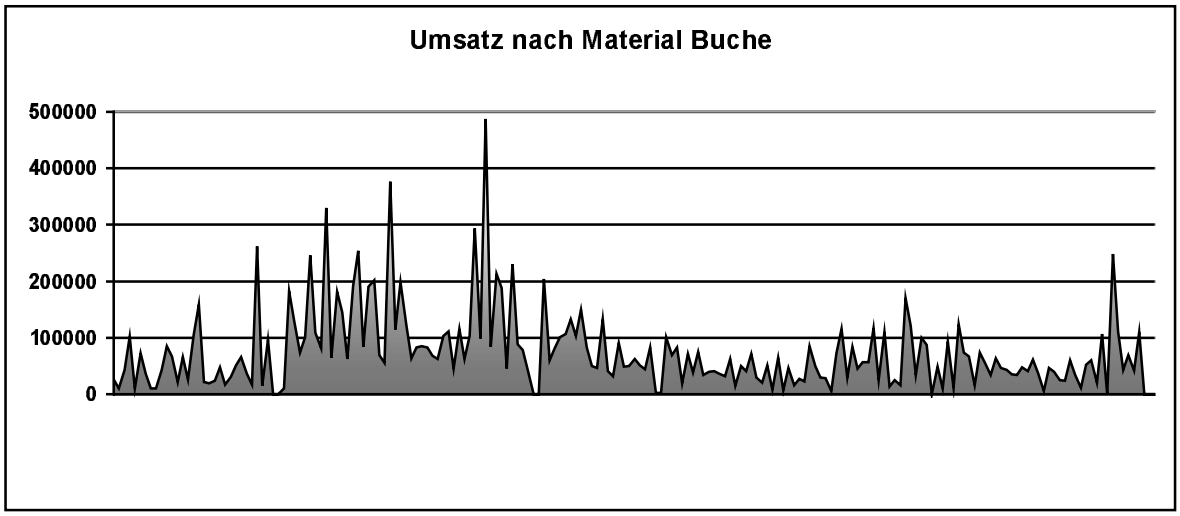


Abbildung 32: Verlauf der Kurve des Umsatzes nach Material Buche

7.1.2 Exogene Zeitreihen

Die o. g. beschriebenen Umsatzzeitreihen dienten während der Modellerstellung bei Prognoseversuchen sowohl als Endogenen als auch als potentielle Exogenen. Außerdem wurden 64 statistische Zeitreihen aus den verwandten Wirtschaftsbereichen — die den Umsatz eventuell beeinflussen könnten — ausschließlich als Exogenen benutzt. Die allgemeinen wirtschaftlichen Zeitreihen werden vom Deutschen Statistischen Bundesamt gepflegt[44]. Jede Zeitreihe besitzt außer dem Namen eine 7–stellige eindeutige Identifikationsnummer (Id.), wobei die ersten vier Ziffern dem Themenbereich (*Segment*) entsprechen und letzten drei Ziffern ihren eindeutigen Index innerhalb des Segmentes darstellen. Diese externen wirtschaftlichen Kennzahlen werden thematisch folgendermaßen angeordnet:

Tabelle 2: Exogene Zeitreihen

Segmentcode	Segmentbeschreibung	Anzahl der Zeitreihen
0088	Statistik der Baufertigstellungen ab 1952	12
2876	Statistik der Wirtschaftsrechnungen ausgewählter privater Haushalte ab 1950	9
1023	Statistik der Arbeitslosigkeit ab 1960	3
0266	Statistik der Bausparkassen ab 1951	6
1025	Statistik der saisonbereinigten Wirtschaftszahlen der Deutschen Bundesbank ab 1960 (Löhne)	3
4654	Statistik der Indizes des Auftragseingangs und des Umsatzes des verarbeitenden Gewerbes ab 1977	19
1531	Statistik des Außenhandels ab 1950	2
2651	Statistik vom Preisindex für den Wareneingang des produzierenden Gewerbes ab 1985	10

Eine genauere Aufzählung und Beschreibung dieser Zeitreihen befinden sich in der Anlage D des Anhangs.

[44] Die Daten sind unter der Internet-Adresse www.statistik-bund.de erhältlich.

7.1.3 Periodizität

Ein Problem lag darin, daß die exogenen Zeitreihen monatlich erfaßt werden, während die

Umsatzzeitreihen wöchentlich. Dadurch stellte sich die Frage über die Auswahl Monate oder Wochen als Zeiteinheiten, die bei Modellerstellungen benutzt werden. Entsprechend dieser Entscheidung mußten dann entweder Umsatz– oder statistische Zeitreihen neu aggregiert werden. Optimal wäre es die Umsatzzeitreihen über Monate zu aggregieren, um sie dadurch zu glätten. Leider kam dieses wegen der niedrigen Quantität der Umsatzdaten nicht in Frage. Deswegen wurden die wirtschaftlichen Kennzahlen in Wochen aufgeteilt, wodurch sich die Qualität der Exogenen erniedrigte. Für die erneute Aggregierung der Zeitreihen wurde das Programm `aggregate.pl` benutzt.

7.1.4 Konvertierung und Organisation der Daten

Die zu untersuchenden Daten lagen anfänglich in unterschiedlichen Formaten for:

- Alle Umsatzdaten waren in einer Datei im `Excel`–Exportformat (`*.xls`) enthalten.
- Bei Exogenen wurde die Menge aller Zeitreihen einer „Sendung“ des Statistischen Bundesamtes mit dem Beispielsnamen „`timeseries`“ mit zwei Dateien „`timeseries.AST`“ und “`timeseries.ASW`“ repräsentiert. Dabei enthielt die erste Datei Beschreibungen der Zeitreihen und die zweite ihre Werte.

Als erstes wurde jede einzelne Zeitreihe extrahiert und in einer Datei in einem simplen, einheitlichen Format (eine Zeile – ein Wert, # – Kommentar) gespeichert. Dafür wurden einige wiederverwendbaren Perl–Programme erstellt. Danach wurden die Daten als Projekt „BCAS“ und in das Prognosesystem der Universität integriert (siehe Tabelle 3).

Tabelle 3: Hierarchie des Projektes BCAS

Unterprojekt der 1. Ebene	Unterprojekt der 2. Ebene	Zeitreihen
Umsatz	Frankreich	`Wochen.Land_Frankreich`
	Dänemark	`Wochen.Land_Daenemark`
	Österreich	`Wochen.Land_ Oesterreich`
	USA	`Wochen.Land_USA`
	Dunkel	`Wochen.Farbe_Dunkel`
	Hell	`Wochen.Farbe_Hell`
	Mittel	`Wochen.Farbe_Mittel`
	Kiefer	`Wochen.Material_Buche`
	Buche	`Wochen.Material_Kiefer`
Exogena	Arbeitslosigkeit	3 Zeitreihen
	Baufertigungen	12 Zeitreihen
	Import-Export	2 Zeitreihen
	Konjunktur	9 Zeitreihen
	Kredit	6 Zeitreihen
	Löhne	3 Zeitreihen
	Umsatzindex	19 Zeitreihen
	Wareneingang-Index	10 Zeitreihen

Dafür wurde seine logische Struktur und notwendige Einstellungen in einer festgelegten Form beschrieben, und aus den rohen Daten wurden Zeitreihen im Format des Prognosesystems erstellt. Das wurde mit implementierten Perl–Skripten durchgeführt, wobei bereits vorhandene Systemfunktionen für Zeitreihen– und Projektverwaltung, Zeitberechnungen u. a. massiv benutzt wurden.

Nach dieser Vorbereitungsphase konnte die Funktionalität des erweiterten[45] Prognosesystem für weitere Untersuchungen vollständig benutzt werden.

7.2 Einsatz der Linearen Regression

In diesem Abschnitt werden Blöcke 1 bis 3 des Schemas in Abbildung 31 beschrieben.

7.2.1 Korrelationsanalyse

Für die Korrelationsanalyse wurde `matcorrelation.pl` und für die Visualisierung der Ergebnisse `plot_cor.pl` benutzt (siehe Abschnitte 6.2.1.3.1 und 6.2.2.3.3).

Eingabeparameter für die Korrelationsanalyse:

- **Endogene:** eine der Umsatzzeitreihen
- **Exogene Basiszeitreihen:** Statistische – und Umsatz– Zeitreihen, insg. 73 Zeitreihen
- **Minimale Verschiebung:** 10 Wochen
- **Maximale Verschiebung:** 29 Wochen
- **Validierungszeitraum:** letzte 10 Wochen

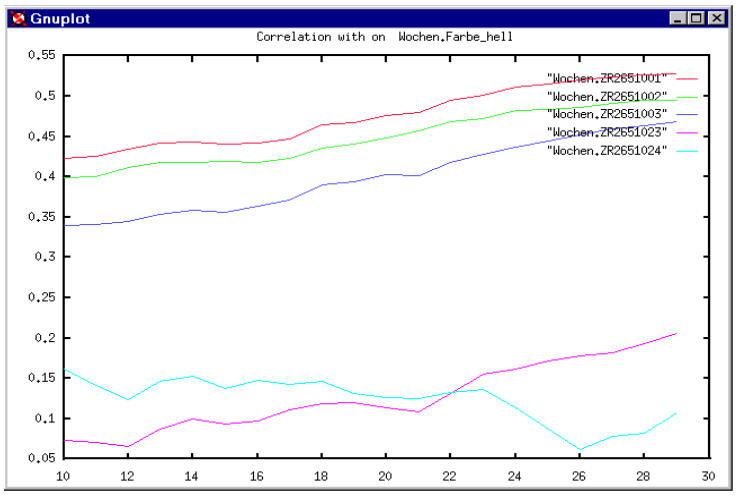


Abbildung 33: Visualisierung der Korrelation mit `plot_cor.pl`

[45] Über die vorgenommenen Erweiterungen siehe Abschnitt 6.2.

Insgesamt wurden 730 Indikatoren erstellt und analysiert. Eine Beispielgraphik des Zusammenhangs zwischen der Verschiebung einiger Indikatoren und ihrer Korrelation mit der Endogene ist in Abbildung 33 dargestellt. Ausgewählte Ergebnisse der Korrelationsanalyse sind in Tabelle 5 zusammengefaßt.

Tabelle 4: Ergebnisse der Korrelationsanalyse

Endogene	Identifikationsnummer der besten Exogene*	Verschiebung	Korrelations-koeffizient	Klassifikation nach *Bravias–Pearson*
Buche	2651087	24	0.4035	ok
Dänemark	2651002	11	0.3401	schlecht
Dunkel	2876136	29	0.3224	schlecht
Frankreich	2651003	12	0.4386	ok
Hell	1025003	10	0.5317	ok
Kiefer	2651080	24	0.4122	ok
Mittel	2651080	24	0.2421	schlecht
Österreich	2651001	25	0.4721	ok
USA	2876133	26	0.3719	schlecht
Durchschnittlich			**0.3927**	**schlecht**

7.2.2 Erstellung der linearen Modelle

In diesem Schritt wurde mit `matmodel.pl` (s. Abschnitt 6.2.1.3.2) ein lineares Modell für jede Endogene erstellt. `matmodel.pl` wurde jeweils zweimal benutzt. Beim ersten Lauf wurden Größe des Validierungszeitraums, Exogenenzahl und minimale Verschiebung der Exogenen (gleich der Größe des Validierungszeitraums) nicht angegeben, sondern dynamisch gefunden:

Eingabeparameter für `matmodel.pl`(1. Lauf):

- **Endogene:** eine der Umsatzzeitreihen;
- **Exogene Basiszeitreihen:** Statistische und Umsatz– Zeitreihen, insg. 73 Zeitreihen;
- **Verschiebungenzahl:** 20

Dem entsprechend wurden für jede Endogene 730 Indikatoren erstellt und analysiert.

Die Ausgabedateien von `matmodel.pl` enthalten unter anderem die folgenden Werte:

- maximal **12** Exogenen ins Modell aufzunehmen;
- maximal **13** Werte zu prognostizieren.

* Thematische Anordnung bzw. Namen der Zeitreihen siehe Abschnitt 7.1.2 und Anlage D.

Außerdem wurden die besten **12** Exogenen gefunden, so daß die in den Ausgabedateien beschriebene Modellen direkt für Prognoseerstellung benutzt werden konnten. Um aber die gefundenen Grenzwerte direkt nicht zu benutzen, wurde `matmodel.pl` für jede Endogene nochmal aufgerufen, diesmal mit den zusätzlichen Eingabeparametern:

- **maximale Exogenenzahl:** 10;
- **minimale Verschiebung:** 10;
- **Größe des Validierungszeitraums:** 10.

7.2.3 *ex–ante*–Beurteilung der linearen Modelle

Die Ausgabe von `matmodel.pl` bezüglich der *ex–ante*–Beurteilungen der Modelle sind in Tabelle 5 dargestellt.

Tabelle 5: ex–ante–Analyse der linearen Modelle

Endogene	Beurteilung der Regressoren im Durchschnitt			Beurteilung des gesamten Modells		
	St–.Fehler	*t*–Wert	*t*–Test (%) (α=0,05)	R^2	F–Wert	F-Test (%) (α=0,05)
Buche	6.77	2.78	98	0.5768	15.13	100
Dänemark	13.59	4.60	99	0.5675	16.14	100
Dunkel	0.48	2.97	97	0.5158	11.40	100
Frankreich	7.61	3.57	97	0.5073	11.43	100
Hell	3.32	4.31	99	0.7164	33.35	100
Kiefer	16.06	3.13	99	0.4222	9.64	100
Mittel	1.25	3.03	98	0.4583	9.56	100
Österreich	100.74	2.27	93	0.3986	9.68	100
USA	0.024	3.41	99	0.536	11.67	100
Durchschnittlich	**16.65**	**3.34**	**98**	**0.5221**	**14.22**	**100**

Es ist zu sehen, daß obwohl die Korrelation zwischen den Zeitreihen nach *Bravias–Pearson* nicht als „gut“ klassifiziert werden konnte, werden alle Modelle mit dem F–Test als signifikant beurteilt. F–Wert jedes Modells fiel zwar runter mit dem Wachstum der Freiheitsgrade des Modells, entsprach aber auch nach dem Einfügen der letzten Exogenen einem höhen (94% bis 100%) Signifikanzniveau des Modells.

7.2.4 *ex–post*–Beurteilung der linearen Modelle

Im Rahmen der Validierung von erstellten Modellen wurden mit jedem Modell Prognoswerte für seinen Validierungszeitraum generiert, und die Prognosegüte wurde *ex–post*–anlysiert. Die Ergebnisse sind in der Tabelle 6 aufgeführt.

