TU

TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Prognose mittels neuronaler Netze am Beispiel der Telemetriedaten

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

O.Univ.-Prof. DI Dr. Adolf Stepan

und

DI Dr. Gerald Brandstetter

E330
Institut für Managementwissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften

von

Beate Edl

e0125291

Finkengasse 27-29/2, 2103 Langenzersdorf

Langenzersdorf, am 14.September 2006

Dank

In erster Linie möchte ich mich bei Prof. Dr. Stepan für die Möglichkeit, diese Arbeit zu verfassen bedanken und für die kritische Durchsicht derselben.

Mein besonderer Dank gilt meinem Betreuer Dr. Brandstetter für seine Unterstützung und sein Engagement bei der Erstellung dieser Arbeit.

Weiters gilt mein Dank meinen Eltern, die mir das Studium ermöglicht haben und auf deren Hilfe jeglicher Art ich mich im Laufe meiner Ausbildung immer verlassen konnte.

Nicht zuletzt möchte ich mich bei meinem Freund Markus Triska für sein reges Interesse an diesem Thema bedanken und für die Empfehlung einiger Computerprogramme, die mir ein angenehmeres und effizienteres Arbeiten ermöglicht haben.

Kurzfassung

Neuronale Netze finden in immer mehr Bereichen der Forschung Einzug. Im Rahmen dieser Arbeit soll einerseits die Theorie hinter den neuronalen Netzen erläutert und andererseits die Anwendung für die Prognose anhand eines praktischen Beispiels, der Prognose von Fernsehreichweiten, gezeigt werden.

Der erste Abschnitt beschäftigt sich mit dem grundlegenden Aufbau neuronaler Netze. Es werden verschiedene Lernverfahren beschrieben und dabei auftretende Probleme, sowie Möglichkeiten diese zu umgehen aufgezeigt. Dies geschieht allerdings – in Hinblick auf den praktischen Teil der Arbeit – mit einem Schwerpunkt auf den Einsatz neuronaler Netze zur Prognose. Weiters wird ein kurzer Einblick in die Begriffswelt der Fernsehforschung gegeben.

Im zweiten Abschnitt wird mit Hilfe des Neuronalen Netzwerksimulators JavaNNS versucht für die Fernsehsender ORF, ORF1, ORF2, ATV, RTL, SAT1, PRO7, RTL2, VOX, SRTL, KAB1, GoTV, MTV und PulsTV Netze zu trainieren, die in der Lage sind Reichweiten von Werbespots, in Abhängigkeit von verschiedenen Inputprädikatoren (Wochentag, Temperatur, Monat, ...) vorherzusagen.

Inhaltsverzeichnis

1 Das biologische neuronale Netz **4**

2 Aufbau künstlicher neuronaler Netze **4**

3 Einsatzmöglichkeiten von neuronalen Netzen **7**
3.1 Einsatz von neuronalen Netzen für Prognosen 8

4 Das Lernen im neuronalen Netz **10**
4.1 Der Back-propagation Algorithmus 11
4.1.1 Die Delta Regel 11
4.1.2 Arbeitsweise 12
4.1.3 Mögliche Probleme 13
4.2 Back-propagation mit Momentum 16
4.3 Quickpropagation 16
4.4 Rpropagation 18
4.5 QRpropagation 21
4.6 Cascade Learning Algorithmus 21
4.7 Genetische Algorithmen 24

5 Preprocessing **26**

6 Probleme bei der Prognose mit neuronalen Netzen **27**
6.1 Stopped Training 28
6.2 Addition von Rauschen zum Input 29
6.3 Die Penalty Verfahren 29
6.3.1 Weight decay 29
6.3.2 Weight elimination 30
6.3.3 Kostenfunktionen für die Gewichte von hidden-units . . . 31
6.3.4 Kostenfunktion für die Ausgaben von hidden-units 31
6.4 Pruning Techniken 33
6.4.1 Löschen der betragsmäßig kleinsten Gewichte 33
6.4.2 Optimal brain damage 33
6.4.3 Optimal brain surgeon 35
6.4.4 Statistische Gewichtsausdünnung 36
6.4.5 Hidden-unit pruning 37
6.4.6 Input-unit pruning 38
6.4.7 Skelettierung 38

7 Auswahl der Inputprädikatoren **39**
7.1 Expert Council Topologien 40
7.2 Selbstorganisierende Netze 40
7.3 Bestimmung der Prognoseunsicherheit 41
7.4 Resampling Methoden 42
7.4.1 Bootstrap-Algorithmus 42
7.4.2 Jackknife-Ansatz 42

8 Gütekriterien für die Prognose **43**

9 Die Sensitivitätsanalyse **47**

10 Informationskriterien **47**

11 Selektionskriterien **48**

12 Begriffe aus der Medienforschung **49**

13 Simulatoren für neuronale Netze **51**
13.1 Freie Software 51
13.1.1 Joone-Java Objected Oriented Neural Engine 51
13.1.2 SNNS 51
13.2 Proprietäre Software 52
13.2.1 EasyNN-plus 52
13.2.2 STATISTICA Neural Networks 52
13.2.3 NeuroSolutions 52
13.2.4 NeuroShell Predictor 53
13.2.5 BrainMaker 53
13.2.6 NeuroIntelligence 53

14 Bisherige Prognosen bei Fernsehnutzung **53**
14.1 Modelle mit gewichteten Aggregatmittelwerten 54
14.2 Lineare Modelle 54
14.3 Nichtlineare univariate Verteilungs- und Saisonmodelle 56
14.4 Nichtlinare multivariate Modelle 57
14.5 Tree-Modelle 60
14.6 Hybride Modelle 61

15 Beispiel: Prognose von Fernsehreichweiten mittels neuronaler Netze **63**
15.1 Einleitung 63
15.2 Ergebnisse 65
15.2.1 ORF1, ORF2 und ORF 65
15.2.2 ATV 81
15.2.3 RTL 83
15.2.4 SAT1 85
15.2.5 PRO7 87
15.2.6 RTL2 89
15.2.7 VOX 90
15.2.8 SRTL 92
15.2.9 KAB1 94
15.2.10 GoTV 95
15.2.11 MTV 97
15.2.12 PulsTV 98
15.3 Zusammenfassung 99

Abbildungsverzeichnis **101**

Tabellenverzeichnis **103**

Literatur **104**

Internetquellen **107**

1 Das biologische neuronale Netz

Der Begriff "neuronales Netz" stammt ursprünglich aus der Biologie – man spricht auch von "biologischen neuronalen Netzen" – und beschreibt unser Nervensystem.

Das Nervensystem ist aus einer Vielzahl von Neuronen (geschätzte 10^{11}) aufgebaut. Ein Neuron besteht aus einem Zellkörper (*soma*) mit einem Zellkern (*nucleus*), einer Nervenfaser (*axon*) und mehreren verzweigten Fortsätzen (*dendriten*). Die einzelnen Neuronen sind über sogenannte Synapsen (geschätzte 10^{14}) miteinander verbunden – man unterscheidet zwei Arten: erregende (exzitatorische) Synapsen und hemmende (inhibitorische) Synapsen. [vgl. [Zel94a]]

Die Informationsübertragung zwischen den Neuronen erfolgt an den Synapsen auf chemischem Weg mittels Aktionspotentialen (*spikes*) (es sind auch elektrische Synapsen möglich, die jedoch weniger häufig auftreten). Ein Aktionspotential wird dann ausgelöst, wenn das Ruhepotential (typisch sind in etwa $-70\,\mathrm{mV}$) einen gewissen Schwellwert überschritten hat. Dabei sind keine Zwischenzustände möglich – entweder ein Potential wird ausgelöst oder nicht ("Alles-oder-nichts Prinzip"). Erregende Synapsen erhöhen dabei die Wahrscheinlichkeit, dass ein Aktionspotential ausgelöst wird, hemmende Synapsen verringern die Wahrscheinlichkeit. Die *dendriten* dienen zur Weiterleitung der Information von den Synapsen zum Zellkörper. Der Zellkörper enthält neben dem Zellkern eine Vielzahl von Organellen, die für den Zellstoffwechsel benötigt werden. Das *axon*, schließt am Axonhügel an den Zellkörper an und leitet die Information vom Zellkörper zu den Synapsen, wo sie wiederum an ein angrenzendes Neuron weitergeleitet werden. Die Informationsübertragung kann dabei immer nur in eine Richtung erfolgen. [vgl. u. a. [SHG90], [NKK94], [Zel94a]]

Künstliche neuronale Netze sind keineswegs ein exaktes Abbild der biologischen neuronalen Netze – sie stellen viel mehr eine grobe Verallgemeinerung dar, die sich an dem biologischen Vorbild orientiert. Der Aufbau eines künstlichen neuronalen Netzes wird im nächsten Kapitel genauer beschrieben.