Tabelle 6: ex–post–Analyse der linearen Modelle

Endogene	MAPE	RMSE	Theil'scher Koeffizient
Buche	49.54	4.850e+04	0.6489
Dänemark	73.2	6.981e+04	0.5528
Dunkel	102.9	7.015e+04	0.6906
Frankreich	30.99	6.793e+04	1.268
Hell	48.66	8.618e+04	0.7674
Kiefer	100.4	1.690e+05	0.5361
Mittel	164.1	7.695e+04	0.4034
Österreich	132.8	3.810e+05	0.8307
USA	156.9	2.846e+04	1.1
Durchschnittlich	**95.5**	**1.109e+05**	**0.7553**

Wie zu sehen ist, obwohl alle Modelle nach F–Test perfekt waren, ist der MAPE bei fast der Hälfte aller Prognosen größer als 100 %. Der *Theil*'sche Koeffizient zeigt, daß bei der Prognose von zwei Zeitreihen („Frankreich" und „Hell") die mit Linearer Regression erstellte Prognose schlechter als die „naive" war. Analyse der visueller Information zeigte, daß „normale" Werte akzeptabel prognostiziert wurden und daß eine Verschlechterung von Ergebnissen der *ex–post*–Analyse wegen der im Prognoseraum vorhandenen Ausreißer (häufig Nullwerte) entsteht. Diese seltenen und nicht typischen Werte können mit Linearer Regression nicht vorhergesagt werden:

7.3 Einsatz Neuronaler Netze

Als nächstes wurden Neuronale Netze für die Prognose der Endogenen eingesetzt und mit der Linearen Regression verglichen. Der Schwerpunkt lag in der Untersuchung beim Problem der Optimierung der neuronalen Modelle. Im Rahmen dieser Untersuchungen wurden drei verschiedene Optimierungsmethoden verwendet, und die Ergebnisse wurden verglichen. Als nicht veränderte Lernparameter der Neuronalen Netze dienten folgende Einstellungen:

- **Allgemeine Lern–, Ausgabe– und Netzeinstellungen**
  - Alle Netztypen

```
#----------allgemeine Daten------------
PARAM EPOCHEN    1500  # Anzahl der Epochen
PARAM ERR        -7    # Fehlerwerte alle N Epochen
PARAM MINEPOCHEN 100   # Mindestanzahl der Epochen
PARAM MAXERROR   0.2   #Abbruchkriterium
#----------Netztopologie---------------
PARAM ACTIVE     3  # Aktive Schichten im Netzwerk
PARAM OUTPUT     1  # Knoten in der Ausgabeschicht
PARAM ACT H1:Act_TanH I:Act_Identity O:Act_Identity
```

- **Reihenfolge von Mustern**
  - *feedforward*-Netz

```
PARAM SHUFFLE   TRUE    # Reihenfolge irrelevant
PARAM RANDOMIZE ON      # Testpaare zufällig
```

  - Jordan- und Elman-Netze

```
PARAM RANDOMIZE OFF     # Testpaare am Ende des
                        #         Trainigsraums
PARAM SHUFFLE   FALSE   # Reihenfolge relevant
PARAM INCONSIST LEAVE   # Widersprüche lassen
```

- **Lernparameter**
  - *feedforward*-Netz

```
PARAM LEARNREM   Verfahren, ETA, IMP, FLATSPOT, DMAX
PARAM LEARNFUNC  BackpropMomentum
PARAM LEARNPARAM 0.1, 0.5, 0.1, 0.0
PARAM UPDATEFUNC Topological_Order
PARAM INITREM    Initialisierung, a, b
PARAM INITFUNC   Randomize_Weights
PARAM INITPARAM  0.05, -0.05
```

  - Jordan- und Elman-Netze

```
#--------------Elman - Jordan Werte fuer BP--------
PARAM ETA         0.3                # Lernrate
PARAM IMP         0.2                # Impulsrate
PARAM WEIGHT      0.1
PARAM LEARNING    ONLINE
#-------------------------------------------------
PARAM LEARNREM    Verfahren, ETA, IMP, FLATSPOT,
                   DMAX , teacher_forcing
PARAM LEARNFUNC   JE_BP_Momentum
PARAM LEARNPARAM  0.3, 0.2, 0.0, 0.0, 1.0
PARAM UPDATEFUNC  JE_Order
PARAM INITREM     Initialisierung, a, b,
                   linkweight_self_recurrent,
                   linkweight_to_context_units,
                   init_act_of_context_units
PARAM INITFUNC    JE_Weights
PARAM INITPARAM   0.01, -0.01 , 0.5, 1.0, 0.0
```

7.3.1 Sukzessive Optimierung

7.3.1.1 Optimierung der Netzarchitektur.

Für diesen Versuch wurde das Programm `optimizeTop.pl` (s. Abschnitt 6.2.2.2.1) benutzt. Als Outputzeitreihe diente jeweils eine der Umsatzzeitreihen, als Inputzeitreihen die mit *forward solution* gefundenen Exogenen. Als Optimierungskriterium wurde der Testfehler benutzt. Es wurden folgende Parameter der Neuronalen Netze variiert:

- **Topologie**: Jordan, Elman, *feedforward*
- **Anzahl von *hidden* Neuronen**: 4 bis 20
- ***shortcuts***: Mit, ohne

Durch die Betrachtung aller möglichen Kombinationen der variierenden Parameter ergaben sich 102 Trainingsläufe für jede Endogene. Die Ergebnisse werden in der Tabelle 7 zusammengefaßt.

Der Prozeß der Optimierung der Netztopologie dauerte für jede Zeitreihe ca. 1,5 – 2 Stunden. Es ist zu sehen, daß partiell rekurrente Netze bei den fünf aus neun Endogenen den *feedforward*–Netzen überlegen waren.

Tabelle 7: Sukzessive Erstellung neuronaler Modelle, Zwischenergebnisse

Endogene	Netztopologie	*hidden*	Testfehler	MAPE (%)	RMSE	Theil'scher Koeffizient
Buche	Jordan	16	0.137317	67.19	5.635e+04	0.7539
Dänemark	Elman	8	0.174938	89.13	7.824e+04	0.6196
Dunkel	FF*	13	0.162201	94.54	6.849e+04	0.6743
Frankreich	FF	12	0.164184	31.12	6.773e+04	1.265
Hell	Jordan	17	0.144797	86.65	8.419e+04	0.7496
Kiefer	Elman, *shortcuts*	16	0.205045	92.15	1.787e+05	0.5668
Mittel	FF, *shortcuts*	14	0.184559	186.1	7.867e+04	0.4125
Österreich	FF, *shortcuts*	8	0.202162	76.17	3.036e+05	0.662
USA	Jordan	14	0.162526	87.4	2.391e+04	0.9246
Durchschnittlich			**0.17086**	**90.05**	**1.044e+05**	**0.746**

7.3.1.2 Optimierung der Auswahl von Inputzeitreihen

In diesem Schritt wurden neuronale Modelle mit `optimizeInput.pl` (s. Abschnitt 6.2.2.2.1) durch das Variieren der Inputzeitreihen weiter optimiert. Als Optimierungskriterium diente der Testfehler. Für die Erstellung der zu untersuchenden Indikatoren wurden alle 71 vorhandenen Zeitreihen benutzt, die jeweils zwischen 10 und 29 Zeiteinheiten verschoben wurden. Dies ergab 1420 Indikatoren. Diese Indikatoren – außer der ersten aus dem linearen Modell übernommenen Indikatoren – wurden bzgl. ihrer Korrelation mit der Endogenen angeordnet. Die Anzahl der Exogenen im Modell wurde konstant gleich 10 gehalten. Daraus ergaben sich 1410 Trainingsläufe. Einige Statistiken werden in der Tabelle 8 dargestellt.

* (*feedforward*-Netz).

Tabelle 8: Statistiken zur Optimierung des Netzinput

Endogene	Schritt der letzten Verbesserung	Anzahl der Verbesserungen	Um wieviel % wird Testfehler verbessert
Buche	200	17	14
Dänemark	135	17	8
Dunkel	30	2	2
Frankreich	326	2	11
Hell	343	9	21
Kiefer	757	15	17
Mittel	1268	5	18
Österreich	597	8	8
USA	123	2	6
Durchschnittlich	**384**	**8**	**11.7**

Interessant war, daß ca. zwei Drittel der alten Exogenen überlebten und mehr als drei Viertel der augewählten Zeitreihen unter den ersten **100** der **1410** (nach der Korrelation mit der Endogene sortierten) Indikatoren lagen; ihr Korrelationskoeffizient war im Durchschnitt über **0,3**.

Es ist deutlich zu sehen, daß die Modelle bezüglich RMSE und des Theil'schen Koeffizienten im Durchschnitt trotz dem riesigen Rechneraufwand sogar schlechter geworden sind. Diese Beobachtung gerechtfertigt die Frage, inwieweit die sukzessive Erstellung der Neuronalen Modelle sinnvoll ist.

Tabelle 9: Sukzessive Erstellung neuronaler Modelle: Testfehler und ex–post–Analyse

Endogene	**Netzarchitektur**	**Testfehler**	**MAPE (%)**	**RMSE**	**Theil'scher Koeffizient**
Buche	Jordan:10:16:1	0.123707	118.1	7.616e+04	1.019
Dänemark	Elman:10:8:1	0.176213	69.94	8.733e+04	0.6915
Dunkel	FF:SC:10:13:1	0.173369	92.35	7.23e+04	0.7118
Frankreich	FF:10:12:1	0.164784	35.86	7.441e+04	1.464
Hell	Jordan:10:17:1	0.13683	61.53	7.189e+04	0.640
Kiefer	Elman:10:16:1	0.199901	103.6	2.874e+05	0.8649
Mittel	FF:SC:10:14:1	0.151689	87.13	6.887e+04	0.3611
Österreich	FF:SC 10:8:1	0.185198	165.5	4.622e+05	1.008
USA	Jordan 10:14:1	0.148204	65.81	1.544e+04	0.597
Durchschnittlich		**0,16221**	**88.87**	**1,35e+05**	**0.817**

7.3.2 Optimierung mit der „*best–first*“–Suche

Während dieser Serie der Experimente wurden neuronale Modelle mit dem im Programm `selectModelNN.pl` (s. Abschnitt 0) implementierten Selektionsalgorithmus optimiert.

Als Eingabeparameter dienten:

- **Testmenge:** 30 %;
- **Toleranz:** 0.005
- **Netztopologien:** Elman, Jordan und *feedforward* je mit und ohne *schortcuts*
- **Anzahl der versteckten Neuronen:** von 4 bis 16
- **Anzahl von Inputzeitreihen:** von 4 bis 10
- **potentielle Inputzeitreihen:** die 10 mit *forward solution* gefundenen Exogenen
- **Validierungsmenge:** letzte 10 Beobachtungswerte
- **Maximale Anzahl der „ineffizienten“ Schritte:** 20

Tabelle 10: Neuronale Modellbildung mit der „best–first"–Suche: Testfehler und ex–post–Analyse

Endogene	Umgebung	Netzarhitektur	Testfehler	MAPE (%)	RMSE	Theil'scher Koeffizient
Buche	breit	Elman:6:4:1	0.119399	60.27	7.13e+04	0.9539
	eng	Elman:SC:4:4:1	0.126807	42.22	5.926e+04	0.7928
Dänemark	breit	FF: SC:7:18:1	0.143493	50.42	6.832e+04	0.541
	eng	FF: SC:4:5:1	0.177077	74.62	1.206e+05	0.9546
Dunkel	breit	FF:10:13:1	0.162201	94.54	6.849e+04	0.6743
	eng	FF: SC:4:5:1	0.18093	83.42	8.069e+04	0.7944
Frankreich	breit	FF:10:12:1	0.164184	33	7.077e+04	1.393
	eng	FF:SC:10:5:1	0.185445	31.94	9.6e+04	1.889
Hell	breit	FF:6:10:1	0.120776	60.99	7.574e+04	0.6744
	eng	FF:5:5:1	0.120849	62.4	9.218e+04	0.8208
Kiefer	breit	FF:5:4:1	0.178314	101.7	2.198e+05	0.6614
	eng	FF:5:4:1	0.178314	101.7	2.198e+05	0.6614
Mittel	breit	FF:SC:10:10:1	0.165564	260.3	1.081e+05	0.5669
	eng	FF:SC:10:4:1	0.178309	173.3	7.813e+04	0.4096
Österreich	breit	FF: SC:6:13:1	0.180927	84.82	3.243e+05	0.7073
	eng	FF: SC:5:10:1	0.185215	137.7	3.719e+05	0.811
USA	breit	FF:6:11:1	0.141223	130.6	2.515e+04	0.9724
	eng	FF:6:5:1	0.144886	82.19	1.675e+04	0.6478
Mittelwert	**breite Umgebung**		**0.15289**	**97.4**	**1.146+05**	**0.793**
	enge Umgebung		**0.16420**	**87.7**	**1.261+05**	**0.865**

Im Rahmen der Optimierung des neuronalen Modells mit „*best–first*"–Suche wurden zwei Umgebungskonzepte getestet[46]: Als die Umgebung eines Knotens wurde einmal die Menge seiner direkten Nachbarn („enge" Umgebung) und einmal die Menge seinen direkten und indirekten Nachbarn („breite" Umgebung) betrachtet. Beim Vergleich zeigte es sich, daß durch die Generierung der Nachfolger aus der engen Umgebung eines Knotens (s. Abbildung 15 und Abbildung 17) das Verfahren viel effizienter wird, dabei aber erniedrigt sich die durchschnittliche Qualität der Modelle etwas (s. Tabelle 11). Bei weiteren Vergleichen wurden die in Während dieser Serie der Experimente wurden neuronale Modelle mit dem im Programm `selectModelNN.pl` (s. Abschnitt 0) implementierten Selektionsalgorithmus optimiert.

[46] Unter einer *Umgebung* eines Knotens wird die Menge seiner Nachfolger verstanden. Genauer siehe Abschnitt 5.3.2.3.1.

Als Eingabeparameter dienten:

- **Testmenge:** 30 %;
- **Toleranz:** 0.005
- **Netztopologien:** Elman, Jordan und *feedforward* je mit und ohne *schortcuts*
- **Anzahl der versteckten Neuronen:** von 4 bis 16
- **Anzahl von Inputzeitreihen:** von 4 bis 10
- **potentielle Inputzeitreihen:** die 10 mit *forward solution* gefundenen Exogenen
- **Validierungsmenge:** letzte 10 Beobachtungswerte
- **Maximale Anzahl der „ineffizienten" Schritte:** 20

Tabelle 10 dargestellten Ergebnisse von „*best–first*"–Suche bei der Betrachtung der „breiten" Umgebung benutzt.

Tabelle 11: Vergleich der „best–first"–Suche bei verschiedenen Umgebungskonzepten

Durchschnittlich für alle Modelle		**Enge Umgebung**	**?**	**Breite Umgebung**
Aufwand	Pfadlänge ohne *backtracks*	28	<	25
	Pfadlänge mit *backtracks*	30	<	29
	Anzahl von Trainings	84	<	188
Qualität	MAPE	87.7	<	97.4
	Testfehler	0.16420	>	0.15289
	RMSE	1.261+05	>	1.146+05
	Theil–scher Koeffizient	0.865	>	0.793

7.3.3 Optimierung durch Selektion

Während dieser Serie der Experimente wurden neuronale Modelle mit dem im Programm `selectModelNN.pl` (s. Abschnitt 0) implementierten Selektionsalgorithmus optimiert.

Als Eingabeparameter dienten:

- **Variation der Testmenge:** von 26 % bis 30 %;
- **Toleranz:** 0.005
- **Netztopologien:** Elman, Jordan und *feedforward* je mit und ohne *schortcuts*
- **Anzahl der versteckten Neuronen:** von 4 bis 16
- **Anzahl von Inputzeitreihen:** von 4 bis 10
- **potentielle Inputzeitreihen:** die 10 mit *forward solution* gefundenen Exogenen
- **Validierungsmenge:** letzte 10 Beobachtungswerte
- **Anzahl der Eltern in jeder Generation:** 10
- **Maximale Anzahl der „ineffizienten" Schritte:** 5

Tabelle 12: Selektionsablauf bei verschiedenen Endogenen

Endogene	Anzahl der Generationen	Anzahl von Trainings
Buche	15	1606
Dänemark	17	1635
Dunkel	10	1617
Frankreich	24	1973
Hell	13	1273
Kiefer	27	2204
Mittel	15	1252
Österreich	23	1628
USA	11	1161

In der oberen Tabelle (Tabelle 12) werden einige Statistiken zu den Selektionsprozessen bei verschiedenen Endogenen angegeben.

Bei allen Endogenen wurde Selektion wegen des Überschrittens der maximalen Anzahl von ineffizienten Schritten (Schritte, während deren sich das Modell nicht *wesentlich* verbessert).

Die Verbesserung der Fitneß der einzelnen Individuen in einer Generation sowie die durchschnittliche Fitneß der Generationen bei nacheinander folgenden Evolutionszyklen können mit `plot_selection.pl` visualisiert werden. Beurteilung der besten mit der Selektion erstellten Modelle wird in der Tabelle 13 angezeigt.

Tabelle 13: Neuronale Modellbildung durch Selektion: Testfehler und ex–post–Analyse

Endogene	Netzarchitektur	Testfehler	MAPE (%)	RMSE	Theil'scher Koeffizient
Buche	Jordan:5:13:1	0.120943	77.64	6.968e+04	0.8796
Dänemark	FF:7:5:1	0.142942	54.59	1.117e+05	0.8842
Dunkel	FF:6:13:1	0.157406	139.9	9.066e+04	0.8925
Frankreich	FF:7:6:1	0.166012	29.46	6.652e+04	1.309
Hell	Elman:6:5:1	0.123808	62.53	9.33e+04	0.8308
Kiefer	FF:SC:5:10:1	0.183696	117.9	2.455e+05	0.7786
Mittel	FF: SC:5:14:1	0.157434	144	8.58e+04	0.4499
Österreich	FF:SC:6:11:1	0.177009	102.8	3.623e+05	0.7901
USA	FF:SC:6:6:1	0.139725	125.8	2.588e+04	1.001
Durchschnittlich		**0.1521**	**87.3**	**1.171e+05**	**0.803**

7.4 Vergleichsanalyse

Als erstes muß gesagt werden, daß die Ergebnisse der Experimente wegen einer niedrigen Datenqualität einerseits eine schlechte Prognosequalität verursachen und andererseits nicht hundertprozentig aussagefächig für den Vergleich von verschiedenen Ansätzen sein können. Dennoch zeigen sie im Kontext der untersuchte Datenbasis die Relationen zwischen sowohl verschiedenen Prognosemethoden als auch zwischen den verwendeten Methoden der konnektionistischen Modellbildung.

In Tabelle 14 werden verschiedene Verfahren der konnektionistischen Modellbildung verglichen. Die Durchschnittswerte aus dieser Tabelle werden auch in Abbildung 34 graphisch dargestellt.

Tabelle 14: Vergleich von Erstellungsmethoden konnektionistischer Modelle: MAPE, RMSE, Theil–scher Koeffizient

Endogene	Erstellungs-methode	Netz	Test-fehler	MAPE	RMSE	Theil
Buche	sukzessiv	Jordan:10:16:1	0.123707	118.1	7.616e+04	1.019
	Suche	Elman:6:4:1	0.119399	60.27	7.13e+04	0.9539
	Selection	Jordan:5:13	0.120943	77.64	6.968e+04	0.8796
Dänemark	sukzessiv	Elman:10:8:1	0.176213	69.94	8.733e+04	0.6915
	Suche	FF: SC:7:18:1	0.143493	50.42	6.832e+04	0.541
	Selektion	FF:7:5	0.142942	54.59	1.117e+05	0.8842
Dunkel	sukzessiv	FF:SC:10:13:1	0.173369	92.35	7.23e+04	0.7118
	Suche	FF:10:13:1	0.162201	94.54	6.849e+04	0.6743
	Selektion	FF:5:11:1	0.157406	139.9	9.066e+04	0.8925
Frankreich	sukzessiv	FF:10:12:1	0.164784	35.86	7.441e+04	1.464
	Suche	FF:10:12:1	0.164184	33	7.077e+04	1.393
	Selektion	FF:7:6:1	0.166012	29.46	6.652e+04	1.309
Hell	sukzessiv	Jordan:10:17:1	0.13683	61.53	7.189e+04	0.640
	Suche	FF:6:10:1	0.120776	60.99	7.574e+04	0.6744
	Selektion	Elman:6:5:1	0.125921	62.53	9.33e+04	0.8308
Kiefer	sukzessiv	Elman:10:6:1	0.199901	103.6	2.874e+05	0.8649
	Suche	FF:5:4:1	0.178314	101.7	2.198e+05	0.6614
	Selektion	FF:SC:5:10:1	0.180754	117.9	2.455e+05	0.7786
Mittel	sukzessiv	FF:SC:10:14:1	0.151689	87.13	6.887e+04	0.3611
	Suche	FF:SC:10:10:1	0.165564	260.3	1.081e+05	0.5669
	Selektion	FF: SC:5:14:1	0.157434	144	8.58e+04	0.4499
Österreich	sukzessiv	FF:10:8:1	0.185198	165.5	4.622e+05	1.008
	Suche	FF:6:13:1	0.180927	84.82	3.243e+05	0.7073
	Selektion	FF:SC:6:11:1	0.177009	102.8	3.623e+05	0.7901
USA	sukzessiv	Jordan:10:14:1	0.148204	65.81	1.544e+04	0.597
	Suche	FF:6:11:1	0.141223	130.6	2.515e+04	0.9724
	Selektion	FF:SC:6:6:1	0.156727	125.8	2.588e+04	1.001
Mittel-wert	**sukzessiv**		**0.1622**	**88.87**	**1,35e+05**	**0.817**
	Suche		**0.1529**	**97.4**	**1.146+05**	**0.793**
	Selektion		**0.1521**	**87.3**	**1.171e+05**	**0.803**

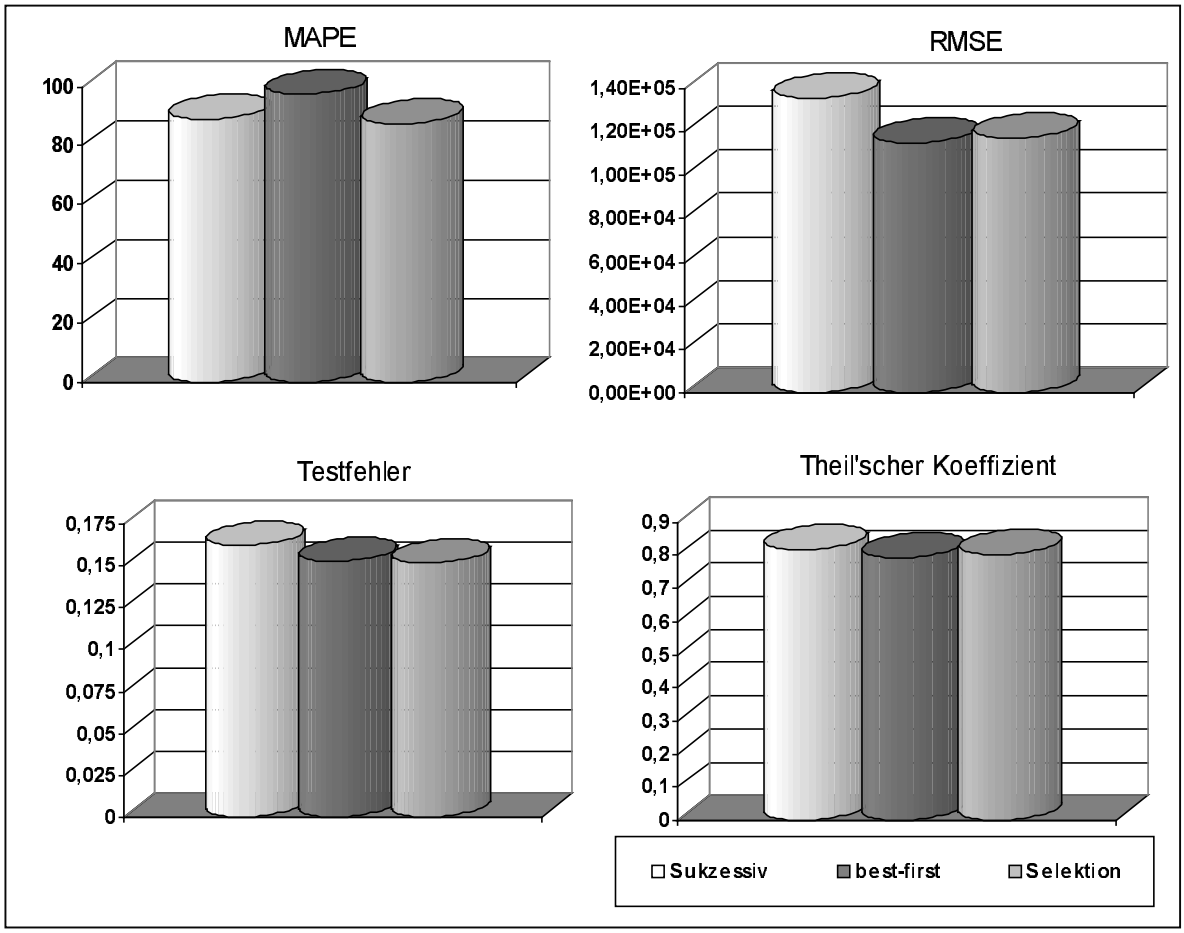


Abbildung 34: Vergleich von Methoden neuronaler Modellbildung (Durchschnittswerte)

Bezüglich des durchschnittlichen Testfehlers, RMSE und des *Theil*–schen Koeffizientes sind die mit Methoden der Künstlichen Intelligenz erstellten Modelle im Durchschnitt besser als die sukzessiv erstellten, obwohl bei der sukzessiven Erstellung eine zusätzliche Information berücksichtigt war: Es wurden alle 1410 Indikatoren betrachtet, während bei der KI–Optimierungsmethoden nur 10.

Es ist auch zu sehen, daß durchschnittlicher Testfehler, MAPE und *Theil*–scher Koeffizient der mit der Selektion erstellten Modelle kleiner sind als bei den mit *best–first*–Suche erstellten Modellen.

Die Senkung des Testfehlers bei der Verwendung der Selektion im Unterschied zu der „best-first"-Suche ist vielversprechend, weil dabei die Testmenge variiiert wurde, um die Gefahr von *overfitting* zu verringern.

Tabelle 15: Vergleich von Neuronalen Netzen und Linearer Regression als Prognosemethoden

Endogene	Prognose–methode	MAPE	RMSE	Theil
Buche	Naiv	52.83	7.475e+04	1
	Lineare Regression	49,54	4,850e+04	0,6489
	NN* (Jordan:5:13:1)	77.64	6.968e+04	0.8796
Dänemark	Naiv	100	1,263e+05	1
	Lineare Regression	73,2	6,981e+04	0,5528
	NN (FF:7:5:1)	54.59	1.117e+05	0.8842
Dunkel	Naiv	65,11	1,016e+05	1
	Lineare Regression	102,9	7,015e+04	0,6906
	NN (FF:6:13:1)	139.9	9.066e+04	0.8925
Frankreich	Naiv	23.5	5.081e+04	1
	Lineare Regression	30,99	6,793e+04	1,268
	NN (FF:7:6:1)	29.46	6.652e+04	1.309
Hell	Naiv	59,11	1,123e+05	1
	Lineare Regression	48,66	8,618e+04	0,7674
	NN (Elman:6:5:1)	62.53	9.33e+04	0.8308
Kiefer	Naiv	76.8	3.323e+05	1
	Lineare Regression	100,4	1,690e+05	0,5361
	NN (FF:SC:5:10:1)	117.9	2.455e+05	0.7786
Mittel	Naiv	72,75	1,907e+05	1
	Lineare Regression	164,1	7,695e+04	0,4034
	NN (FF:4:7)	144	8.58e+04	0.4499
Österreich	Naiv	72,36	4,586e+05	1
	Lineare Regression	132,8	3,810e+05	0,8307
	NN (FF:SC:6:8)	102.8	3.623e+05	0.7901
USA	Naiv	100	2,586e+04	1
	Lineare Regression	156,9	2,846e+04	1,1
	NN (Elman:7:17)	166,3	2,107e+04	0,8147
Mittelwert	Naiv	**69.16**	**1.64e+05**	**1**
	Lineare Regression	**95.5**	**1.11e+05**	**0.7553**
	Neuronale Netze	**87.3**	**1.171e+05**	**0.803**

In der Tabelle 15 sowie in Abbildung 35 werden die Ergebnisse der *ex–post*–Analyse der verschiedenen Prognosemethoden zusammengefaßt. Bei der Prognose mit Neuronalen Netzen werden die mit der Selektion erstellten konnektionistische Modelle benutzt. Es ist zu sehen, daß Neuronale Netze im Durchschnitt bessere Ergebnisse als Lineare Regression liefern. Der durchschnittliche MAPE der naiven Prognose ist fast zweimal kleiner als der MAPE der mit anderen Methoden erstellten Prognosen. Das indiziert den chaotischen Charakter und niedrige Prognostizierbarkeit der Zeitreihen.

* Neuronale Netze.

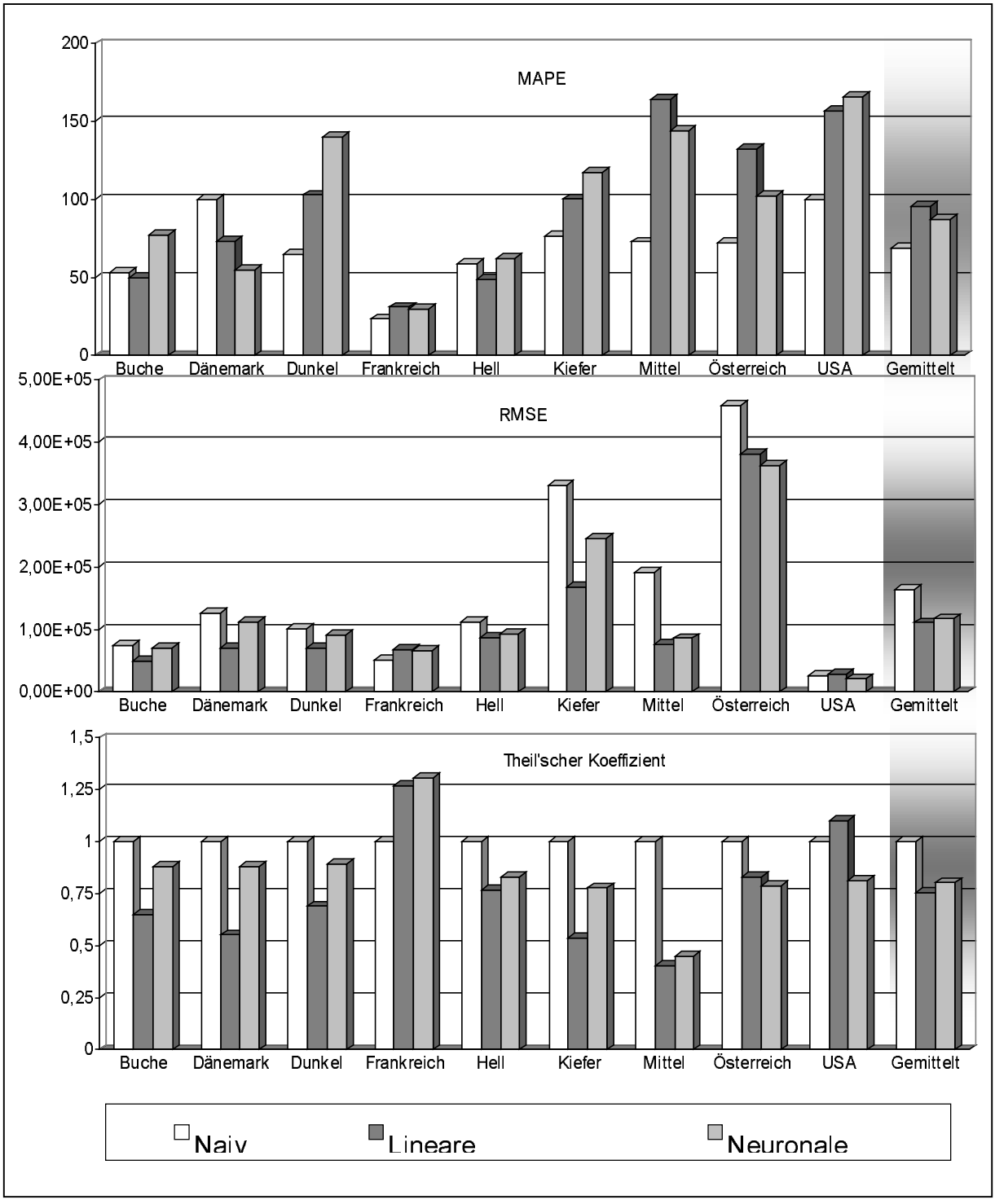


Abbildung 35: Vergleich der Prognosemethoden: Neuronale Netze, Lineare Regression und „naiv“.