2 Aufbau künstlicher neuronaler Netze

Ein künstliches neuronales Netz besteht aus einer Vielzahl von einfachen Verarbeitungselementen, die sich mittels einer Verbindungsstruktur gegenseitig beeinflussen können. Die Verarbeitungselemente werden *units*, Knoten oder Neuronen (in Anlehnung an das biologische Vorbild) genannt – man unterscheidet, sofern es sich um ein geschichtetes Netz handelt, *input-, output-*, und *hidden-units*, die sich zu einem sogenannten *input-layer*, der die Eingabesignale erfasst, einem oder mehreren *hidden-layern*, und einem *output-layer*, der die Ausgabewerte liefert, zusammenfassen lassen. [vgl. [Mec94]]

Unter einem geschichteten Netz versteht man eine Struktur, bei der die *units* zu Neuronenschichten (*layern*) zusammengefasst werden. Innerhalb einer Schicht bestehen in den meisten Fällen keine Verbindungen zwischen den *units* – sie sind lediglich mit Neuronen der vorgelagerten bzw. nachfolgenden *layer* verbunden. Eine Ausnahme bilden beispielsweise selbstorganisierende Netze (siehe

dazu Kapitel 7.2), die auch Verbindungen innerhalb einer Schicht aufweisen. [vgl. [Mec94], [Zel94a], [Web00]]

Im Gegensatz dazu gibt es ungeschichtete Netze, bei denen sämtliche *units* untereinander verbunden sind. Hier werden die Inputdaten an alle Neuronen angelegt und der *output* ebenfalls an allen Neuronen abgelesen – man bezeichnet diese, oft für die Mustererkennung eingesetzten Netze, als Hopfield-Netze. In der Regel werden jedoch geschichtete Netze verwendet, weswegen in der weiteren Arbeit auch nur solche besprochen werden. [vgl. [Mec94], [Zel94a], [NKK94]]

Abgesehen von der Unterscheidung nach der Struktur des Netzes kann man auch eine Unterscheidung bezüglich der Ausbreitungsrichtung der Informationen innerhalb des Netzes treffen. In einem *feed-forward*-Netz fließt die Information, ausgehend vom *input-layer* zum *output-layer* immer in eine Richtung. Dabei können auch sogenannte lineare Konnektoren (*shortcut connections*) zum Einsatz kommen – darunter versteht man direkte Verbindungen zwischen zwei nicht unmittelbar aufeinanderfolgenden *layern* (z. B. Verbindungen zwischen *input-units* und *output-units* bei Vorhandensein von *hidden-layern*). Mittels *shortcut connections* können lineare Zusammenhänge direkt erfasst werden. Eine andere Variante sind *feed-back*-Netze, in denen Rückkoppelungen eingebaut sind – die Information fließt also in beide Richtungen. [vgl. [Mec94], [Web00]]

Die Veränderungen innerhalb des Netzes können durch drei Funktionen beschrieben werden [vgl. u. a. [Mec94], [Zel94a]]:

1. Propagierungsfunktion
2. Aktivierungsfunktion
3. Ausgabefunktion

Die Propagierungsfunktion *net* berechnet, aus den Ausgaben der vorgeschalteten Einheiten o_i und den jeweiligen Verbindungsgewichten w_{ij}, die Netzeingabe für eine *unit*. In den meisten Fällen wird hier die Summierungsfunktion angewendet.

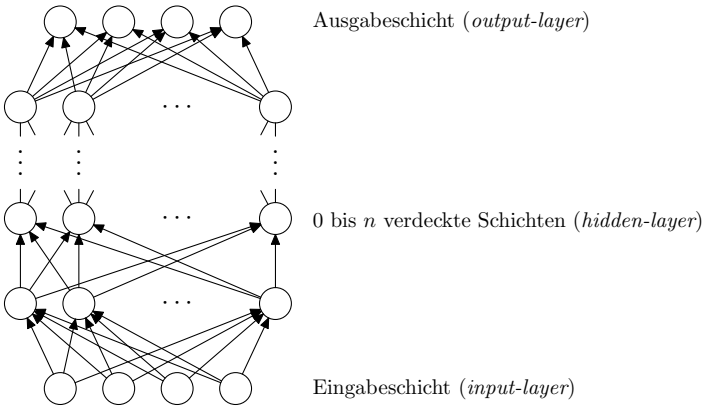


Abbildung 1: Mehrschichtiges *feed-forward*-Netz

$$net_j = \sum_i w_{ij} o_i \tag{1}$$

Die Aktivierungsfunktion F bestimmt, abhängig von der vorangegangenen Aktivierung a_j^{alt}, von der Netzeingabe net_j und von einer möglichen externen Eingabe (Schwellenwert, *bias*) ex_j, den aktuellen Aktivierungszustand a_j für die einzelnen *units*. Es gibt eine Reihe von möglichen Funktionen, die hier zum Einsatz kommen können – für welche Funktion man sich letztendlich entscheidet ist modellabhängig.

$$a_j = F(a_j^{\text{alt}}, net_j, ex_j) \tag{2}$$

Die Spezialisierung von Gleichung 2, bei der Verwendung eines Schwellenwerts lautet daher:

$$a_j = F(a_j^{\text{alt}}, net_j - ex_j) \tag{3}$$

Wird der Schwellenwert in der *unit* berücksichtigt, so muss er beim Lernen mittrainiert werden.

Eine andere Möglichkeit, einen *bias* einzusetzen, ist die Verwendung eines *on*-Neurons. Das *on*-Neuron, das immer den Wert Eins liefert, wird mit allen *units*, die Schwellenwerte besitzen verbunden. Die negative Gewichtung der Verbindung zwischen dem *on*-Neuron und der *unit* entspricht dann einem *bias*.

Formel 1 ändert sich zu:

$$net_j = \left(\sum_i w_{ij} o_i \right) - ex_j \tag{4}$$

und für die Aktivierungsfunktion gilt:

$$a_j = F(a_j^{\text{alt}}, net_j) \tag{5}$$

In diesem Fall muss die Lernregel den Sonderfall des *bias* nicht berücksichtigen. Allerdings ist durch die erhöhte Anzahl an Gewichten die Implementierung aufwendiger.

Die am häufigsten eingesetzten Aktivierungsfunktionen sind Sigmoide – einerseits die logistische Aktivierungsfunktion:

$$a_j = \frac{1}{1 + e^{-x}} \tag{6}$$

und andererseits die Tangens-Hyperbolicus Funktion:

$$a_j = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7}$$

Die Verwendung einer solchen stetigen, nichtlinearen Aktivierungsfunktion gewährleistet, dass die Fehlerfunktion nach ihren Gewichten differenzierbar ist. Sowohl die logistische als auch die Tangens-Hyperbolicus Funktion haben einen ausgeprägten linearen Bereich um ihren Schwellenwert – dadurch können lineare und nichtlineare Zusammenhänge in einer Funktion dargestellt werden.

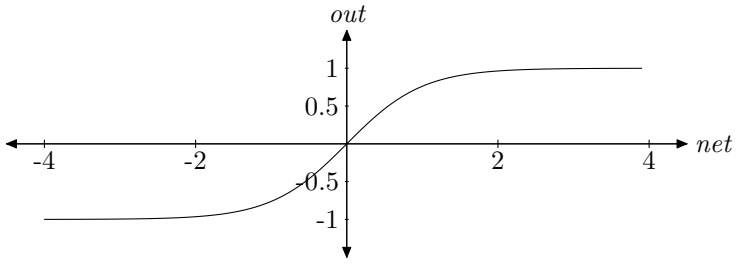


Abbildung 2: Tangens-Hyperbolicus tanh(x)

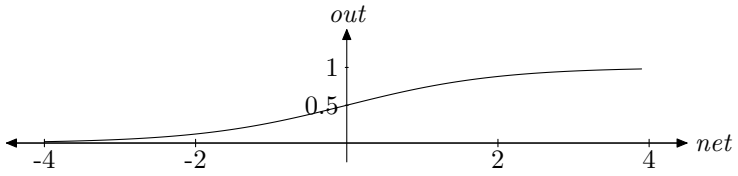


Abbildung 3: Logistische Aktivierungsfunktion

Während lineare Zusammenhänge durch kleine Gewichte charakterisiert werden, charakterisieren große Gewichte nichtlineare Zusammenhänge. Der Wertebereich der logistischen Funktion ist das offene Intervall $]0, 1[$. Die Tangens-Hyperbolicus Funktion ist im Intervall $]-1, 1[$ definiert.

Einer weitere Aktivierungsfunktionen ist die lineare Schwellenwertfunktion, die jedoch nur die Werte Null und Eins annehmen kann und daher für die Prognose diskreter Werte nicht geeignet ist.

Die Ausgabefunktion f bestimmt aus der aktuellen Aktivierung a_j die Ausgabe der *unit*.

$$o_j = f(a_j) \tag{8}$$

Da meistens bereits bei der Aktivierungsfunktion eine nichtlineare Funktion zum Einsatz kommt, wird als Ausgabefunktion üblicherweise eine Identitätsfunktion verwendet. Es kann jedoch auch eine lineare Funktion als Aktivierungsfunktion eingesetzt werden und eine nichtlinare Funktion, wie die beschriebenen Sigmoiden (Gleichungen 6 und 7), als Ausgabefunktion verwendet werden.

3 Einsatzmöglichkeiten von neuronalen Netzen

Der Ursprung der künstlichen neuronalen Netze geht auf das Jahr 1943 zurück, als die beiden Forscher W. McCulloch und W. Pitts in ihrem Aufsatz *"A logical calculus of the ideas immanent in nervous activity"* erstmals die Funktionsweise der künstlichen neuronalen Netze vorstellten. Die weitere Entwicklung wurde von vielen verschiedenen Bereichen der Forschung beeinflusst. Da künstliche neuronale Netze auf biologischen Systemen basieren, beschäftigten sich zu Beginn vor allem die Neurophysiologen und Psychologen mit dem Thema. Heute

werden neuronale Netze in der Mathematik, der Medizin, der Informatik, der Elektrotechnik, der Biologie und der Psychologie angewendet. [vgl. [Luk05], [Zel94a]]

Der Einsatz von neuronalen Netzen für die Prognose ist vor allem dann von Interesse, wenn viele Einflussfaktoren vorhanden sind, die auch nichtlineare Zusammenhänge aufweisen können. Weiters erlauben neuronale Netze die Verwendung sowohl qualitativer als auch quantitativer Inputdaten [vgl. [Web00]]. Im folgenden sollen einige häufige Anwendungen genannt werden:

- Überwachung
- Erkennung
- Optimierung
- Vorhersagen und Prognosen
- Analysen
- Steuerung und Regelung

Zur Erkennung werden neuronale Netze in vielen verschiedenen Bereichen eingesetzt: beispielsweise in der Kommunikationstechnik zur Spracherkennung, beim Militär zur Objekterkennung oder im Bereich von Versicherungen und Banken zur Unterschriftserkennung. Im Bereich der Finanzwirtschaft kommen neuronale Netze unter anderem auch für Aktienkursprognosen und Kreditrisikoabschätzungen zum Einsatz. In der Industrie dienen sie zur Produktionsoptimierung und zur Robotersteuerung. Den Einsatz für Prognosen macht sich die Meterologie für Wettervorhersagen zu Nutze. [vgl. [SHG90]] .