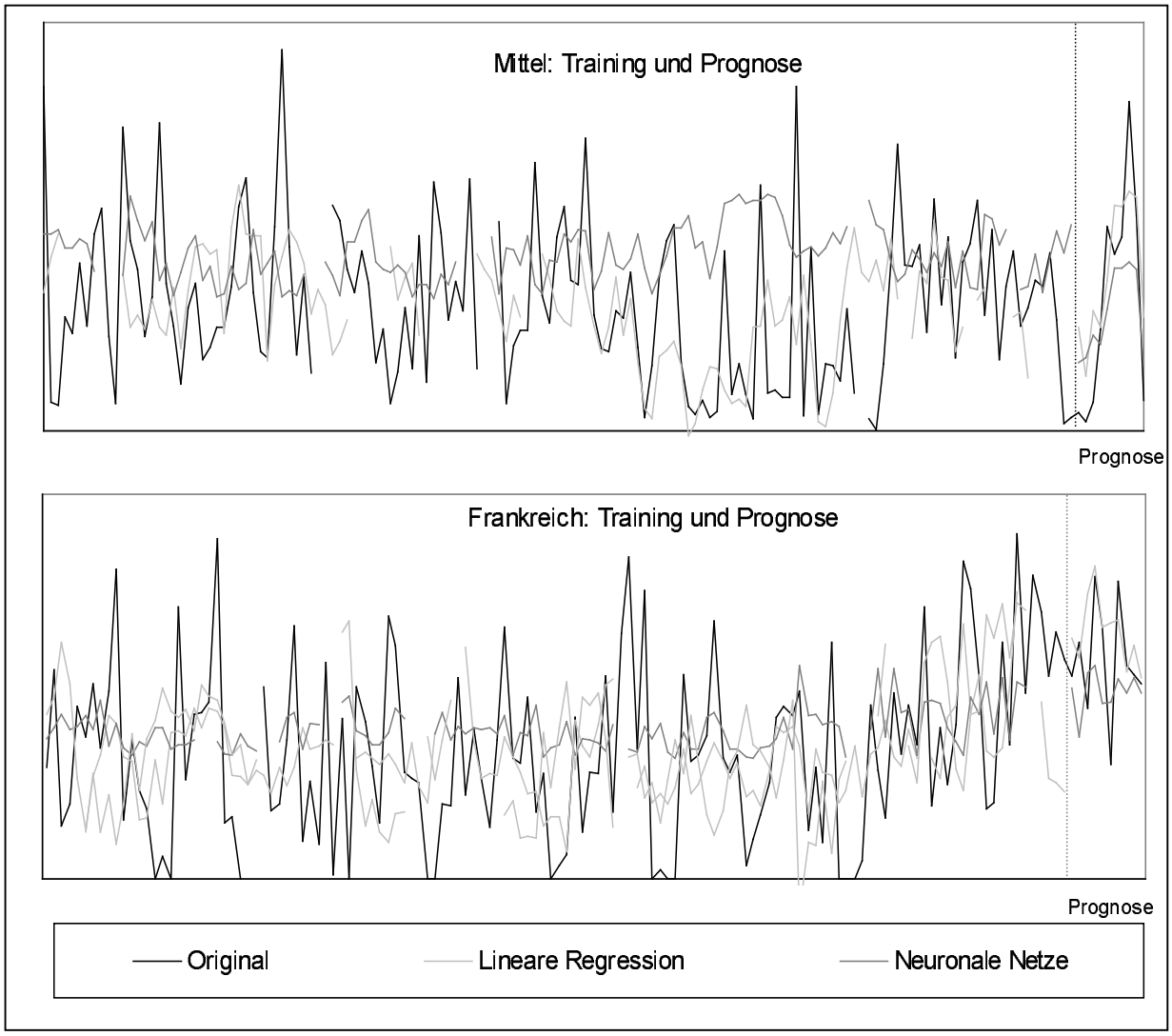


Abbildung 36: Training und Prognose von „Mittel“ und „Frankreich“

In der Abbildung 36 wird zwei Beispiele von Training und Prognose mit Linearer Regression und Neuronalen Netzen angezeigt. Das erste Beispiel „Mittel“ ist bzgl. MAPE der Prognose bei beiden Prognoseverfahren sehr schlecht (s. Tabelle 15). Das zweite Beispiel „Frankreich“ ist im Gegenteil anhand des MAPE akzeptabel. Man sieht, daß in beiden Fällen die Schwankungen im Prognosebereich nicht vorhergesagt werden können, obwohl der generelle Ablauf der Kurve „Mittel“ bei beiden Prognosenmethoden deutlich erkennbar ist. Es ist auch zu sehen, daß die mit Neuronalen Netzen generierte Prognose den Ablauf der ursprünglichen Zeitreihe „Frankreich“ besser als Lineare Regression wiederholt. Die zu sehenden Ausreißer können aber mit Neuronalen Netzen nicht vorhergesagt werden.

8 Zusammenfassung und Ausblick

In diesem Schlußkapitel werden entsprechend der Problemstellung und aufgrund der durchgeführten Experimente folgende Fragen beantwortet:

(1) Prognostizierbarkeit der Umsatzzeitreihen;

(2) Vergleich von Linearer Regression und Neuronalen Netzen als Prognosemethoden;

(3) Konnektionistische Modellierung mit den Methoden der Künstlichen Intelligenz;

8.1 Daten– vs. Prognosequalität

Alle untersuchten wöchentlichen Umsatzzeitreihen sind schlecht prognostizierbar. Der Grund liegt an der niedrigen Quantität und Qualität der Daten. Die Zeitreihen sind chaotisch und enthalten viele Ausreißer und starke Schwankungen, die keine der verwendeten Prognosemethoden erfolgreich abbilden kann: Bei 5 von 9 Zeitreihen lag der MAPE weit über 100% und die *ex–post*–Analyse zeigte, daß die Qualität der mit Linearer Regression oder mit Neuronalen Netzen erstellten Prognosen vergleichbar mit der Qualität der naiven Prognose war. Aufgrund dessen soll die Frage über die Prognostizierbarkeit der vorliegenden Zeitreihen im allgemein mit „nein“ beantwortet werden.

Eventuell können die Daten durch verschiedene Transformationen und spezielle Behandlungen von Ausreißer sowie durch die Aggregation über größere Zeiteinheiten verbessert werden.

8.2 Vergleich Neuronaler Netze und Linearer Regression

Wie es bereits am Anfang des Kapitels 7 bemerkt wurde, sind die Ergebnisse des Vergleiches von Neuronalen Netzen und Linearer Regression als Prognosemethoden wegen einer niedrigen Datenqualität nur im Kontext dieser Daten oder höchstens bei Daten der vergleichbaren Qualität aussagefähig und sollen keineswegs als allgemeingültige betrachtet werden.

Die durchschnittliche Prognosequalität bei der Anwendung Neuronaler Netze ist bzgl. aller Fehlermaße besser als bei der Anwendung Linearer Regression. Beim Vergleich der Qualität von Prognosen der einzelnen Zeitreihen kommt raus, daß Lineare Regression bei 4 aus 9 Zeitreihen besser als Neuronale Netze ist. Dabei ist der Aufwand bei der Verwendung von Neuronalen Netzen unvergleichbar höher.

Interessant, daß die untersuchten Zeitreihen entweder mit beiden oder mit keiner der Prognosemethoden vorhersagbar waren. Nur in einem Fall (bei „hell“) war Lineare Regression schlechter, während Neuronale Netze besser als die „naive“ Prognosemethode. Das bedeutet, daß falls Lineare Regression schlechter als die „naive“ Methode ist, sind die

Chancen, eine zufriedenstellende Prognose mit Neuronalen Netzen zu generieren, eher niedrig. Nicht ausgeschlossen ist, daß dieses ähnliche Verhalten von Neuronalen Netzen und Linearer Regression durch die Auswahl des linearen Modells als Ausgangspunkt bei der konnektionistischen Modellbildung bedingt ist.

Es sei bemerkt, daß beim Vergleich der Methoden die mit der *Selektion* erstellten konnektionistischen Modelle betrachtet wurden, und bei der Modellbildung mit *Selektion* wurde als die Menge der potentiellen Einflußgrößen aus Effizienzgründen ausschließlich die Exogenen der entsprechenden linearen Modelle benutzt (s. Abschnitt 7.3.3). Eventuell hätte die Untersuchung einer größeren Menge der externen Zeitreihen bessere Ergebnisse erzielt.

8.3 KI Methoden als Instrument der konnektionistischen Modellbildung

In der Literatur über Anwendungen Neuronaler Netze wird das Problem der Netzoptimierung oft behandelt. Meistens kann dabei folgendes bemerkt werden:

- Inputdaten und dadurch die entsprechende Anzahl von Inputneuronen sind immer fest vorgegeben, wobei nicht klar ist, warum insbesondere diese Daten benutzt wurden;
- Es wird immer nur eine Teiloptimierung durchgeführt, wie z. B. Optimierung der Lernrate oder Ausdünnung der Gewichte/Neuronen (sog. *pruning*).

Es fehlt im Prinzip die Betrachtung eines konnektionistischen Modells als komplexes, aus mehreren Teilen bestehendes Objektes. Es können z. B. solche Komponenten eines neuronalen Modells genannt werden:

-Inputdaten;

-Einstellungen des Neuronalen Netzes:

- Topologie (z. B. *feedforward*, Elman, Jordan etc.)
- Lernparameter;
- Anzahl von Inputneuronen;
- Anzahl von versteckten Neuronen.

Da es anzunehmen ist, daß die o. g. Modellkomponenten zusammenhängen, ist es sinnvoll, diese Parameter nicht wie üblich sukzessiv, sondern im gleichen Optimierungsprozess einzustellen. Weil einerseits keine analytischen Verfahren dafür eingesetzt werden können, und andererseits eine zu optimierende Größe (die der Qualität des Modells entspricht, z. B. Testfehler) gewählt werden kann, entspricht dieses Optimierungsproblem dem Problem der heuristischen Suche im großen mehrdimensionalen Suchraum – einem üblichen Bereich der Anwendung von Methoden der Künstlichen Intelligenz.

Diese Überlegungen sowie die Ergebnisse der durchgeführten Experimente motivieren den Einsatz von Methoden der Künstlichen Intelligenz, insbesondere von Suchverfahren und Evolutionären Algorithmen (vs. Selektion) für die Lösung des o. g. Problem.

In folgenden Unterkapiteln werden einige Überlegungen über die Bildung der heuristischen Funktion und über Selektion kurz erläutert.

Im Ganzen muß die Verwendung von Methoden der Künstlichen Intelligenz für neuronale Modellbildung der Gegenstand einer tieferen Untersuchung auf besseren Daten sein.

8.3.1 Heuristik

Im Rahmen der Arbeit wurde der Testfehler als heuristische Funktion benutzt. Dabei entsteht die Gefahr der Überanpassung des Modells an die Testmenge während der Optimierung. Dieses Problem läßt sich durch die Bildung einer komplexeren Heuristik lösen. Es kann z. B. eine zusätzliche Validierung des Modells nach jedem Training verwendet werden[47], deren Ergebnisse (z. B. MAPE oder *Theil*'scher Koeffizient) dann neben dem Testfehler in der Heuristik einfließen. Auch das Bevorzugen von kleineren Netzen kann in einer Form in die Heuristik mitberücksichtigt werden.

Ein andere Ansatz zur Verbesserung der Heuristik ist bei der *Selektion* verwendbar (und wurd im Rahmen der Arbeit benutzt): Dabei, um den *Overfitting*– Effekt zu vermeiden, wird, wie es bei [Bra97][48] empfohlen wird, bei jeder Generation eine neue Testmenge gewählt, und die Fitneß der Populationsmitglieder wird durch die Ermittlung ihrer durchschnittlichen Fitneß korrigiert[49].

8.3.2 Selektion

Entsprechend dem biologischen Vorbild ist es naheliegend, neuronale Modelle nach den Prinzipien der natürlichen Evolution (vs. Selektion) zu optimieren. Darüber hinaus, wie es in [Bra97] (Abschnitt 4.1.8) angegeben wird, stellen die meisten Suchalgorithmen (*Hillclimbing*, einige Varianten von *Simulated Annealing*, u. a.) triviale Fälle der Selektion dar (mit der aus einem oder aus zwei Individuen bestehenden Population).

Das motiviert die Verwendung der Selektion für die Optimierung der neuronalen Modelle eher als die Anwendung der Suchalgorithmen. Der Aufwand ist dabei, wie es sich während der Experimente zeigte, vergleichbar. Ein weiterer großer Vorteil der Selektion ist ihre Steuerung durch die Verwendung von verschiedenen Populationsmodellen. Im Rahmen der Arbeit wurde nur das einfachste Populationsmodell implementiert. Durch die Verwendung der komplizierteren Modelle sind bessere Ergebnisse zu erwarten. Z. B. bei der Selektion konnte beobachtet werden, wie schnell Modelle mit einer bestimmten Netztopologie die

---

[47] Dafür wird eine weitere Validierungsmenge aus der Datenmenge ausgewählt, die aus der Trainings- und Testmenge ausgeschloßen wird und nur für die Berechnung der Heuristik benutzt wird. Die erste Validierungsmenge dient wie früher für die anschließende Beurteilung des erstellten Modells.

[48] Siehe [Bra97], Abschnitt 4.1.4.3.

[49] Als Fitneß—Wert berechnet man den durchschnittlichen Generalisierungsfehler über die aktuelle und alle vorherige Test—Mengen.

anderen verdrängen (s. Abbildung 38 im Anhang). Wenn ein „Inselmodell“ verwendet wird, wobei sich Modelle mit verschiedenen Netztopologien auf lokalen „Inseln“ entwickeln, können Modelle mit verschiedenen Netztopologien ihre Entwicklungslinien ungestört fortsetzen und beim Schluß in einem globalen Wettbewerb teilnehmen. Außerdem auch die Evolutionsparameter selbst können Objekte der Evolution sein (s. [Bra97],Abschnitt 4.1.6).