Auch wurden in der Vergangenheit bereits Versuche zur Prognose der Fernsehnutzung mittels neuronalen Netzen durchgeführt. Eine nähere Erläuterung dazu findet man im Kapitel 14.

3.1 Einsatz von neuronalen Netzen für Prognosen

Der Einsatz von neuronalen Netzen für die Prognose hat sich in der Vergangenheit vor allem im Bereich der Finanzwirtschaft bewährt. Eine typische Anwendung ist die Prognose von Aktienkursen. Die folgenden Ausführungen sind im Detail in [Pod94] zu finden.

Es gibt eine Reihe von möglichen Unterscheidungen, die bei der Erstellung eines Prognosemodells zu beachten sind.

Grundsätzlich kann man bei der Bildung eines Prognosemodells für die Vorhersage von Finanzmarktdaten zwei Ansätze unterscheiden: die technische Analyse und die Fundamentalanalyse. Bei der technischen Analyse werden als Inputprädikatoren Daten verwendet, die sich direkt aus der vergangenen Kursentwicklung ergeben. Bei der Fundamentalanalyse werden neben der Kursentwicklung auch allgemein zugängliche Informationen, wie beispielsweise die zeitliche Entwicklung von gesamtwirtschaftlichen Größen miteinbezogen.

Bezüglich des Informationsgehaltes, der durch den Kurs zum Ausdruck kommt unterscheidet man:

- Strenge Informationseffizienz: der Kurs enthält sämtliche Informationen, die verfügbar sind (auch Insiderinformation)
- Halbstrenge Informationseffizienz: der Kurs spiegelt alle öffentlich zugänglichen Informationen wieder
- Schwache Informationseffizienz: der Kurs gibt nur die historische Kursentwicklung wieder

Es stellt sich nun die Frage, ob die Prognose von Finanztiteln überhaupt möglich ist. Würde man von einer halbstarken Informationseffizienz ausgehen, so hätte eine Prognose wenig Sinn, da der derzeitige Kurs alle relevanten Informationen bereits enthält. Die technische Analyse hätte selbst unter der Voraussetzung einer schwachen Informationseffizienz keine sinnvolle Anwendung.

Eine Einteilung der Prognosearten kann auch nach dem zeitlichen Aspekt erfolgen: technische Analysen werden eher für kurzfristige Prognosen eingesetzt, während Fundamentalanalysen für mittel- und langfristige Prognosen verwendet werden. Der Grund dafür ist, dass gewisse Informationen, die für die fundamentale Analyse von Interesse sind, im kurzfristigen Bereich nicht zur Verfügung stehen, sondern nur auf Monatsebene oder Quartalsebene erhältlich sind.

Weiters unterscheidet man drei Arten der Prognose:

- die Richtungsprognose (auch Trendprognose), die eine Aussage darüber trifft, ob der Kurs steigen oder fallen wird
- die Punktprognose, deren Ziel die Vorhersage des tatsächlichen Kurses ist
- und die Punktprognose der Veränderung, die die Differenzreihe eines Finanztitels prognostiziert

Beim Prognosemodell selbst gibt es die Unterscheidung von bedingten und unbedingten Prognosemodellen. Beim bedingten Prognosemodell ist die zu prognostizierende Größe zum Zeitpunkt (t) abhängig von den erklärenden Variablen, ebenfalls zum Zeitpunkt (t). Daher müssen, bevor die eigentliche Variable prognostiziert werden kann, alle unabhängigen Größen bestimmt werden. Der Nachteil dabei ist, dass sich kleine Fehler bei der Vorhersage der unabhängigen Größen bei der Prognose des tatsächlich gesuchten Wertes negativ auswirken können. Diese Art des Prognosemodells eignet sich jedoch gut für sogenannte *case scenarios*.

Das unbedingte Prognosemodell verlangt keine Aussagen über die unabhängigen Variablen in der Zukunft, da sie anders als beim bedingten Modell zeitversetzt berücksichtigt werden. Der Prognosehorizont richtet sich dabei nach dem kleinsten vorkommenden *lag*. In den meisten Fällen ist das unbedingte dem bedingten Prognosemodell vorzuziehen.

Die erste bekannte Arbeit auf dem Gebiet der Aktienkursprognose stammt von White (1988), der die Veränderung der IBM-Aktie zum vergangenen Tag prognostizierte. Er wählte dafür den technischen Analyseansatz und verwendete als Inputprädikatoren die letzten fünf täglichen Differenzen.

4 Das Lernen im neuronalen Netz

Die wichtigste Eigenschaften von neuronalen Netzen ist ihre Fähigkeit zu lernen. Das Ziel des Lernvorgangs ist es, das Netz so zu trainieren, dass es für "unbekannte" Eingaben "adäquate" Ausgaben liefert. [vgl. [NKK94]]

Der Lernprozeß kann durch folgende Mittel umgesetzt werden [vgl. [Zel94a]]:

- Entwickeln neuer Verbindungen
- Löschen existierender Verbindungen
- Modifikation der Gewichte
- Modifikation der Propagierungs-, Aktivierungs- oder Ausgabefunktion
- Modifikation des Schwellenwertes von *units*
- Entwickeln neuer *units*
- Löschen existierender *units*

Von diesen Möglichkeiten werden die drei erstgenannten am häufigsten eingesetzt.

Grundsätzlich unterscheidet man drei verschiedene Arten des Lernens:

1. das überwachte Lernen (*supervised learning*),
2. das unüberwachte Lernen (*unsupervised learning*) und
3. das bestärkende Lernen (*reinforcement learning*).

Beim überwachten Lernen wird das Netz mit Paaren von Eingabe- und Ausgabedaten trainiert. Die Eingabedaten werden in das Netz, das zu Beginn mit kleinen, zufällig gewählten Gewichten belegt ist, eingespeist. Es wird eine Ausgabe erzeugt, die mit dem bekannten Ausgabewert verglichen wird. Die Gewichte werden nun mit Hilfe eines Lernalgorithmus angepasst. Das Ziel ist, dass das Netz nach einem ausreichend langem Training in der Lage ist zu generalisieren – das heißt auch für unbekannte Eingaben entsprechende Ausgaben zu finden. [vgl. u. a. [Zel94a], [Roj93], [NKK94]]

Das unüberwachte Lernen wird dann eingesetzt, wenn lediglich Eingabedaten bekannt sind. Das Netz versucht die vorgegebenen Daten in ähnliche Kategorien (*cluster*) zu unterteilen – die tatsächlichen Ausgabewerte sind dabei nicht von Interesse. Es soll erreicht werden, dass ähnliche Eingaben zu ähnlichen Ausgaben führen. Für die Prognose selbst ist das Verfahren des *unsupervised learning* folglich nicht geeignet. Es kann jedoch für die anfängliche Auswahl der für die Prognose verwendeten Prädikatoren eingesetzt werden (siehe dazu Kapitel 7). [vgl. [Zel94a], [Roj93]]

Das bestärkende Lernen kann als eine Form des überwachten Lernens betrachtet werden. Es unterscheidet sich darin, dass keine genaue Fehlergröße ermittelt wird, sondern nur eine Überprüfung stattfindet, ob die Ausgabe, die das Netz liefert, wahr oder falsch ist. Die Gewichte werden anhand dieser Information angepasst. [vgl. [Roj93]]

4.1 Der Back-propagation Algorithmus

Der *back-propagation* Algorithmus ist eine der derzeit am weitesten verbreiteten und am häufigsten eingesetzten Lernmethoden für neuronale Netze. Sie wurde erstmals 1974 von P. Werbos in seiner Dissertation *"Beyond regression: new tools for prediction and analysis in the behavioral sciences"* vorgestellt. Seinen heutigen Bekanntheitsgrad erreichte der *back-propagation* Algorithmus jedoch erst durch die Veröffentlichung des Artikels *"Learning internal representations by error propagating"* [RHW86] von D. E. Rumelhart, G. E. Hinton und R. J. Williams im Jahr 1986. Da der *back-propagation* Algorithmus zur Gruppe der "überwachten Lernverfahren" gehört, das heißt mit Eingabe- und Ausgabewerten arbeitet, ist er auch für die Prognose gut geeignet. Im Grunde stellt er eine Erweiterung der Delta Regel auf mehrschichtige neuronale Netze dar, weswegen er auch als "verallgemeinerte Delta Regel" bezeichnet wird. [vgl. u. a. [Zel94a]]

4.1.1 Die Delta Regel

Da der *back-propagation* Algorithmus im Grunde eine Erweiterung der Delta Regel auf mehrschichtige neuronale Netze darstellt (daher auch die Bezeichnung "verallgemeinerte Delta Regel"), soll an dieser Stelle, die Delta Regel kurz erläutert werden. [vgl. [NKK94]]

Die Delta Regel ist auch unter den Namen Widrow-Hoff Regel, Adaline Regel oder *least mean squares rule* in der Literatur zu finden [vgl. [Bis95a]]. Sie baut auf der Hebbschen Lernregel auf, die besagt:*"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing i, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B is increased."* [Heb49]

Die einfachste Form des Hebb'schen Lernens sieht wie folgt aus:

$$\Delta w_{ij} = \eta o_i a_j \tag{9}$$

η wird als Lernrate bezeichnet, o_i ist der *output* der Vorgängerzelle i und a_j gibt die Aktivierung der Nachfolgerzelle j an.

Die Delta Regel lautet schließlich:

$$\Delta w_{ij} = \eta o_i \delta_j \tag{10}$$

wobei gilt:

$$\delta_j = t_j - a_j \tag{11}$$

Die Delta Regel besagt, dass die Gewichtsänderung proportional zur Differenz δ_j der aktuellen Aktivierung a_j und der erwarteten Aktivierung t_j ist. Es ist zu beachten, dass die Regel nur bei linearen Netzwerken ohne *hidden-layer* anzuwenden ist. Für Netze mit sigmoiden Aktivierungsfunktionen und mehreren Schichten ist der *back-propagation* Algorithmus zu verwenden.

4.1.2 Arbeitsweise

Es handelt sich beim *back-propagation* Algorithmus um ein Gradientenabstiegsverfahren auf der Fehleroberfläche, das versucht jene Gewichtskombination zu finden, die den Berechnungsfehler minimiert. Bei Netzen, die mit dem *back-propagation* Algorithmus arbeiten, handelt es sich meistens um *feed-forward*-Netze die aus einem *input-layer*, einem *output-layer* und mindestens einem *hidden-layer* bestehen. Weiters muss der Gradient für alle Punkte des Gewichtsraums existieren – das heißt die partiellen Ableitungen der Fehlerfunktion nach den Gewichten müssen überall definiert sein, was wie bereits erwähnt durch den Einsatz einer stetigen Aktivierungsfunktion möglich ist. [vgl. [Roj93]]

Im Grunde arbeitet der *back-propagation* Algorithmus in zwei Schritten [vgl. [RHW86]]:

- *Feed-forward* Berechnung
- *Backward pass*

Bei der *feed-forward* Berechnung wird das Netz zunächst mit zufällig gewählten, kleinen Anfangsgewichten besetzt. Danach werden die Inputdaten angelegt und das Netz wird schichtweise bis zum *output-layer* durchlaufen. Es kann nun eine Fehlerbestimmung für jede *output-unit* vorgenommen werden, indem mit Hilfe einer Fehlerfunktion die Abweichung des *outputs* vom gewünschten *target* berechnet wird.

Der zweite Schritt, der *backward pass*, ist die entscheidende Phase. Hier wird das Netz schichtweise rekursiv, das heißt beginnend beim *output-layer*, über die *hidden-layer* zum *input-layer* durchlaufen. Dabei können sukzessive die Fehlersignale für jede einzelne *unit* berechnet werden – im Anschluss kann eine Modifikation der Gewichte vorgenommen werden, so dass der negativen Gradientenrichtung gefolgt wird und sich der Gesamtfehler minimiert.

Die Anpassung der Gewichte kann entweder nach dem Durchlauf jedes Paares von Eingabe- und Ausgabedaten erfolgen – in diesem Fall spricht man von einem *on-line-learning*, oder nach einem kompletten Durchlauf aller Trainingsmuster – dann handelt es sich um ein *batch-learning* oder *off-line-learning*. [vgl. [Web00], [Oss90], [Roj93]]

In der Anfangsphase des Trainings hat das *on-line-learning* den Vorteil, dass die Konvergenz beschleunigt wird, da jedes neu präsentierte Muster einen Schritt in Richtung des Minimums führt. Wenn das Training schon fortgeschritten ist und bereits eine gute Präsentation der Daten stattfindet, gewinnt das *batch-learning* an Bedeutung – das Zusammenspiel der Daten wird nun wichtiger. Es ist jedoch zu beachten, dass bei einer sehr großen Datenmenge das Verfahren stark verlangsamt wird. Das *on-line-learning* hat den Nachteil, dass die Reihenfolge der Musterpräsentation unter Umständen einen Einfluß auf das Ergebnis nehmen kann. In den meisten Fällen wird dem *batch-learning* der Vorzug gegeben. [vgl. [Oss90], [Web00], [Mec94]]

Als Fehlerfunktion wird häufig die Summe der quadrierten Abweichungen zwischen den vom Netz generierten *outputs* und den gewünschten Zielwerten verwendet:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{12}$$

t_{pj} steht für *target* und bezeichnet den gewünschten *output* des p-ten präsentierten Musters der j-ten *output-unit*. o_{pj} ist hingegen die tatsächliche, durch das Netz generierte Ausgabe, der j-ten *output-unit*, bei Präsentation des p-ten Musters.

Der Gesamtfehler des Netzes ergibt sich aus der Summe aller E_p:

$$E = \sum E_p \tag{13}$$

Die quadratische Zielfunktion hat den Vorteil, dass sie in jedem Punkt differenzierbar ist – das ist eine Vorraussetzung für das Lernen mit *back-propagation*. Die Multiplikation mit $\frac{1}{2}$ wird oft aus Gründen der einfacheren Berechnung verwendet, man kann jedoch gleich gute Ergebnisse erzielen, wenn man den Faktor nicht berücksichtigt.

Die Anpassung der Gewichte erfolgt nach folgender Regel, wobei w_{ij} das Gewicht der Verbindung zwischen den *units* i und j beschreibt:

$$w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta_p w_{ij} \tag{14}$$

Es gilt folgende Beziehung – dabei handelt es sich um die Delta Regel in einer abgewandelten Schreibweise (siehe Gleichung 10):

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \tag{15}$$

Die Lernrate (*learning rate*) η gibt dabei an, um welche Länge in Richtung des Gradienten herabgestiegen wird. Bei der Bestimmung von δ_{pj} muss unterschieden werden, ob es sich um eine *output-unit* oder eine *hidden-unit* handelt. Es gilt:

$$\delta_{pj} = \begin{cases} f'_j(net_{pj})(t_{pj} - o_{pj}) & \text{falls } j \text{ eine } \textit{output-unit} \text{ ist} \\ f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj} & \text{falls } j \text{ eine } \textit{hidden-unit} \text{ ist} \end{cases} \tag{16}$$

4.1.3 Mögliche Probleme

Die Verwendung des *back-propagation* Algorithmus bringt allerdings auch einige Probleme mit sich. Es kann nicht gewährleistet werden, dass tatsächlich das globale Minimum auf der Fehleroberfläche gefunden wird. Der Grund dafür liegt in der freien Wahl des Startpunktes (Gewichtsinitialisierung) – unterschiedliche Startpunkte können zu unterschiedlichen Minima führen. Liegt der Startpunkt in der Nähe des globalen Minimums, so wird das Gradientenabstiegsverfahren dorthin konvergieren. Andernfalls, sollte man in der Nähe eines suboptimalen Minimums mit dem Verfahren starten, wird man dieses erreichen. Je größer die Dimension des Netzes ist, das heißt, je mehr Verbindungen innerhalb des Netzes bestehen, umso mehr Zerklüftungen befinden sich in der Fehleroberfläche und damit steigt die Gefahr ein lokales Minimum an Stelle des gewünschten globalen Minimums zu erreichen. [vgl. [Zel94a]]

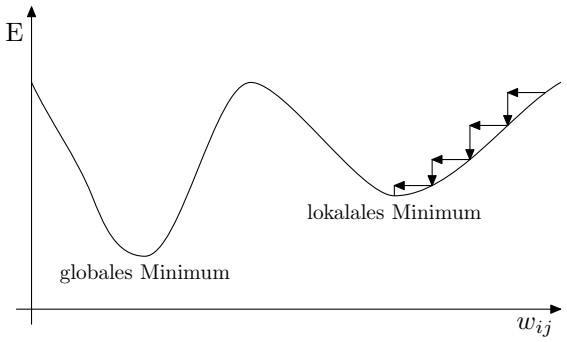


Abbildung 4: Konvergieren in ein lokales Minimum

Eine Möglichkeit mit der man die Wahrscheinlichkeit das globale Minimum zu finden erhöhen kann, ist die Verwendung eines Multi-Start-Verfahrens. Das bedeutet, das Netz mehrmals mit verschiedenen, zufällig gewählten Anfangsgewichten, also verschiedenen Startpunkten auf der Fehleroberfläche, zu trainieren. Dadurch, dass immer an einer anderen Stelle mit dem Gradientenabstieg begonnen wird, konvergiert das Verfahren in unterschiedliche Minima – die Wahrscheinlichkeit das globale Minimum zu finden steigt. [vgl. [Web00]]

Die Größe der Gewichtsänderung hängt stark vom Betrag des Gradienten ab. Befindet man sich in einem flachen Plateau, so ist dieser Betrag sehr klein und das Verfahren wird nur sehr langsam konvergieren. Das kann sogar soweit führen, dass es zu einer Stagnation kommt, da das Plateau für ein Minimum gehalten wird. Man nennt diesen Effekt auch *flat-spot problem*. Befindet man sich umgekehrt in einem sehr steilen Tal, so kann eine Oszillation auftreten – das heißt, der Gradient ist so groß, dass die Gewichtsänderung einen Sprung an die gegenüberliegende Seite des Tals bewirkt. Ist das Tal an dieser Stelle ebenso steil, so springt das Verfahren zurück zum Ausgangspunkt und der Vorgang beginnt von neuem. [vgl. [Zel94a]]

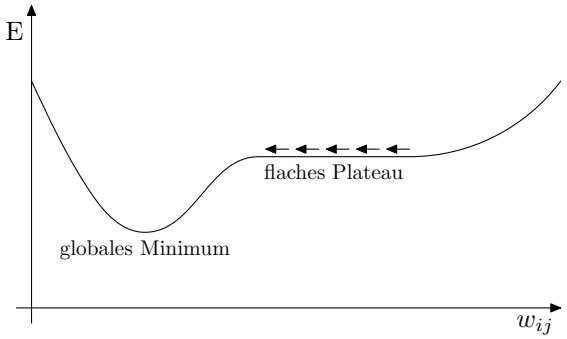


Abbildung 5: Auftreten des *flat-spot* Problems

Für das Konvergenzverhalten spielt vom allem die Lernrate η eine große Rolle. Ist η zu klein gewählt, kann dies zu einer sehr langen Trainingszeit führen und, die vorhin erwähnte Möglichkeit des Stagnierens in einem flachen Plateau,