Anhang

Anlage A: Klassen der Bibliothek „regression“

• Correlation

Konstruktor: `Correlation()`

Tabelle 16: Öffentliche Methoden der Klasse Correlation

1	Rückgabewert	Methodenname	Argumente	
2	void	assign	matrix String String int int	_pairs _endoName _exoName _startShift _endShift
3	void	changeExo	matrix String	newExo s
4	void	clear		
5	int	df_FTest_1		
6	int	df_FTest_2		
7	double	Fsign		
8	double	Ftest		
9	string	getEndoName		
10	string	getExoName		
11	double	getMax		
12	int	getMaxShift		
13	matrix	getModel		
14	double	getRforPair	TimeSeries& TimeSeries& int	endo exo val_pairs
15	matrix	getRs		
16	matrix	getRsForShifts		
17	void	init	TimeSeries& TimeSeries& int int int	endo exo _startShift cntShift val_pairs
18	void	process		
19	void	removeLastExo		

- **Model**

Konstruktor: `Model()`

Tabelle 17: Öffentliche Methoden der Klasse Model

	Rückgabewert	Methodenname	Argumente	
1	void	analyse	TimeSeries**	_tss
			int	count
			int	_nExos
			int	startShift
			int	shiftCnt
			int	val_pairs
2		clear		
3	int	maxExoCount		
4	int	maxProgLength		
5		printResults	String	fname
			int	verbose
6	Correlation*	results		

- **Prognose**

Konstruktor: `Prognose(String method)`

Tabelle 18: Öffentliche Methoden der Klasse Prognose

	Rückgabewert	Methodenname	Argumente	
1	matrix	predict	TimeSeries**	tss
			int	count
			long	start
			long	end
			Regression &	reg
2	matrix	predictReal	matrix	Y
			matrix	coeffs
			int	uCount
3	matrix	predictSingle	matrix	Y
			int	uCount
			Regression &	reg

- **Reader**

Konstruktor: `Reader(char * path)`

Tabelle 19: Öffentliche Methoden der Klasse Reader

	Rückgabewert	Methodenname	Argumente
1	void	checkTimes	
2	void	clear	
3	int	getAdvance	
4	String	getDecomp	
5	int	getExosCount	
6	long	getFirstProg	
7	long	getLastProg	
8	String	getMeasure	
9	String	getMethod	
10	int	getTrace	
11	Long	getTrainEnd	
12	Long	getTrainStart	
13	TimeSeries**	getTss	
14	Int	getTssCount	
15	Int	getValPairs	

- **Synchronizer**

Konstruktor: `Synchronizer();`

Tabelle 20: Öffentliche Methoden der Klasse Synchronizer

	Rückgabewert	Methodennamen	Argumente	
1	long	getCommonStart		
2	matrix	getSyncMatrix	TimeSeries **	tss
			int	cnt
3	matrix	getSyncMatrixWithNaiv	TimeSeries **	tss
			int	cnt
4	matrix	removeMissings	matrix	m
5	long	synchronize	TimeSeries **	tss
			int	cnt
			long	last_prog
6	matrix	synchronizePair	TimeSeries&	y
			TimeSeries&	x
			int	how

- **Regression**

Konstruktor: `Regression(String method)`

Tabelle 21: Öffentliche Methoden der Klasse Regression

	Rückgabewert	Methodenname	Argumente	
1	void	init	TimeSeries**	tss
			int	count
			long	end
2	void	apply	TimeSeries**	tss
			int	count
			long	end
			int	cur
3	void	decompose	TimeSeries**	tss
			int	count
			long	end
4	int	df_Ftest_1		
5	int	df_Ftest_2		
6	int	df_tTest		
7	matrix	evaluateCoeffs	matrix	onEndo
			matrix	onExos
8	double	FSign		
9	double	FTest		
10	matrix	getCoeffs		
11	matrix	getStdErr		
12	matrix	getTSign		
13	matrix	getTValues		
14	double	setTolerance	double	e
15	void	clear()		
zusätzliche Beurteilungen n-tes Regressanden				
16	double	dw	int	n
17	double	rBarSq	int	n
18	double	rsq	int	n
19	double	rss	int	n
20	double	se	int	n
21	double	tss	int	n
22	double	tTest	int	n

- **TimeSeries**

Konstruktor: `TimeSeries()`

Tabelle 22: Öffentliche Methoden der Klasse Timeseries

	Rückgabewert	Methodennamen	Argumente	
1	void	init	int int int int int string string	_type _count _shift _start _end _home _name
2	int	getCount		
3	long	getEnd		
4	string	getHome		
5	string	getName		
6	long	getRealEnd		
7	int	getCount		
8	long	getEnd		
9	string	getHome		
10	string	getName		
11	long	getRealStart		
12	int	getShift		
13	long	getStart		
14	int	getType		
15	double	getValue	long	timePoint
16	matrix	getValues		
17	int	length		
18	void	getName	String	s
19	void	setShift	int	_shift
20	void	synchronize	long long	up to
21	long	getRealEnd		

Anlage B: C++–Module

- **matcorlist**

```
usage: matcorlist shifts infoFile [outFile]
```

Argumente:

`shifts` die Anzahl der Verschiebungen der Exogenen;

`infoFile` der Name der Infodatei mit der Deklarationen von Zeitreihen (im `Reader`–Format);

`outFile` der Name der Ausgabedatei; optional, voreingestellt Standardausgabe.

- **matcorrelation**

```
usage: matcorrelation [-v] shifts infoFile [outFile]
```

Argumente:

`-v` verlangt eine ausführlichere Ausgabe;

`shifts` die Anzahl der Verschiebungen der Exogenen; die Anfangsverschiebung wird in `infoFile` angegeben als der Wert des Parameters `ADVANCE;`

`infoFile` der Name der Infodatei (im `Reader`–Format);

`outFile` der Name der Ausgabedatei; optional, voreingestellt Standardausgabe.

- **materror**

```
usage: materror infoFile [mask]
      mask := binary, from '00000000' to '11111111'
      to show ME,MAE,MRE,MAPE,SSE,MSE,RMSE,THEIL
      default '00010011' to show MAPE, RMSE, THEIL
      example: materror inFile 01110011
```

Argumente:

`infoFile` der Name der Infodatei mit Einstellungen im `Reader`–Format;

`mask` gibt an, welche Kennzahlen angezeigt und welche ausgeblendet werden sollen (s. `usage`).

- **matmodel**

```
usage: matmodel [-v] [-e nExos] -s shifts infoFile [outFile]
```

Argumente:

`-v`	verlangt eine ausführlichere Ausgabe;
`-e nExos`	die maximale Anzahl der Exogenen; optional, falls nicht angegeben wird, wird automatisch berechnet;
`shifts`	die Anzahl der Verschiebungen der Exogenen. Die Anfangsverschiebung wird in `infoFile` angegeben als der Wert des Parameters `ADVANCE`;
`infoFile`	der Name der Infodatei (im `Reader`–Format);
`outFile`	der Name der Ausgabedatei; optional, voreingestellt Standardausgabe.

- **matregression**

```
usage: matregression infoFile [outFile]
```

Argumente:

`infoFile`	der Name der Infodatei (in `Reader`–Format) mit Modelleinstellungen;
`outFile`	der Name der Ausgabedatei; optional, voreingestellt Standardausgabe.

Anlage C: Perl–Module

- **aggregate.pl**

```
usage: aggregate.pl [-how a|s] -old d|w5|w6|w|m|q|y -new d|w|m|q|y comp1 .. compN
      -how m : average(default), s : sum
      -old    : old period: d:daily, w:weekly, m:monthly, q:quotly,
                            w5(6):5(6)workdays-a-week, y:yearly
      -new : new period
```

Argumente:

`-how a|s`	die Aggregationsfunktion: Mittelwertbildung „a“ (*average*) oder Summieren „s“; optional, voreingestellt „a“;
`-old`	alte Zeiteinheiten, siehe `usage`;
`-new`	neue Zeiteinheiten, siehe `usage`;
`komp1...kompN`	der logische Pfad zur Zeitreihe, anfangend mit dem Namen des aktuellen Projekts und endend mit dem Namen der Zeitreihe (ohne Zeiteinheiten).

- **convertForReader.pl**

```
usage: convertForReader.pl inFile outFile proj(KOMP1) [KOMP2 ...]
```

Argumente:

`inFile`	die Eingabedatei im Format des Prognosesystems;
`outFile`	Ausgabedatei im `Reader`–Format;
`proj`	Aktives Projekt;
`KOMP`*i*	Eine Komponente des logischen Pfades zu den Zeitreihen, wodurch der Ausdruck `$KOMP`*i* in `inFile` ersetzt wird; `$KOMP1`wird immer durch den Namen des aktiven Projektes ersetzt.

- **matcorrelation.pl**

```
usage:matcorrelation.pl [-v] [-minshift n] -shifts m   infofile comment
      outfnm proj(KOMP1) [KOMP [..]]
      minShift : default 0 / ADVANCE from infofile
      maxShift : default   VAL_PAIRS from infofile
```

Argumente:

-v	verlangt eine ausführlichere Ausgabe;
-minshift n	die kleinste Verschiebung der Exogenen, optional, voreingestellt 0;
-shifts m	die Anzahl der Verschiebungen;
infoFile	der Name der Infodatei im Format vom Prognosesystem;
comment	der Kommentar, der in den Kopf der erstellten Prognosezeitreihen eingetragen wird;
outfnm	der Name der Ausgabedatei;
proj(KOMP1)	aktives Projekt;
KOMP*i*	eine Komponente des logischen Pfades zu den Zeitreihen, wodurch der Ausdruck $KOMP*i* in inFile ersetzt wird; $KOMP1 wird immer durch den Namen des aktiven Projektes ersetzt. Optional.

- **materror.pl**

```
usage: materror.pl [-mask iiiiiiii] [-out outFnm] [-com] [-split] {-info
infofile]..|-paths origZR progZR1 .. progZRn}
     -coms   := with comments
     i=0|1   for ME,MAE,MRE,MAPE,SSE,MSE,RMSE,THEIL
     default: outFnm="errTbl", iiiiiiii=00010011
```

Argumente:

-mask	gibt an, welche Kennzahlen angezeigt und welche ausgeblendet werden sollen (s.usage) ;
-out outFnm	der Name der Ausgabedatei, optional, voreingestelt „errTbl“ ;
-com	falls dieses angegeben wird, werden Kommentare der verglichenen Zeitreihen in den Ausgabetabellen angezeigt;
-split	falls dieses angegeben wird, werden nach MAPE und RMSE sortierte Ergebnistabellen nicht in einer Datei <outFnm>, sondern in zwei Dateien „<outFnm>.MAPE“ und „<outFnm>.RMSE“ gespeichert;
-info infoFile	der Name der Infodatei mit Deklarationen der Zeitreihen (im Format des Prognosesystems) ;
-paths origZR progZR1... progZRn	alternativ zu dem Parameter -info, erlaubt die Angabe der Pfade zu den Rümpfen der analysierten Zeitreihe (reguläre Ausdrücke sind erlaubt.

- **matmodel.pl**

```
usage: matmodel.pl [-v] [-minshift n] -shifts m [-exos l]      infofile
       comment outfnm proj(KOMP1) [subpr(KOMP2)[..]]
```

Argumente:

-v	verlangt ausführlichere Ausgabe;
`-minshift n`	die kleinste Verschiebung jeder Exogenen; optional;
`-shifts m`	die Anzahl der betrachteten Verschiebungen;
`-exos l`	die maximale Exogenenzahl; optional, falls nicht angegeben, wird sie automatisch berechnet;
`infoFile`	der Name der Infodatei im Format des Prognosesystems;
`outfnm`	der Name der Ausgabedatei;
`proj(KOMP1)`	aktives Projekt;
`KOMP`*i*	eine Komponente des logischen Pfades zu den Zeitreihen, wodurch der Ausdruck `$KOMP`*i* in Deklarationen von Zeitreihen in der Datei `inFile` ersetzt wird. `$KOMP1` wird immer durch den Namen des aktiven Projektes ersetzt; optional.

- **matregression.pl**

```
usage: matregression.pl infofile comment proj(KOMP1) [subpr (KOMP2) [..]]
```

Argumente:

`infoFile`	der Name der Infodatei im Format des Prognosesystems;
`comment`	der Kommentar, der in den Kopf der erstellten Prognosezeitreihen eingetragen wird. Alle Ausgaben des Programms werden in dem Verzeichnis „`comment`“ gespeichert;
`proj(KOMP1)`	aktives Projekt;
`KOMP`*i*	eine Komponente des logischen Pfades zu den Zeitreihen, wodurch der Ausdruck `$KOMP`*i* in `inFile` ersetzt wird. `$KOMP1` wird immer durch den Namen des aktiven Projektes ersetzt; optional.

- **optimizeTop.pl**

```
usage: optimizeHidden.pl proj Hmin Hmax [sk] [rmse|mape|tste]
       default err - rmse
       in templates $KOMP1 admissible only for project
```

Argumente:

`proj`	aktives Projekt;
`Hmin`	die minimale Anzahl von versteckten Neuronen;
`Hmax`	die maximale Anzahl von versteckten Neuronen;
`sk`	optional, falls angegeben, werden die Netze jeweils mit und ohne *shortcuts* überprüft;
`rmse|mape|tste`	optional, bestimmt den Optimierungskriterium, voreingestellt ist RMSE[50].

Im Arbeitsverzeichnis müssen folgende Dateien vorhanden sein:

1. `BP.<name>` - eventuell mehrere Dateien im Prognosesystem–Format, die jeweils ein bestimmtes Neuronales Netz beschreiben. Der Parameter `HIDDEN` ist irrelevant, weil er in jedem Schritt ersetzt wird. Dasselbe gilt für den Parameter `SHORTCUT`, wenn das Kommandozeileargument `sk` angegeben wird;
2. `komps` - die Datei mit der Modellbeschreibung im Format des Prognose–systems (außer der Beschreibung des Neuronalen Netzes), die mindestens die Parameter `MEASURE`, `VAL_PAIRS`, `OUTPUT`, `USE` und optional `UNKNOWN` enthält. Die Ausdrücke „`$KOMP1`“ bei den Zeitreihenbeschreibungen werden durch den Namen des aktiven Projektes `proj` ersetzt.

- **optimizeInput.pl**

```
usage: optimizeInput.pl project shifts [rmse|mape|tste]
     default err - rmse
     in templates admissible only $KOMP1 (for project)!
```

Argumente:

`projekt`	aktives Projekt;
`shifts`	die Anzahl der in Frage kommenden Verschiebungen der Exogenen;
`rmse|mape|tste`	optional, bestimmt den Optimierungskriterium, voreingestellt ist RMSE;

Im Arbeitsverzeichnis müssen folgende Dateien vorhanden sein:

1. `BP.head` - die Infodatei im Format des Prognosesystems, wobei ein bestimmtes Neuronales Netz beschrieben wird (die Parameter

[50] Über Fehlerarten siehe Kapitel 4.

USE, UNKNOWN, OUTPUT, VAL_PAIRS und MEASURE sind irrelevant und werden ignoriert) und die Datei mit den Einstellungen für die Sensitivitätsanalyse „BP.sense". Optional, voreingestellt „ . “ ;

2. komps - die Datei mit der Modellbeschreibung (außer der Beschreibung des Neuronalen Netzes) im Format des Prognosesystems. Die Ausdrücke $KOMP1 bei den Zeitreihenbeschreibungen werden durch den Namen des aktiven Projektes proj ersetzt. Ein Beispiel siehe in der **Fehler! Verweisquelle konnte nicht gefunden werden.**;

3. ALL.template - die Datei im Format des Prognosesystems mit Beschreibungen der Zeitreihen, aus deren durch die Verschiebungen Indikatoren erstellt werden. Diese Indikatoren werden als potentielle Inputzeitreihen betrachtet. Anfangsverschiebung bzw. die Anzahl der Verschiebungen wird durch den Kommandozeileargument shifts bzw. durch den Parameter ADVANCE in dieser Datei angegeben;

4. sense.block - der Teil einer Infodatei im Format des Prognosesystems mit Parametern der Sensitivitätsanalyse.