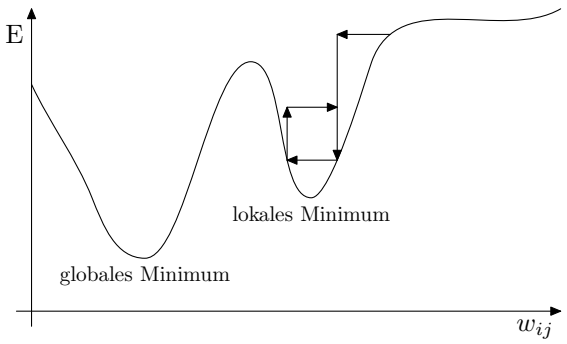


Abbildung 6: Auftreten von Oszillation

nimmt zu. Für große η besteht allerdings die Gefahr, dass sehr enge Täler, die aber möglicherweise das globale Minimum enthalten, übersprungen werden. Ein weiterer Nachteil ist, dass Oszillationen leichter auftreten können. Außerdem können große η dazu führen, dass die Umgebung eines Minimums wieder verlassen wird. [vgl. [Zel94a]]

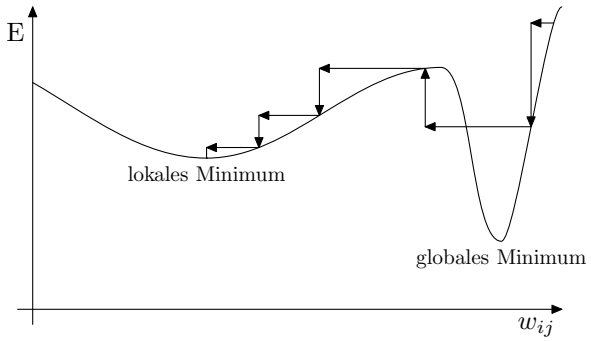


Abbildung 7: Überspringen des globalen Minimums

Grundsätzlich gilt, dass kleine Lernraten für das Erlernen von komplexen Daten und eine gute Generalisierung besser geeignet sind. In [Zel94a] wird empfohlen mit einer Lernrate von $\eta = 0{,}9$ zu beginnen – sollte damit nach einem ausreichend langem Training keine befriedigende Lerngüte erreicht werden, so empfiehlt es sich die Lernrate schrittweise um jeweils 0,1 herabzusetzen, bis das gewünschte Ergebnis erzielt ist.

Wichtig ist auch zu beachten, dass in jedem Trainingszyklus die Reihenfolge variiert, in der die Trainingsdaten präsentiert werden. Sonst besteht die Gefahr, dass das Netz die Reihenfolge der Daten lernt, was wiederum die Generalisierungsfähigkeit einschränken würde.

Bei ebenenweise vollständig verbundenen *feed-forward*-Netzen kann weiters das Problem des *symmetrie breaking* auftreten – nämlich dann, wenn allen Anfangsgewichten derselbe Wert zugewiesen wird (beispielsweise Null). In diesem Fall erhalten alle *hidden-units*, die dem *output-layer* vorgelagert sind dasselbe Fehlersignal, was zur Folge hat, dass die Verbindungen zwischen allen *hidden-*

units, die zur selben *output-unit* führen, stets gleich bleiben. Auch nach der Präsentation jedes weiteren Musters bleibt die Symmetrie der Gewichte erhalten. Es ist jedoch sehr einfach eine Lösung für dieses Problem zu finden: man kann zu Beginn das Netz mit zufälligen kleinen Gewichte belegen – in der Praxis werden die Gewichte meistens mit Werten zwischen -1 und 1 initialisiert. Das führt dazu, dass alle *units* eine Netzeingabe von ungefähr Null haben. Bei Verwendung einer logistischen Aktivierungsfunktion wird die Konvergenz zusätzlich beschleunigt, da im Bereich um Null, die Ableitung am größten ist. [vgl. [Zel94a], [RHW86]]

Es gibt verschiedene Lernverfahren, die einige der beschriebenen Nachteile des *back-propagation* Algorithmus aufheben sollen. Dazu zählen folgende:

- *Back-propagation* mit Momentum
- *Quickpropagation*
- *Resilient propagation*
- *QRpropagation*

4.2 Back-propagation mit Momentum

Beim *back-propagation* Verfahren mit Momentum (*conjugate gradient descent*), das auf G. E. Hinton und R. J. Williams zurückgeht (erste Veröffentlichung 1986 in *"Paralell Distributed Processing: Explorations in the Microstructure of Cognititon"*) [RHW86], wird durch die Einführung eines Momentumterms, zum aktuellen Zeitpunkt auch die zuletzt vorgenommene Gewichtskorrektur berücksichtigt.

$$\Delta w_{ij}(t+1) = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ij}(t) \tag{17}$$

$\Delta w_{ij}(t)$ gibt dabei die Gewichtsänderung im letzten Schritt an – sie wird mit α (Moment) gewichtet. Durch das Addieren des Momentumterms soll eine Beschleunigung des Trainings auf flachen Plateaus und ein Abbremsen in engen Tälern erreicht werden. Durch das Filtern hochfrequenter Änderungen erhält das Verfahren eine gewisse Trägheit. Es wird robuster und die Tendenz eine eingeschlagene Richtung einzuhalten steigt. Dadurch können flache Täler schneller überwunden werden und das Ziel wird im Allgemeinen in kürzerer Zeit erreicht. [vgl. [NKK94]]

Die Bestimmung von α und η ist problemspezifisch. Allgemein gilt jedoch, dass das Moment α größer als die Lernrate η gewählt werden soll – beispielsweise $\alpha = 0{,}9$ und $\eta = 0{,}05$. In den meisten Fällen liefern Werte zwischen 0,6 und 0,9 gute Ergebnisse. [vgl. u. a. [NKK94], [Zel94a], [RHW86], [Roj93]]

4.3 Quickpropagation

Eine andere Möglichkeit den Lernvorgang zu beschleunigen, ist die Verwendung von Informationen über die Krümmung der Fehlerkurve. S. Fahlmann hat 1989

den *quickpropagation* Algorithmus, als Verbesserung des *back-propagation* Algorithmus in seiner Arbeit *"An empirical study of learning speed in back-propagation networks"* [Fah88] vorgestellt. Der *qickpropagation* Algorithmus basiert auf der Idee, dass die Fehlerfunktion annähernd durch eine nach oben offene Parabel beschrieben werden kann. Es wird nun versucht den Scheitelpunkt der Parabel und damit das Minimum der Fehlerfunktion in einem Schritt zu erreichen [vgl. u. a. [Zel94b], [Roj93]]. Dazu wird die partielle Ableitung der Fehlerfunktion nach dem betrachteten Gewicht zum aktuellen Zeitpunkt (t) und zum vergangenen Zeitpunkt $(t-1)$ berechnet. Mit $\Delta w_{ij}(t-1)$ ergibt sich die neue Gewichtsänderung zu:

$$\Delta w_{ij}(t) = \frac{S(t)}{S(t-1) - S(t)} \Delta w_{ij}(t-1) \tag{18}$$

mit

$$S(t) = \frac{\partial E}{\partial w_{ij}}(t) \tag{19}$$

bzw.

$$S(t-1) = \frac{\partial E}{\partial w_{ij}}(t-1) \tag{20}$$

Da die Parabel nur eine grobe Approximation der Fehlerfunktion darstellt, wird man das tatsächliche Minimum nicht in einem Schritt erreichen, sondern lediglich in dessen Nähe gelangen. Da das Verfahren jedoch iterativ angewendet wird, nähert man sich dem Minimum immer weiter an.

Um das Verfahren anwenden zu können müssen zwei Annahmen getroffen werden. Einerseits muss die Fehlerfunktion durch eine nach oben offene Parabel angenähert werden können und andererseits muss die Änderung eines Gewichts unbeeinflusst von den Änderungen der anderen Gewichte zur gleichen Zeit vorgenommen werden können.

Bei Anwendung von Gleichung 18 können verschiedene Fälle eintreten. Wenn die aktuelle Steigung $S(t)$ kleiner als die vorangehende $S(t-1)$ ist und dasselbe Vorzeichen hat, so erfolgt die Gewichtsänderung in die gleiche Richtung – die Größe der Gewichtsänderung ist abhängig von der Größe der vorangegangenen Änderung. Haben die Steigungen zum Zeitpunkt $(t-1)$ und (t) nicht dasselbe Vorzeichen, so wurde das Minimum übersprungen – in diesem Fall befindet man sich im nächsten Schritt auf einem Punkt, zwischen der aktuellen und der vorhergehenden Position. Für den Fall, dass $S(t)$ größer als $S(t-1)$ ist und das gleiche Vorzeichen hat, würde man sich rückwärts in Richtung eines Maximums bewegen.

Probleme können auch auftreten, wenn $S(t) \approx S(t-1)$ gilt – in diesem Fall würde die Schrittlänge aufgrund der Divison durch Null "unendlich groß". Um dieses Problem zu lösen kann ein maximaler Wachstumsfaktor μ (*maximum growth factor*) eingeführt werden, so dass gilt:

$$\mid \Delta w_{ij}(t) \mid \leq \mu \mid \Delta w_{ij}(t-1) \mid \tag{21}$$

Für den Fall, dass die Steigungen zweier aufeinanderfolgender Zeitpunkte gleich groß ist, oder ein Schritt rückwärts auf der Parabel eintreten würde, wird anstelle von Gleichung 18 das μ-fache der vorherigen Gewichtsänderung für den aktuellen Schritt verwendet. Gute Werte von μ liegen zwischen $1,75$ und $2,25$, wobei für die meisten Simulationen der Wert $1,75$ verwendet wird. [vgl. [Zel94a], [Fah88]]

Das Verfahren wird zu Beginn, oder nachdem eine Schrittlänge von Null eingetreten ist neu gestartet (*bootstrapping*). Dies geschieht mit Hilfe eines Gradientenabstiegverfahrens, wie beispielsweise *back-propagation*.

Fahlmann hat bei seinen Versuchen immer, mit Ausnahme des Überspringens eines Minimums, einen Gradiententerm zu der der von ihm definierten Gewichtsänderung addiert. Außerdem wird ein kleiner *weight decay* Term eingeführt, um zu verhindern, dass die Gewichte zu große Werte annehmen.

4.4 Rpropagation

Erstmals wurde dieses Verfahren 1992 in der Arbeit *"Rprop – A fast adaptive learning Algorithm"* [RB92] von M. Riedmiller und H. Braun vorgestellt. Beim Rprob (*resilient propagation*) handelt es sich um einen *adaptive step algorithm* [vgl. [PR94]] – das heißt es erfolgt eine direkte Anpassung der Gewichte aufgrund des Gradienten der Fehlerfunktion. Die Gewichtsänderung ist nun jedoch nicht mehr von der Größe der partiellen Ableitung abhängig. Stattdessen wird die Modifikation gemäß dem Vorzeichen des Gradienten geändert. [vgl. [RB92], [RH93]]

Die Gewichtsänderung wird in zwei Schritten durchgeführt. Zunächst wird der sogenannte Änderungsparameter (*update value*) bestimmt:

$$\Delta_{ij}^{(t)} = \begin{cases} \min(\eta^+ \Delta_{ij}^{(t-1)}, \Delta_{\max}) & , \ \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \max(\eta^- \Delta_{ij}^{(t-1)}, \Delta_{\min}) & , \ \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)} & , \ \text{sonst} \end{cases} \tag{22}$$

Dabei sind η^- und η^+ Konstanten, für die gilt: $0 < \eta^- < 1 < \eta^+$.

Im zweiten Schritt erfolgt die eigentliche Berechnung der Gewichtsänderung:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{23}$$

mit

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & , \ \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t)} & , \ \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ 0 & , \ \text{sonst} \end{cases} \tag{24}$$

Bei der *resilient propagation* geht man also davon aus, dass ein Wechsel des Vorzeichens gleichbedeutend mit dem Überspringen eines Minimums auf der Oberfläche der Fehlerfunktion ist. Daher wird in diesem Fall der *update value* Δ_{ij} um den Faktor η^- reduziert. Andernfalls, wenn das Vorzeichen unverändert bleibt, wird Δ_{ij} um den Faktor η^+ vergrößert und somit die Konver-

genz beschleunigt. Als gute Werte für η^- und η^+ haben sich 0,5 und 1,2 erwiesen. Für die maximale bzw. minimale Schrittweiten $\Delta_{\max}$ und $\Delta_{\min}$ sind 50 und 10^{-6} gute Richtwerte. Diese Beschränkung ist wichtig, um Problemen mit Gleitkommaüberlauf bzw. Gleitkommaunterlauf vorzubeugen. Zu Beginn werden sämtliche *update values* auf einen Anfangswert Δ_0 gesetzt – eine gute Wahl ist $\Delta_0 = 0,1$, kleinere und größere Werte können in manchen Fällen jedoch ebenfalls eine schnelle Konvergenz erzielen.

Sobald der *update value* Δ_{ij} nach Gleichung 22 bestimmt ist, kann die Gewichtsänderung Δw_{ij} gefunden werden. Ist die Ableitung des Fehlers nach den Gewichten kleiner Null, wird Δ_{ij} addiert, ist die Ableitung größer Null, wird Δ_{ij} subtrahiert.

Es gibt jedoch eine Ausnahme: wenn das Minimum übersprungen wurde, muss die vorangegangene Gewichtsänderung rückgängig gemacht werden:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)} \quad \text{falls} \quad \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \tag{25}$$

Zusätzlich wird in diesem Fall in Gleichung 22 $\frac{\partial E}{\partial w_{ij}}^{(t-1)} := 0$ gesetzt – sonst würde im folgenden Schritt wiederum eine Änderung des Vorzeichens eintreten und der *update value* somit erneut reduziert werden.

Die Abbildungen 8, 9, 10 und 11 zeigen die vier möglichen Fälle, die bei der *resilient propagation* eintreten können. Fall 4 folgert dabei automatisch aus Fall 3. [vgl. [Zel94a]]

Die Muster werden bei der *resilient propagation* im *batch-mode* präsentiert – das heißt die Anpassung der Gewichte erfolgt erst, nachdem alle Musterpaare einmal durch das Netz propagiert wurden. [vgl. [Zel94b]]

Im Allgemeinen wird mit dem *resilient propagation* Algorithmus das Lernen gegenüber dem *back-propagation* Algorithmus beschleunigt. Allerdings neigt die *resilient propagation* eher zu einem *overlearning*. Durch die Einführung eines *weight decay* Terms kann dieser Effekt gemindert werden. [vgl. [Zel94a]]

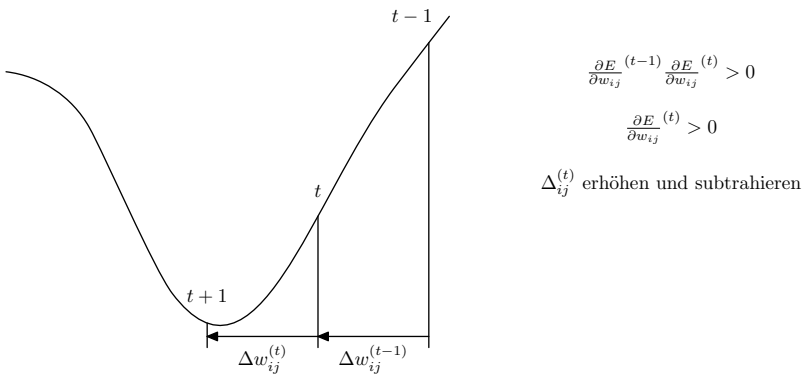
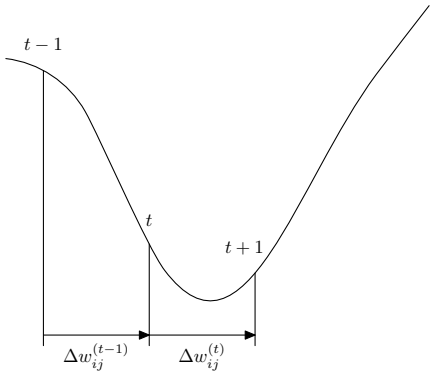


Abbildung 8: Rprop – Fall 1

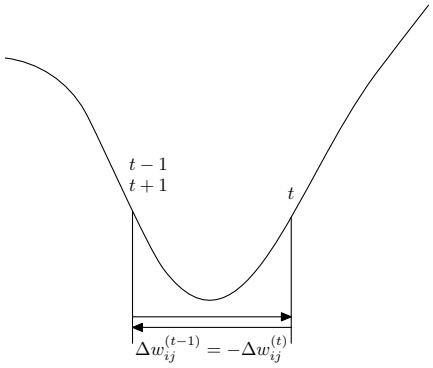


$\frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} > 0$

$\frac{\partial E}{\partial w_{ij}}^{(t)} < 0$

$\Delta_{ij}^{(t)}$ erhöhen und addieren

Abbildung 9: Rprop – Fall 2

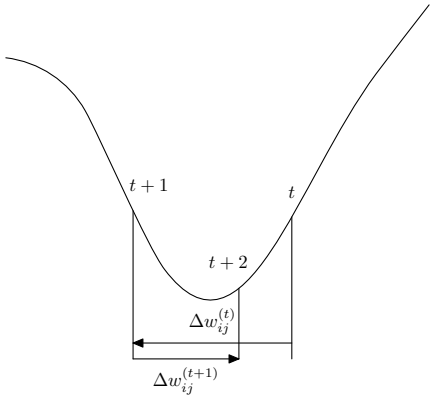


$\frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0$

$\frac{\partial E}{\partial w_{ij}}^{(t)} > 0$

$\Delta_{ij}^{(t)}$ verringern und
die letze Gewichtsänderung rückgängig machen

Abbildung 10: Rprop – Fall 3



$\frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} = 0$

$\frac{\partial E}{\partial w_{ij}}^{(t)} = 0$

$\Delta_{ij}^{(t)} = \Delta_{ij}^{(t-1)}$ wird addiert

Abbildung 11: Rprop – Fall 4

4.5 QRpropagation

QRprob ist eine Kombination der beiden zuletzt vorgestellten Verfahren – also eine Kombination von *quickpropagation* und *resilient propagation* und ist somit ein hybrider Lernalgorithmus. Dieses Verfahren arbeitet folgendermaßen: falls das Vorzeichen des Gradienten der Fehlerfunktion in zwei hintereinanderfolgenden Schritten gleich bleibt, so wird *resilient propagation* angewendet, was ein schnelles Konvergieren in Richtung des Minimums bewirkt. Andernfalls, wenn sich das Vorzeichen ändert, ist dies ein Zeichen dafür, dass ein Minimum übersprungen wurde und man verwendet *quickpropagation*. Es werden somit die Vorteile beider Verfahren miteinander verknüpft, was auch zu einem schnelleren Lernen gegenüber Rprop führt. Beim QRprop werden die Daten im *batch-mode* präsentiert. [vgl. [PR94]]

4.6 Cascade Learning Algorithmus

Der *cascade learning* Algorithmus ist kein reines Lernverfahren, sondern er dient gleichzeitig dem Aufbau einer geeigneten Netzwerktopologie. Man geht zu Beginn von einem minimalen Netz aus, das sukzessive um Neuronen erweitert wird.