- **plot_cor.pl**

```
usage: plot_cor.pl  [-max nExos] fname
      nExos -     Maximale Anzahl der Exogena in einem Fenster
      fname -     inputFile (output von correlation.pl -v ...)
```

- **plot_selection.pl**

```
usage: plot_selection fname
      fname -     inputFile (output von selectModelNN.pl)
```

- **scale_for_snns.pl**

```
usage: scale_for_snns.pl [-lin] [-model file] infoFile proj(KOMP1) [KOMP2 ...]
          -lin   :=  SHIFT k | LIN (default SHIFT k | SIG )
          -model :=  MODELL-file-path,(default activePrjHome/MODELL)
```

Argumente:

-lin falls dieses angegeben ist, werden Zeitreihen linear skaliert, sonst normiert;

`-model`	der Pfad zu der Datei in einem vorgegebenen Format, die Beschreibungen von Skalierungen und dazugehörigen Suffixes der erstellten Zeitreihen enthält;
`infoFile`	die Datei mit Beschreibungen der Zeitreihen im Format des Prognosesystems;
`proj`	aktives Projekt;
`KOMP`*i*	eine Komponente des logischen Pfades zu den Zeitreihen, wodurch der Ausdruck `$KOMP`*i* in den Zeitreihendeklarationen in der Datei `inFile` ersetzt wird. `$KOMP1` wird immer durch den Namen des aktiven Projektes ersetzt; optional.

- **searchNN.pl**

```
usage: searchNN.pl project Imin Imax Hmin Hmax [-t tolerance] [-h
headerDir] [-s steps] [-c m|l]
         tolerance default 0.001, headerDir default '.'
         steps (max.number ineff.steps) default 20
         c(hildren): m(ore) or l(ess), default m
```

Argumente:

`projekt`	aktives Projekt;
`Imin`	minimale Anzahl von Inputzeitreihen;
`Imax`	maximale Anzahl von Inputzeitreihen;
`Hmin`	minimale Anzahl von versteckten Neuronen;
`Hmax`	maximale Anzahl von versteckten Neuronen;
`-t tolerance`	die obige Grenze der Verschlechterung des Modells, die toleriert wird;
`-h headerDir`	das Verzeichnis mit den Dateien `BP.FF`, `BP.Elman` und `BP.Jordan`, die entsprechende Neuronale Netze beschreiben (die Parameter `HIDDEN`, `SHORTCUT`, `USE`, `UNKNOWN`, `OUTPUT`, `VAL_PAIRS` und `MEASURE` sind irrelevant und werden ignoriert) und die Datei mit den Einstellungen für die Sensitivitätsanalyse `BP.sense`. Optional, voreingestellt „ . “ ;
`-s steps`	die maximale Anzahl der Schritte nach der letzten wesentlichen Verbesserung des Modells; falls überschritten wird, bricht das Programm ab;
`-c m|l`	`l`(less) bedeutet, daß als Nachfolger nur direkte Nachbarn betrachtet werden (s. **Fehler! Verweisquelle konnte nicht gefunden werden.**); m(more), daß auch direkte Nachbarn von direkten

Nachbarn als Nachfolger genommen werden (s. **Fehler! Verweisquelle konnte nicht gefunden werden.**, U_3) .

Im Arbeitsverzeichnis muß die Datei `komps` (im Format des Prognosesystems) mit der Modellbeschreibung vorhanden sein. Ein Beispiel siehe in **Fehler! Verweisquelle konnte nicht gefunden werden.**.

- **selectModelNN.pl**

```
usage: a.pl project Imin Imax Hmin Hmax [-h headerDir] [-m mu] [-t
{j,e,f}] [-test <Tmin>-<Tmax>] [-T tolerance]
      headerDir_g default '.'
      mu parents in every generation, default 10
      topologies, default jef := jordan,elman,ff, SK on and off
      -test := variation der Testmenge (in %), default Tmin=26, Tmax=30
      -T tolerance, default 0.05
```

Agrumente:

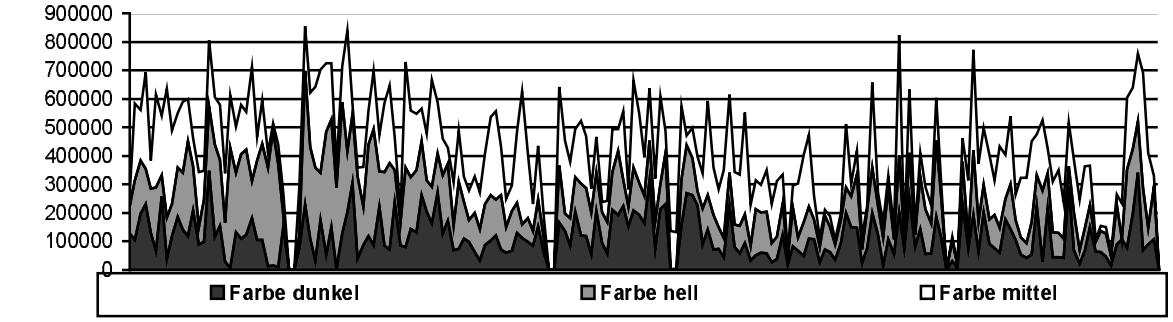
`-t jef`	stehen für Topologien der Neuronalen Netze j(ordan), e(lman) und f(eedforward), die jeweils mit und ohne *schortcuts* an der Selektion teilnehmen; voreingestellt sind alle drei;
`projekt`	aktives Projekt;
`Imin` bzw. `Imax`	minimale bzw. maximale Anzahl von Inputzeitreihen;
`Hmin` bzw. `Hmax`	minimale bzw. maximale Anzahl von versteckten Neuronen;
`-h headerDir`	Die Verzeichnis mit Dateien „`BP.FF`", „`BP.Elman`" und „`BP.Jordan`", die entsprechende Neuronale Netze beschreiben (die Parameter `HIDDEN`, `SHORTCUT`, `USE`, `UNKNOWN`, `OUTPUT`, `VAL_PAIRS` und `MEASURE` sind irrelevant und werden ignoriert) und die Datei mit den Einstellungen für die Sensitivitätsanalyse „`BP.sense`". Optional, voreingestellt „ .“ ;
`-m mu`	Die Anzahl der Individuen mu (μ), die in jeder Generation als Eltern gewählt werden, voreingestellt ist μ =10;
`-test Tmin-Tmax`	mit `Tmin` und `Tmax` kann angegeben werden, in welchen Grenzen wird die Größe der Testmenge (in %) bei verschiedenen Generationen variiert;
`-T tolerance`	die maximale zu tolerierende Verschlechterung des Modells und die maximale „unwesentliche“ Verbesserung des Modells.

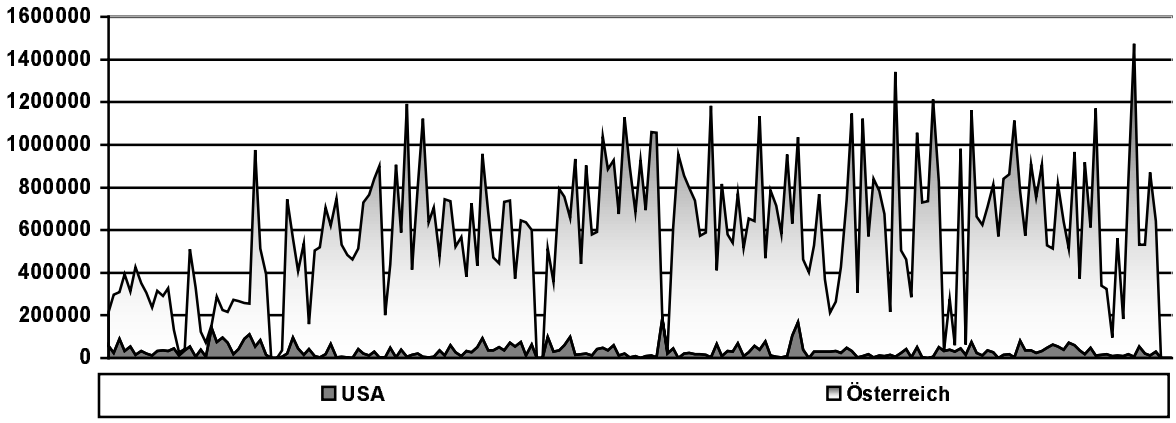
Im Arbeitsverzeichnis muß die Datei „`komps`" (im Format des Prognosesystems) mit der Modellbeschreibung vorhanden sein. Ein Beispiel siehe in **Fehler! Verweisquelle konnte nicht gefunden werden.**.

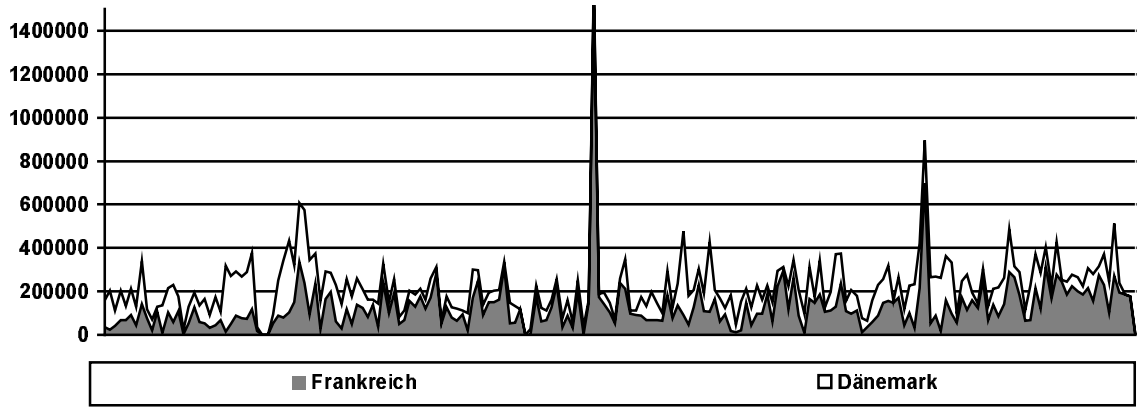
Anlage D: Empirische Daten

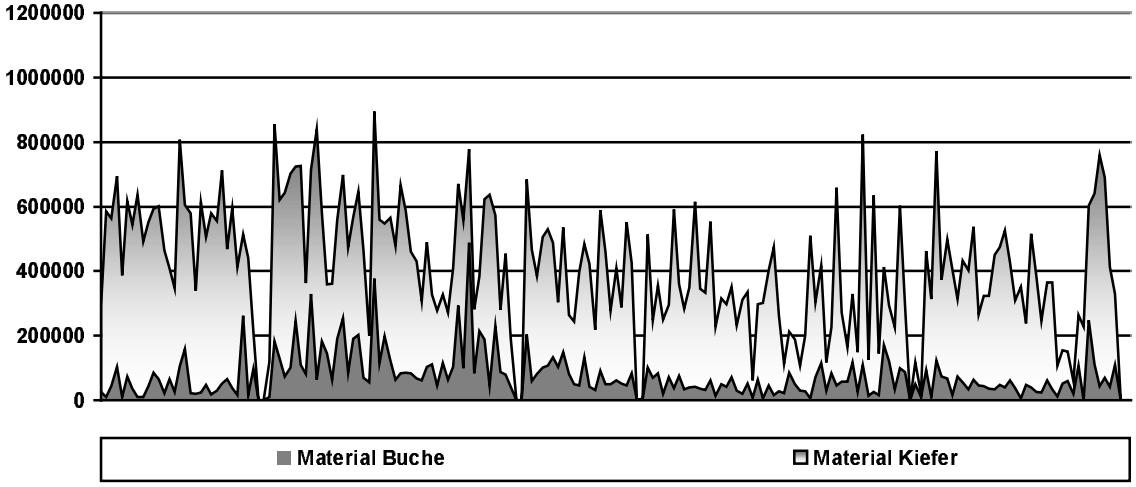
Endogene Umsatzzeitreihen

Beschreibung:	Umsatzdaten eines Küchenmöbelherstellers, gruppiert nach verschiedene Farbtöner, Material und Zielländer
Periodizität:	Wochen
Maßeinheit:	DM
Zeitraum:	1. Woche 1994 – 39. Woche 1997
Anzahl der Werte: (in jeder Zeitreihe)	195 (abzgl. 4 bis 8 fehlende Werte)









Exogene wirtschaftliche Zeitreihen vom Statistischen Bundesamt[51]

Statistik der Arbeitslosigkeit ab 1960

```
------------------------------------------------------------------------
ZR-NR:1023001                                                  ME:1000
FRUEHERES BUNDESGEBIET
    BESCHAEFTIGUNG UND ARBEITSMARKT
        SAISONBEREINIGT
            DURCHSCHNITT IM MONAT
                ERWERBSTAETIGE(PERSONEN MIT ARBEITSORT IM FRUE-
                 HEREN BUNDESGEBIET(INLANDSKONZEPT))
------------------------------------------------------------------------
ZR-NR:1023002                                                  ME:1000
FRUEHERES BUNDESGEBIET
    BESCHAEFTIGUNG UND ARBEITSMARKT
        SAISONBEREINIGT
            STAND AM MONATSENDE
                ARBEITSLOSE
------------------------------------------------------------------------
ZR-NR:1023003                                                  ME:%
FRUEHERES BUNDESGEBIET
    BESCHAEFTIGUNG UND ARBEITSMARKT
        SAISONBEREINIGT
            STAND AM MONATSENDE
                ARBEITSLOSENQUOTE GEMESSEN AN DEN ERWERBSPER-
                 SONEN INSGESAMT(ERWERBSPERSONEN NACH DEM MIKRO-
                 ZENSUS(JEWEILS OHNE SOLDATEN))
```

[51] Die Daten sind unter der Internet-Adresse www.statistik-bund.de erhältlich

Statistik der Baufertigstellungen ab 1952

```
------------------------------------------------------------------------
ZR-NR:0088001                                              ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            GEBAEUDE
                WOHNGEBAEUDE
------------------------------------------------------------------------
ZR-NR:0088008                                              ME:TSDCBM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            RAUMINHALT
                 WOHNGEBAEUDE
------------------------------------------------------------------------
ZR-NR:0088015                                              ME:1000QM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            NUTZFLAECHE
                WOHNGEBAEUDE
------------------------------------------------------------------------
ZR-NR:0088022                                              ME:1000QM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            WOHNFLAECHE IN WOHNUNGEN
                WOHNGEBAEUDE

------------------------------------------------------------------------
ZR-NR:0088029                                              ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            WOHNUNGEN
                INSGESAMT
------------------------------------------------------------------------
ZR-NR:0088030                                              ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            WOHNUNGEN
                WOHNGEBAEUDE
                    ZUSAMMEN

------------------------------------------------------------------------
ZR-NR:0088041                                              ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        ERRICHTUNG NEUER GEBAEUDE
            WOHNRAEUME EINSCHL.KUECHEN
                WOHNGEBAEUDE
------------------------------------------------------------------------
ZR-NR:0088049                                              ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
```

```
        ERRICHTUNG NEUER GEBAEUDE
            VERANSCHLAGTE KOSTEN DES BAUWERKES
                WOHNGEBAEUDE
                    ZUSAMMEN
------------------------------------------------------------------
ZR-NR:0088054                                          ME:1000QM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        BAUMASSNAHMEN AN BESTEHENDEN GEBAEUDEN
            NUTZFLAECHE
WOHNGEBAEUDE
------------------------------------------------------------------
ZR-NR:0088061                                          ME:1000QM
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        BAUMASSNAHMEN AN BESTEHENDEN GEBAEUDEN
            WOHNFLAECHE IN WOHNUNGEN
                WOHNGEBAEUDE
------------------------------------------------------------------
ZR-NR:0088068                                          ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        BAUMASSNAHMEN AN BESTEHENDEN GEBAEUDEN
            WOHNUNGEN
                WOHNGEBAEUDE
------------------------------------------------------------------
ZR-NR:0088075                                          ME:ANZAHL
FRUEHERES BUNDESGEBIET
    BAUFERTIGSTELLUNGEN IM HOCHBAU
        BAUMASSNAHMEN AN BESTEHENDEN GEBAEUDEN
            WOHNRAEUME EINSCHL.KUECHEN
                WOHNGEBAEUDE
```