Das Verfahren wurde 1990 von S. E. Fahlmann und C. Lebiere in ihrer Arbeit *"The Cascade-Correlation Learning Architecture"* [FL90] vorgestellt. Die *cascade-correlation* soll die Probleme, die der *back-propagation* Algorithmus aufwirft – das *flat-spot problem* und das *moving-target problem* – minimieren. Unter *flat-spot problem* versteht man das langsame Konvergieren des Gradientenabstiegsverfahren auf flachen Gebieten der Fehlerfunktion (siehe dazu Kapitel 4.1.3).

Das *moving-target problem* resultiert daraus, dass jedes Neuron versucht ein Detektor für ein Teilmuster zu werden, welches einen wichtigen Beitrag zur gesamten Aufgabe liefert. Dadurch, dass sich durch die gleichzeitige Anpassung aller Neuronen der Fehlervektor ständig ändert und die *hidden-units* untereinander nicht direkt kommunizieren können, wird die Anpassung für das einzelne Neuron erheblich erschwert. [vgl. [FL90], [Zel94a]]

Eine Ausprägung des *moving-target problem* ist der Herden-Effekt. Er tritt auf, wenn mehrere Teilprobleme gelöst werden sollen, von denen eines ein stärkeres Fehlersignal liefert als das andere. Zu Beginn werden alle Neuronen das Problem mit dem stärkeren Fehlersignal zu lösen versuchen und erst danach das andere Teilproblem betrachten. Dessen Lösung kann jedoch das erneute Auftreten des ersten Problems bewirken und so weiter. Im Normalfall kommt es letztendlich doch zu einer Aufspaltung der "Herde" und somit zum gleichzeitigen Lösen beider Probleme, doch durch die häufige Änderung der Teilprobleme kann sich das Finden einer Gesamtlösung stark verzögern. [vgl. [FL90], [Zel94a]]

Das Verfahren verläuft in folgenden Schritten [vgl. [Zel94b]]:

1. Der Algorithmus startet mit einem minimalen Netzwerk, das nur aus einem *input-layer* und einem *output-layer* besteht. Alle *units* sind miteinander verbunden. Es existiert außerdem ein *bias*-Neuron, oder *on*-Neuron, das immer die Ausgabe Eins liefert.

2. Das Netz wird nun mit einem Lernverfahren (es kann praktisch jedes Lernverfahren für *single-layer* Netze verwendet werden, beispielsweise die Delta Regel (siehe dazu Kapitel 4.1.1)) so lange trainiert, bis über mehrere Epochen keine signifikante Verbesserung des Fehlers mehr erreicht wird. Nun wird das Netz ein letztes Mal mit dem gesamten Datensatz durchlaufen um den Fehler zu bestimmen. Sollte man zu diesem Zeitpunkt bereits mit dem Ergebnis zufrieden sein, kann das Verfahren angehalten werden. Andernfalls existiert ein Restfehler, der reduziert werden soll. Man fährt also mit Schritt 3 fort.

3. Im diesem Schritt wird ein verdecktes Neuron (*candidate-unit*) eingefügt und mit sämtlichen *input-units* und bereits bestehenden *candidate-units* verbunden. Der Ausgang der *candidate-unit* hat vorerst keine Verbindung zum Netz.

4. Nun wird ein Training aller Verbindungen zur *candidate-unit* vorgenommen, mit dem Ziel, die Summe der Beträge der Korrelation[1] zwischen dem *output* der *candidate-unit* o_j und dem Restfehler e_k des Netzes zu maximieren.

   S ist dabei wie folgt definiert:

$$S = \sum_k \mid \sum_p (o_{pj} - \bar{o}_j)(e_{pk} - \bar{e}_k) \mid \tag{26}$$

   j gibt den Index der *candidate-unit* an, k den Laufindex über alle Ausgabeneuronen und p den Index der *pattern* (präsentierte Muster). $\bar{o}_j$ und $\bar{e}_k$ bezeichnen die mittleren Werte des *outputs* der j-ten *unit* beziehungsweise des Restfehlers der Ausgabe k über alle Muster p.

   Damit S maximiert wird, muss die partielle Ableitung nach jedem Gewicht w_{ij} gebildet werden:

$$\frac{\partial S}{\partial w_{ij}} = \sum_k \sum_p \sigma_k a'_j(net_{pj}) o_{pi}(e_{pk} - \bar{e}_k) \tag{27}$$

   mit σ_k als dem Vorzeichen der Korrelation zwischen der Ausgabe der *candidate-unit* j und dem Fehler der Ausgabezelle k für das *pattern* p:

$$\sigma_k = sgn\left(\sum_p (o_{pj} - \bar{o}_j)(e_{pk} - \bar{e}_k)\right) \tag{28}$$

   Sobald die Ableitungen nach allen Gewichten bestimmt sind, kann ein Gradientenaufstieg durchgeführt werden, um S zu maximieren. Er wird gestoppt sobald keine wesentlichen Veränderungen mehr erfolgen. Der

[1] Genaugenommen handelt es sich hier um die Kovarianz, da einige Normalisierungsterme vernachlässigt werden. Ursprünglich verwendeten S. E. Fahlman und C. Lebiere jedoch tatsächlich die Korrelation, woher auch der Name *cascade-correlation* stammt. Für viele Fälle liefert die Kovarianz bessere Ergebnisse.

Gradientenaufstieg kann wiederum durch die Delta Regel ausgeführt werden.

5. An diesem Punkt werden nun alle Gewichte, die zu der *candidate-unit* führen "eingefroren" und können im weiteren Verlauf nicht mehr geändert werden. Die *candidate-unit* wird jetzt mit allen *output-units* verbunden und es wird mit Schritt 2 fortgefahren.

Dieser Algorithmus wird so lange durchlaufen, bis der Gesamtfehler des Netzes einen festgesetzten Wert unterschreitet. [vgl. [FL90],[Zel94b]]

Abbildung 12 zeigt die Architektur einer *cascade-correlation*, vor dem Hinzufügen einer dritten *hidden-unit*. Die Quadrate stellen die Gewichte w_{ij} dar – die dunklen stehen dabei für die eingefrorenen, die hellen für noch nicht fertig trainierte Gewichte. Die Eingaben werden entlang der vertikalen Verbindungen aufsummiert. [vgl. [Zel94a]]

Es ist möglich eine Gruppe von *candidate-units* mit verschiedenen Startgewichten und unterschiedlichen Aktivierungsfunktionen gleichzeitig mit denselben Inputdaten zu trainieren (*pools* von möglichen Kandidaten). Das Training kann aufgrund der Unabhängigkeit der *units* voneinander parallel ablaufen. Jene *unit*, die die beste Korrelation aufweist, wird schließlich in das Netz integriert. Dadurch wird die Möglichkeit, dass ein Neuron permanent hinzugefügt wird, das in einem lokalen Minimum hängen geblieben ist, verringert. Außerdem kann das Lernverfahren damit beschleunigt werden. Fahlmann hat in seinen Untersuchungen *pools* von vier bis acht Kandidaten verwendet. [vgl. [FL90]]

Die bei der *cascade-correlation* entstehenden Netze sind sehr tief – besitzen also eine große Zahl an *hidden-layern*, die jedoch jeweils nur ein Neuron enthalten. Das Verfahren konvergiert schnell, da die einzelnen *units* getrennt voneinander trainiert werden und somit ihre Teilmuster ohne das Auftreten des Herden-Effekts finden. Es findet während des Trainings keine Rückpropagierung von Fehlern statt – die Signale werden nur in eine Richtung geleitet. [vgl. [FL90], [Zel94a]]

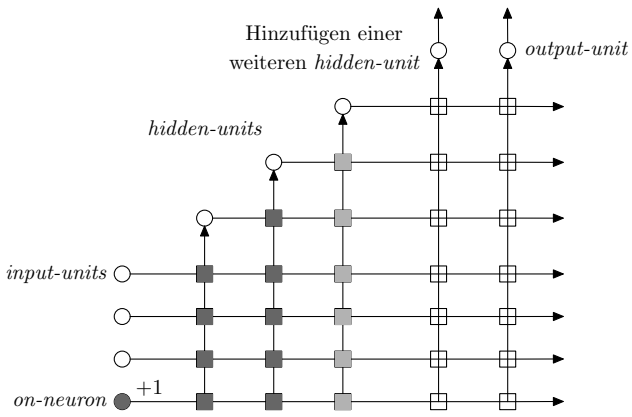


Abbildung 12: *Cascade-correlation* Architektur

Die *cascade-correlation* ist außerdem gut geeignet um bereits bestehende Netze mit neu gewonnenen Daten zu trainieren (inkrementelles Lernen). Bereits gelernte Strukturen werden nicht zerstört – sie können jedoch durch eine Änderung der Gewichte zum *output-layer* an Wichtigkeit verlieren. [vgl. [FL90], [Zel94a]]

4.7 Genetische Algorithmen

Die Aufgabe neuronaler Netze ist es, eine bestimmte Aufgabe so gut wie möglich zu lernen. Das Lernen ist ein Optimierungsprozess, dessen Ziel es ist die Fehlerfunktion zu minimieren – er kann auf verschiedenste Weise umgesetzt werden. Eine Möglichkeit ist die Verwendung Genetischer Algorithmen, die ein stochastisches Such- und Optimierungsverfahren darstellen. Genetische Algorithmen orientieren sich an den Verfahren der biologischen Evolution. [vgl. u. a. [Zel94a], [Roj93]]

Grundsätzlich gehören zu einem genetischen Algorithmus folgende Elemente [vgl. [Roj93]]:

1. Kodierung des Optimierungsproblems
2. Mutatationsoperator
3. Informationsaustauschoperatoren

Zu Beginn wird eine gewisse Menge an Individuen (mögliche Lösungen) bestimmt, die die erste Generation (erste Population) bilden. Da bei neuronalen Netzen in der Prognose meist reelle Zahlen als Inputdaten verwendet werden, ist eine anfängliche Kodierung in einen Binärcode nötig. Die Prädikatoren sind nun in Form eines *bitstreams* darstellbar.