Statistik der Wirtschaftsrechnungen ausgewählter privater Haushalte ab 1950

```
------------------------------------------------------------------
ZR-NR:2876019    SYST-NR:4111-4959                          ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 1:2-PERSONEN-HAUSHALTE VON REN-
         TEN-UND SOZIALHILFEEMPFAENGERN MIT GERINGEM
         EINKOMMEN
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL,HAUSHALTSGERAETE U.AE.FUER DIE HAUSHALTS-
                 FUEHRUNG
------------------------------------------------------------------
ZR-NR:2876020    SYST-NR:4111-4199                          ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 1:2-PERSONEN-HAUSHALTE VON REN-
         TEN-UND SOZIALHILFEEMPFAENGERN MIT GERINGEM
         EINKOMMEN
```

```
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL(EINSCHL.MATRATZEN)
-----------------------------------------------------------------------
ZR-NR:2876022    SYST-NR:4311-4389                              ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 1:2-PERSONEN-HAUSHALTE VON REN-
         TEN-UND SOZIALHILFEEMPFAENGERN MIT GERINGEM
         EINKOMMEN
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                HEIZ- UND KOCHGERAETE,ANDERE ELEKTRISCHE
                 HAUSHALTSGERAETE
-----------------------------------------------------------------------
ZR-NR:2876076    SYST-NR:4111-4959                              ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 2:4-PERSONEN-HAUSHALTE VON ARBEI-
         TERN UND ANGESTELLTEN MIT MITTLEREM EINKOMMEN
         EINES ALLEINVERDIENENDEN EHEPARTNERS
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL,HAUSHALTSGERAETE U.AE.FUER DIE HAUSHALTS-
                 FUEHRUNG
-----------------------------------------------------------------------
ZR-NR:2876077    SYST-NR:4111-4199                              ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 2:4-PERSONEN-HAUSHALTE VON ARBEI-
         TERN UND ANGESTELLTEN MIT MITTLEREM EINKOMMEN
         EINES ALLEINVERDIENENDEN EHEPARTNERS
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL(EINSCHL.MATRATZEN)
-----------------------------------------------------------------------
ZR-NR:2876079    SYST-NR:4311-4389                              ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 2:4-PERSONEN-HAUSHALTE VON ARBEI-
         TERN UND ANGESTELLTEN MIT MITTLEREM EINKOMMEN
         EINES ALLEINVERDIENENDEN EHEPARTNERS
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                HEIZ- UND KOCHGERAETE,ANDERE ELEKTRISCHE
                 HAUSHALTSGERAETE
-----------------------------------------------------------------------
ZR-NR:2876133    SYST-NR:4111-4959                              ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
```

```
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 3:4-PERSONEN-HAUSHALTE VON BE-
         AMTEN UND ANGESTELLTEN MIT HOEHEREM EINKOMMEN
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL,HAUSHALTSGERAETE U.AE.FUER DIE HAUSHALTS-
                 FUEHRUNG
-----------------------------------------------------------------
ZR-NR:2876134    SYST-NR:4111-4199                          ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 3:4-PERSONEN-HAUSHALTE VON BE-
         AMTEN UND ANGESTELLTEN MIT HOEHEREM EINKOMMEN
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                MOEBEL(EINSCHL.MATRATZEN)
-----------------------------------------------------------------
ZR-NR:2876136    SYST-NR:4311-4389                          ME:DM
FRUEHERES BUNDESGEBIET
    AUSGABEN FUER DEN PRIVATEN VERBRAUCH JE HAUS-
     HALT UND MONAT
      * HAUSHALTSTYPEN
        HAUSHALTSTYP 3:4-PERSONEN-HAUSHALTE VON BE-
         AMTEN UND ANGESTELLTEN MIT HOEHEREM EINKOMMEN
            VERWENDUNGSZWECK
              * AUSGABEN(ANGELEHNT AN DIE SEA,AUSGABE 1983)
                HEIZ- UND KOCHGERAETE,ANDERE ELEKTRISCHE
                 HAUSHALTSGERAETE
```

Statistik der Bausparkassen ab 1951

```
-----------------------------------------------------------------
ZR-NR:0266001                                        ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        EINZAHLUNGEN
            SPARGELDEINZAHLUNGEN
-----------------------------------------------------------------
ZR-NR:0266002                                        ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        EINZAHLUNGEN
            WOHNUNGSBAUPRAEMIEN
-----------------------------------------------------------------
ZR-NR:0266003                                        ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        EINZAHLUNGEN
            ZINS-UND TILGUNGSLEISTUNGEN
-----------------------------------------------------------------
ZR-NR:0266004                                        ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        EINLAGEN UND AUFGENOMMENE GELDER
            SPAREINLAGENBESTAND (EINSCHL.VERRECHNETER
             BAUSPARPRAEMIEN)
```

```
------------------------------------------------------------------
ZR-NR:0266005                                          ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        EINLAGEN UND AUFGENOMMENE GELDER
            AUFGENOMMENE FREMDMITTEL
------------------------------------------------------------------
ZR-NR:0266006                                          ME:1000DM
FRUEHERES BUNDESGEBIET
    BAUSPARGESCHAEFTE
        BAUDARLEHEN
            HYPOTHEKARKREDITE
```

Statistik der saisonbereinigte Wirtschaftszahlen der Deutschen Bundesbank ab 1960

```
------------------------------------------------------------------
ZR-NR:1025003                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PRODUKTIVITAET UND LOHNKOSTEN IM PRODUZIERENDEN
     GEWERBE(OHNE BAUGEWERBE UND ENERGIEVERSORGUNG)
        SAISONBEREINIGT,1991=100
            LOEHNE UND GEHAELTER JE BESCHAEFTIGTEN
------------------------------------------------------------------
ZR-NR:1025004                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PRODUKTIVITAET UND LOHNKOSTEN IM PRODUZIERENDEN
     GEWERBE(OHNE BAUGEWERBE UND ENERGIEVERSORGUNG)
        SAISONBEREINIGT,1991=100
            LOEHNE UND GEHAELTER JE BESCHAEFTIGTENSTUNDE

------------------------------------------------------------------
ZR-NR:1025005                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PRODUKTIVITAET UND LOHNKOSTEN IM PRODUZIERENDEN
     GEWERBE(OHNE BAUGEWERBE UND ENERGIEVERSORGUNG)
        SAISONBEREINIGT,1991=100
            LOEHNE UND GEHAELTER JE PRODUKTEINHEIT
```

Statistik der Indizes des Auftragseingangs und des Umsatzes des verarbeitende Gewerbe ab 1977

```
------------------------------------------------------------------
ZR-NR:4654118   SYST-NR:20.20                          ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                FURNIER-,SPERRHOLZ-,HOLZFASERPLATTEN-UND
                 HOLZSPANPLATTENWERKE(20.20)
                    INSGESAMT
------------------------------------------------------------------
ZR-NR:4654119   SYST-NR:20.20                          ME:PUNKTE
DEUTSCHLAND
```

```
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                FURNIER-,SPERRHOLZ-,HOLZFASERPLATTEN-UND
                 HOLZSPANPLATTENWERKE(20.20)
                    INLAND
------------------------------------------------------------------------
ZR-NR:4654120   SYST-NR:20.20                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                FURNIER-,SPERRHOLZ-,HOLZFASERPLATTEN-UND
                 HOLZSPANPLATTENWERKE(20.20)
                    AUSLAND
------------------------------------------------------------------------
ZR-NR:4654133   SYST-NR:21.11                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON HOLZSTOFF UND ZELLSTOFF
                 (21.11)
                    INSGESAMT
------------------------------------------------------------------------
ZR-NR:4654134   SYST-NR:21.11                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON HOLZSTOFF UND ZELLSTOFF
                 (21.11)
                    INLAND
------------------------------------------------------------------------
ZR-NR:4654135   SYST-NR:21.11                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON HOLZSTOFF UND ZELLSTOFF
                 (21.11)
                    AUSLAND
```

```
------------------------------------------------------------------------
ZR-NR:4654207   SYST-NR:24.16                                ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON KUNSTSTOFF IN PRIMAERFORMEN
                 (24.16)
                    AUSLAND
------------------------------------------------------------------------
ZR-NR:4654553   SYST-NR:29.71                                ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON ELEKTRISCHEN HAUSHALTSGERAETEN
                 (29.71)
                    INSGESAMT
------------------------------------------------------------------------

ZR-NR:4654554   SYST-NR:29.71                                ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON ELEKTRISCHEN HAUSHALTSGERAETEN
                 (29.71)
                    INLAND
------------------------------------------------------------------------
ZR-NR:4654555    SYST-NR:29.71                               ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON ELEKTRISCHEN HAUSHALTSGERAETEN
                 (29.71)
                    AUSLAND
------------------------------------------------------------------------

ZR-NR:4654670    SYST-NR:36.1                                ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
```

```
                GEWERBE
               HERSTELLUNG VON MOEBELN(36.1)
                   INSGESAMT
------------------------------------------------------------------------
ZR-NR:4654671   SYST-NR:36.1                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
               HERSTELLUNG VON MOEBELN(36.1)
                   INLAND
------------------------------------------------------------------------
ZR-NR:4654672   SYST-NR:36.1                                  ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
               HERSTELLUNG VON MOEBELN(36.1)
                   AUSLAND
------------------------------------------------------------------------
ZR-NR:4654673   SYST-NR:36.11                                 ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
               HERSTELLUNG VON SITZMOEBELN(36.11)
                   INSGESAMT
------------------------------------------------------------------------
ZR-NR:4654674   SYST-NR:36.11                                 ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
               HERSTELLUNG VON SITZMOEBELN(36.11)
                   INLAND
------------------------------------------------------------------------
ZR-NR:4654675   SYST-NR:36.11                                 ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
               HERSTELLUNG VON SITZMOEBELN(36.11)
                   AUSLAND
```

```
------------------------------------------------------------------
ZR-NR:4654679   SYST-NR:36.13                          ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON KUECHENMOEBELN(36.13)
                    INSGESAMT
------------------------------------------------------------------
ZR-NR:4654680   SYST-NR:36.13                          ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON KUECHENMOEBELN(36.13)
                    INLAND
------------------------------------------------------------------
ZR-NR:4654681    SYST-NR:36.13                          ME:PUNKTE
DEUTSCHLAND
    INDEX DES UMSATZES
        WERTINDEX(1991=100)
            ORIGINALWERT
              * KLASSIFIKATION DER WIRTSCHAFTSZWEIGE AUSGABE
                 1993(WZ93)-GLIEDERUNG FUER DAS PRODUZIERENDE
                 GEWERBE
                HERSTELLUNG VON KUECHENMOEBELN(36.13)
```

Statistik vom Preisindex für den Wareneingang des produzierenden Gewerbes (ab 1985)

```
------------------------------------------------------------------
ZR-NR:2651001                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
          * WARENEINGANG
            WARENEINGANG INSGESAMT
 ----------------------------------------------------------------
ZR-NR:2651002                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
          * WARENEINGANG
            WARENEINGANG INLAENDISCHER HERKUNFT
 ----------------------------------------------------------------
ZR-NR:2651003                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
          * WARENEINGANG
```

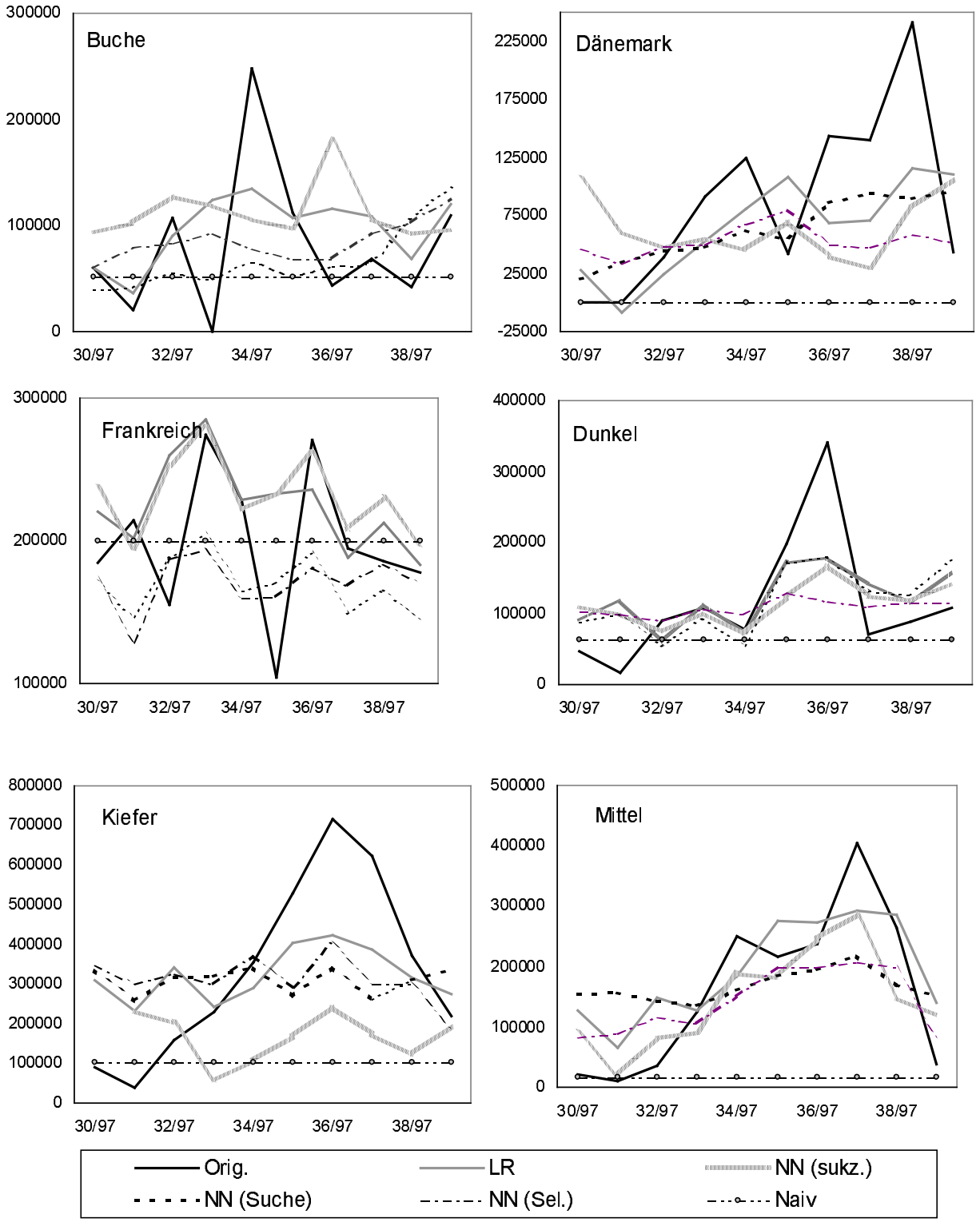
```
            WARENEINGANG AUSLAENDISCHER HERKUNFT
 ------------------------------------------------------------------
ZR-NR:2651023                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                ROHHOLZ
 ------------------------------------------------------------------
ZR-NR:2651024                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                LAUBROHHOLZ

------------------------------------------------------------------
ZR-NR:2651025                                          ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                NADELROHHOLZ
------------------------------------------------------------------
ZR-NR:2651080                                            ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                FARBSTOFFE,FARBEN UND VERWANDTE ERZEUGNISSE
------------------------------------------------------------------
ZR-NR:2651085                                            ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                SCHNITTHOLZ,SPERRHOLZ UND SONSTIGES BEARBEITE-
                 TES HOLZ
------------------------------------------------------------------
ZR-NR:2651086                                            ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                SCHNITTHOLZ,SPERRHOLZ UND SONSTIGES BEARBEITE-
                 TES HOLZ,INLAENDISCH
```

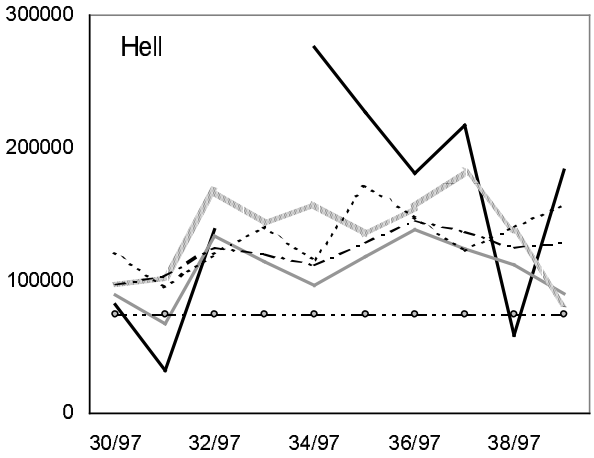
```
------------------------------------------------------------------------
ZR-NR:2651087                                                 ME:PUNKTE
FRUEHERES BUNDESGEBIET
    PREISINDEX FUER DEN WARENEINGANG DES PRODU-
     ZIERENDEN GEWERBES
        1985=100
            NACH DER HERKUNFT
              * GUETERGRUPPEN,-UNTERGRUPPEN BZW.-ARTEN
                SCHNITTHOLZ,SPERRHOLZ UND SONSTIGES BEARBEITE-
                 TES HOLZ,IMPORTIERT
```

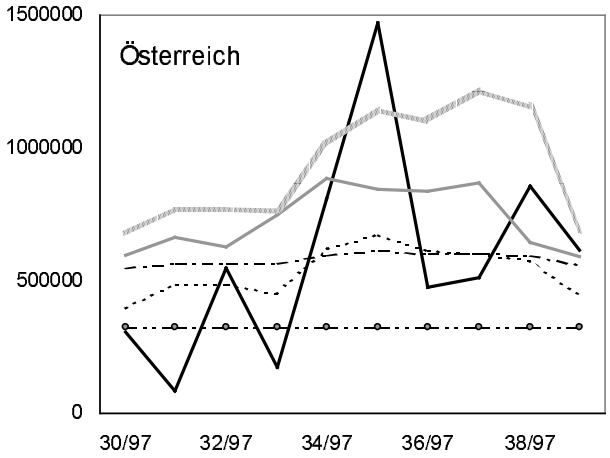
Statistik der Außenhandel ab 1950

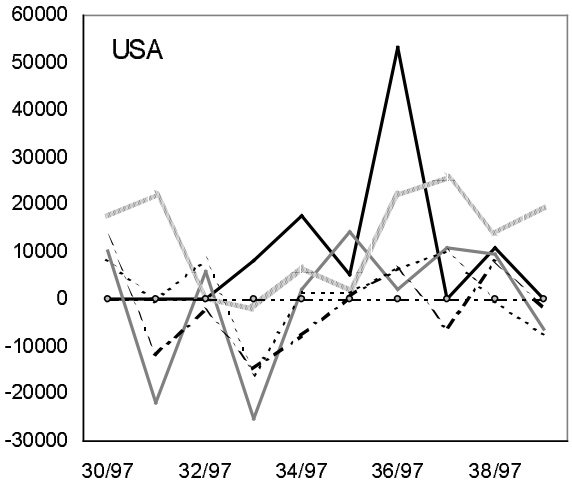
```
------------------------------------------------------------------------
ZR-NR:1531024   SYST-NR:53                                  ME:1000DM
DEUTSCHLAND(BIS 1995 FRUEHERES BUNDESGEBIET)
    WARENEINFUHR
        VOLUMEN
          * SYSTEMATISCHES GUETERVERZEICHNIS FUER PRO-
             DUKTIONSSTATISTIKEN,AUSGABE 1982
            ERZEUGNISSE DES GRUNDSTOFF-UND PRODUKTIONS-
             GUETERGEWERBES
              * SYSTEMATISCHES GUETERVERZEICHNIS FUER PRO-
                 DUKTIONSSTATISTIKEN,AUSGABE 1982
                SCHNITTHOLZ,SPERRHOLZ UND SONSTIGES BEARBEI-
                 TETES HOLZ
------------------------------------------------------------------------
ZR-NR:1531076   SYST-NR:53                                  ME:1000DM
DEUTSCHLAND(BIS 1995 FRUEHERES BUNDESGEBIET)
    WARENAUSFUHR
        VOLUMEN
          * SYSTEMATISCHES GUETERVERZEICHNIS FUER PRO-
             DUKTIONSSTATISTIKEN,AUSGABE 1982
            ERZEUGNISSE DES GRUNDSTOFF-UND PRODUKTIONS-
             GUETERGEWERBES
              * SYSTEMATISCHES GUETERVERZEICHNIS FUER PRO-
                 DUKTIONSSTATISTIKEN,AUSGABE 1982
                SCHNITTHOLZ,SPERRHOLZ UND SONSTIGES BEARBEI-
                 TETES HOLZ
```

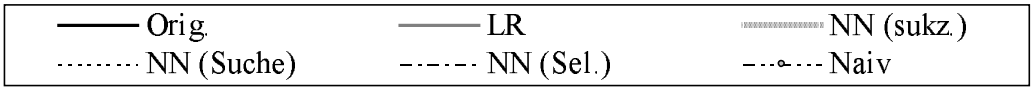
Anlage E: Prognosengraphiken




Hell
300000
200000
100000
0
30/97
32/97
34/97
36/97
38/97


Österreich
1500000
1000000
500000
0
30/97
32/97
34/97
36/97
38/97


USA
60000
50000
40000
30000
20000
10000
0
-10000
-20000
-30000
30/97
32/97
34/97
36/97
38/97


Orig.
LR
NN (sukz.)
NN (Suche)
NN (Sel.)
Naiv

Anlage F: Beispiel eines Selektionsablaufs

Es werden 5 beste Individuen der ersten 8 Generationen bei der Selektion der neuronale Modelle für die Endogene „hell“ angezeigt. Die erste Spalte enthält Kodierungen der Individuen, die zweite ihre Fitneß und die dritte ihren Alter in Evolutionszyklen (*epochs*).

```
***** GENERATION 1 *****
0) 0:11:6:0:1:2:3:4:5        0.14395          1 ep.
1) 2:5:6:0:1:2:3:4:7         0.146242         1 ep.
2) 2:15:7:0:1:2:3:4:6:7      0.146487         1 ep.
3) 1:15:7:0:1:2:3:4:6:8      0.147889         1 ep.
4) 2:8:5:0:1:2:3:4           0.147989         1 ep.
5) 0:6:6:0:1:2:3:4:5         0.148142         1 ep.
...
***** GENERATION 2 *****
0) 2:5:6:0:1:2:3:4:8         0.1386715        1 ep.
1) 2:15:7:0:1:2:3:4:7:8      0.143143         1 ep.
2) 2:13:7:0:1:2:3:4:6:8      0.1439615        1 ep.
3) 2:6:7:0:1:2:3:4:6:8       0.144135         1 ep.
4) 2:9:5:0:1:2:4:8           0.145426         1 ep.
5) 4:12:5:0:1:2:4:8          0.1455515        1 ep.
...
***** GENERATION 3 *****
0) 0:9:5:0:1:2:4:8           0.1298515        1 ep.
1) 0:12:5:0:1:2:4:8          0.133561         1 ep.
2) 2:9:6:0:1:2:3:4:8         0.1375785        1 ep.
3) 2:14:7:0:1:2:3:4:6:8      0.1398905        1 ep.
4) 2:5:6:0:1:2:3:4:8         0.1400363333     2 ep.
5) 2:6:6:0:1:2:3:4:8         0.14074          1 ep.
...
***** GENERATION 4 *****
0) 0:12:5:0:1:2:4:8          0.1331493333     2 ep.
1) 0:9:5:0:1:2:4:8           0.1356506667     2 ep.
2) 0:9:4:1:2:4:8             0.137703         1 ep.
3) 0:13:5:0:1:2:4:8          0.141522         1 ep.
4) 2:4:6:0:1:2:3:4:8         0.1416906667     2 ep.
5) 2:9:6:0:1:2:3:4:8         0.1424226667     2 ep.
...
```

```
***** GENERATION 5 *****
0) 0:9:4:1:2:8:9             0.128876         1 ep.
1) 0:13:4:1:2:4:8            0.1365           1 ep.
2) 0:10:4:1:2:4:8            0.1387595        1 ep.
3) 0:8:4:1:2:4:8             0.142013         1 ep.
4) 2:4:6:0:1:2:3:4:8         0.1428435        3 ep.
5) 0:12:5:0:1:2:4:8          0.1428445        3 ep.
...
***** GENERATION 6 *****
0) 0:9:4:1:2:8:9             0.1264836667     2 ep.
1) 0:13:4:1:2:4:8            0.1281606667     2 ep.
2) 1:9:4:1:2:8:9             0.128833         1 ep.
3) 0:11:5:0:1:2:4:8          0.12952          2 ep.
4) 0:8:4:1:2:8:9             0.130507         1 ep.
5) 0:9:5:0:1:2:8:9           0.1308015        1 ep.
...
***** GENERATION 7 *****
0) 0:13:4:1:2:4:8            0.12796325       3 ep.
1) 0:9:4:1:2:8:9             0.12799175       3 ep.
2) 1:8:4:1:2:8:9             0.128076         1 ep.
3) 0:11:5:0:1:2:4:8          0.12831775       3 ep.
4) 0:8:5:0:1:2:8:9           0.1301975        1 ep.
5) 0:12:5:0:1:2:8:9          0.1303105        1 ep.
...
***** GENERATION 8 *****
0) 0:8:4:1:2:8:9             0.1258255        3 ep.
1) 0:11:4:1:2:8:9            0.1271836667     2 ep.
2) 0:10:5:0:1:2:8:9          0.12775          2 ep.
3) 0:15:5:0:1:2:8:9          0.1290425        1 ep.
4) 0:12:5:0:1:2:8:9          0.1303416667     2 ep.
5) 0:14:5:0:1:2:8:9          0.13044          2 ep.
...
```

Abbildung 1: Ausschnitt aus der Ausgabedatei von `selectModelNN.pl`

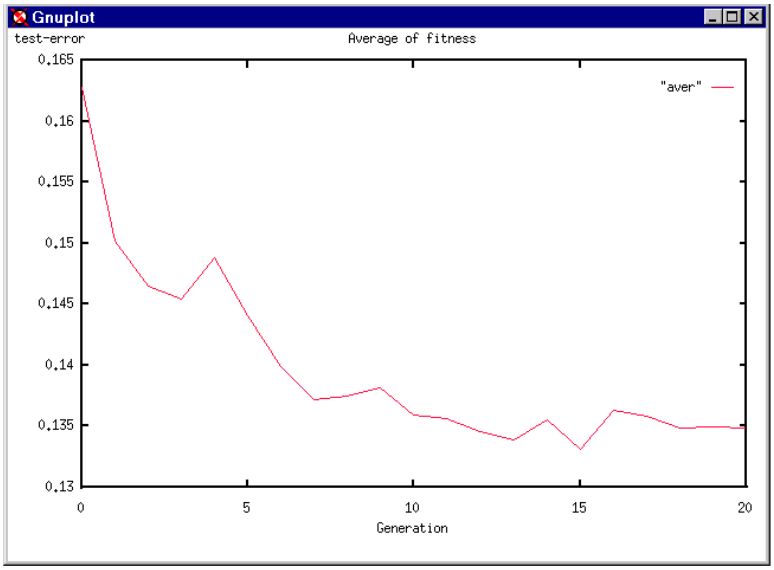


Abbildung 2: Durchschnittliche Fitneß bei der Evolution der Modelle für „hell“

Literaturverzeichnis

[Edw84] Edwards, Allen L.: An Introduction to Linear Regression and Correlation, W.H. Freeman and Company, New York 1984

[Bir93] Birchenhall, C.R.: A Draft Guide to MatClass: A matrix class for C++, Version 1.0d. ftp://ftp.mcc.ac.uk//pub/matclass 1993.

[Behn83] Behncke, Horst: Mathematische Modelle, Univ. Osnabrueck, FB Mathematik; Osnabruecker Schriften zur Mathematik. Reihe Vorlesungsskripten 1983.

[Bra97] Braun, Heinrich: Neuronale Netze: Optimierung durch Lernen und Evolution. Springer–Verlag, Berlin HeidelBerg 1997.

[CoM96] Corsten, Hans; May, Constantin (Hrsg.): Neuronale Netze in der Betriebswirtschaft: Anwendung in Prognose, Klassifikation und Optimierung, Gabler, Wiesbaden 1996.

[RZi94] Rehkugler, Heinz; Zimmermann, Hans Georg (Hrsg.): Neuronale Netze in der Ökonomie: Grundlagen und finanzwirtschaftliche Anwendungen, Franz Vahlen, München 1994.

[Roj96] Rojas, Raul: Theorie der neuronalen Netze: Eine systematische Einführung, Springer–Verlag, Berlin 1996.

[Rüd96] Rüdiger, Wilbert: Interpretation und Anwendung Neuronaler Netze in den Wirtschaftswissenschaften, Peter Lang, Frankfurt am Main 1996.

[Duf92] Dufner, Julius; Jensen, Uwe, Schumacher, Erich: Statistik mit SAS, B. G. Teubner Stuttgart 1992.

[Thi98] Thiesing, Frank M.: Analyse und Prognose von Zeitreihen mit Neuronalen Netzen. Dissertation, Fachbereich Mathematik/Informatik der Universität Osnabrück, Mai 1998.

[Web90] Weber, Karl: Wirtschaftsprognostik, Franz Vahlen, München 1990

[Zel94] Zell, Andreas: Simulation Neuronaler Netze. Addison–Wesley, 1994

[Zel95] Zell, Andreas: *SNNS Stuttgart Neural Network Simulator User Manual, Version 4.1*. University of Stuttgart, Germany, 1995.Report No. 6/95.

Verpflichtungserklärung

Ich versichere, die vorliegende Arbeit selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Osnabrück, den _____

Maria Ananieva