Mittels genetischer Algorithmen können nicht nur die Gewichte eines neuronalen Netzes gefunden werden, sondern es ist auch möglich die Architektur und Struktur der Netze zu bestimmen. Folglich können die Individuen Informationen über die Netzgröße, die Gewichtskonfiguration, die Aktivierungsfunktionen usw. enthalten. Je mehr Eigenschaften des Netzes enthalten sind, umso mehr Kombinationen eines *bitstreams* sind möglich.

Nachdem sämtliche Individuen der ersten Generation trainiert worden sind, kann mittels einer Zielfunktion (Fitnessfunktion) für jedes Individuum eine Fitness bestimmt werden. Anhand der bestimmten Funktionswerte kann weiters eine Reihung vorgenommen werden. Bei einem Minimierungsproblem werden jene Individuen mit einem niedrigen Funktionswert vor jenen mit einem höheren Funktionswert gereiht.

Bei der Selektion wird nun aus allen möglichen Lösungen eine gewisse Anzahl ausgewählt, die weiter betrachtet werden soll. Es gibt verschiedene Möglichkeiten die Auswahl vorzunehmen. Es kann eine gewisse Anzahl, beispielsweise die Hälfte, der besten Individuen gewählt werden. Man kann aber auch eine zufällige Auswahl treffen und dabei jene Individuen, die besser gereiht sind mit einer höheren Wahrscheinlichkeit wählen. Entscheidet man sich für die erste Variante müssen die 50 % der eliminierten Individuen in der nächsten Generation

ersetzt werden (*refill*). Das kann durch einfaches Klonen der bereits ausgewählten Individuen oder durch eine Zufallsauswahl geschehen.

Bei der Rekombination sollen verschiedene Individuen so gemischt werden, dass eine neue Generation an Individuen entsteht. Eine bekannte Möglichkeit dies durchzuführen ist die Anwendung des *crossover*-Operators (siehe Abbildung 13). Es werden dabei zwei Ketten ausgewählt (die Eltern), die beide an derselben, zufällig bestimmten Stelle getrennt werden (*one-cut-swap*). Nun werden zwei neue Ketten gebildet, von denen die eine aus dem linken Teil des einen Elter und dem rechten Teil des anderen besteht und umgekehrt. Diese neuen Ketten werden auch als Kinder bezeichnet. Da beide Abschnitte wichtige Informationen beinhalten, kann im günstigen Fall das neue Individuum näher am Minimum liegen, als die Eltern aus denen es produziert wurde. Anstelle des *one-cut-swap* kann auch ein *two-cut-swap* vorgenommen werden, bei dem die Ketten an zwei Punkten getrennt werden. Da jedoch kaum Konvergenzunterschiede zu bemerken sind ist ein *one-cut-swap* im Normalfall ausreichend. [vgl. [Web00], [Roj93]]

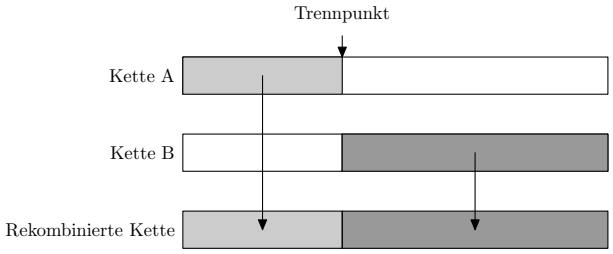


Abbildung 13: Beispiel für *crossover*

Bei der Mutation findet ein zufälliger Informationstausch innerhalb der Individuen statt – im Fall einer Binärcodierung geschieht das, indem an zufällig gewählten Stellen der Kette eine Null durch eine Eins ersetzt wird. Die Wahrscheinlichkeit mit der es zu einer Mutation kommt wird anhand eines Parameters, der Mutationsrate p festgelegt. Man unterscheidet zwei Arten der Mutation: die *random exchange*, bei der jeweils zwei einzelne *bits* getauscht werden und die *selection reversal*, bei der eine kleine Abfolge an *bits* vertauscht wird. Durch die Mutation soll ein zu schnelles Konvergieren in ein lokales Minimum verhindert werden. Sie mindert die Anzahl identischer Individuen und schafft auf diese Weise neue Startpunkte im Definitionsraum. Ausschlaggebend ist hier die Einstellung der Mutationsrate p – ist sie zu hoch ($p \approx 1$) eingestellt, so werden sämtliche Bitmuster systematisch zerstört. In den ersten Generationen sollte die Mutationsrate höher sein, damit ein weiter Bereich der Fehleroberfläche abgesucht wird. Bei späteren Generationen kann die Mutationsrate gesenkt werden, um eine bessere Anpassung an das Optimum zu erzielen. [vgl. [Web00], [Roj93]]

Nach der Anwendung der genetischen Operatoren (Mutationsoperator, *crossover*) entstehen neue Ketten, die nicht perfekte Kopien der Individuen der ersten Generation darstellen. Diese nächste Generation kann nun erneut getestet werden. Nach einiger Zeit wird man eine Konvergenz der Individuen feststellen –

das heißt einige Individuen werden aussterben und andere werden mit kleinen Abwandlungen immer wieder entstehen. Es ist jedoch auch möglich, dass Bitmuster aussterben, die im weiteren Verlauf des Verfahrens notwendig wären. Dieses Problem kann einerseits durch "Genetisches Driften" auftreten – Bitmuster die in einer Population verstärkt vertreten sind können sich durchsetzen, auch wenn ihre Fitness nicht überdurchschnittlich ist – und andererseits, wenn viele Individuen mit ähnlicher Anpassungsfähigkeit vorhanden sind. Um diese Probleme zu vermeiden ist es wichtig Mutation und *crossover* gut aufeinander abzustimmen. [vgl. [Roj93]]

Die Vorteile der Genetischen Algorithmen gegenüber anderen Optimierungsverfahren, wie zum Beispiel der Gradientenabstiegsmethode, lassen sich wie folgt zusammenfassen [vgl. [Roj93]]:

- Sie ermöglichen eine einfache Parallelisierung, da die Auswertungen der zu optimierenden Funktion in allen Punkten unabhängig sind.
- Sie benötigen keine Gradienteninformation – es ist daher nicht erforderlich, dass die zu optimierende Funktion differenzierbar ist.
- Sie verhindern eher ein "gefangen werden" in einem lokalem Minimum, da sie Informationen aus verschiedenen Bereichen der zu optimierenden Funktion gleichzeitig benützen. Sollten an einer anderen Stelle bessere Funktionswerte auftreten, kann die Umgebung des lokalen Minimums verlassen werden.

5 Preprocessing

Eine gute Möglichkeit die Lerngeschwindigkeit eines neuronalen Netzes zu verbessern und gleichzeitig einfachere Bedingungen für die spätere Optimierung zu schaffen, ist eine Vorverarbeitung der Inputdaten in Form einer Transformation. Dadurch wird erreicht, dass die Daten einen einheitlichen Signalpegel aufweisen – folglich in einem ähnlichen Intervall skaliert sind. Ohne eine Vortransformation der Inputdaten müsste das Netz durch eine entsprechende Anpassung der Gewichte, die Größenunterschiede der Daten ausgleichen. Das würde jedoch dazu führen, dass man die Lernrate zu Gunsten der kleinen Gewichte adaptieren muss, was die Konvergenz stark verlangsamt. Andernfalls, wenn die Lernrate entsprechend dem Verhalten der großen Gewichte eingestellt wird, kann es aufgrund der Übersteuerung der Gewichte dazu kommen, dass der Lernvorgang überhaupt nicht konvergiert. [vgl. [Zim94], [Web00]]

Im folgenden sollen die am häufigsten angewendeten Transformationen beschrieben werden [vgl. [Zim94]]:

1. Einfache Skalierung

$$z_t = \frac{x_t - \text{aver}(x)}{\text{std}(x)} \tag{29}$$

aver(x) bezeichnet hier den Mittelwert der Daten und std(x) deren Standardabweichung. Es wird versucht verschiedene Inputdaten auf eine glei-

che mittlere Signalvariabilität zu bringen. Extremwerte können dabei nach oben bzw. unten abweichen.

2. Die [0,1]-Normierung

$$z_t = \frac{x_t - \min(x)}{\max(x) - \min(x)} \tag{30}$$

Wie schon die Bezeichnung sagt, wird hier eine Transformation der Daten auf den Bereich von Null bis Eins vorgenommen. $\min(x)$ und $\max(x)$ stehen für das Minimum bzw. das Maximum der Inputdaten. Ein Nachteil dieser Transformation ist ihre Sensitivität gegenüber Ausreißern. Ein Extremwert kann dazu führen, dass der Großteil der Daten, der im mittleren Streubereich liegt, beinahe auf eine Konstante transformiert wird.

3. Z-Transformation

$$z_t = \frac{x_t - \mathrm{AM}(x)}{\mathrm{std}(x)} \tag{31}$$

AM steht hier für arithmetisches Mittel. Diese Transformation führt dazu, dass alle Daten einen Mittelwert von Null und eine Varianz von Eins aufweisen. Die Korrelationsstruktur der Variablen bleibt dabei unverändert.

6 Probleme bei der Prognose mit neuronalen Netzen

Die größte Schwierigkeit bei der Anwendung von neuronalen Netzen für die Prognose ist die Bestimmung der Modellgröße. Erhöht man die Anzahl der *units* im *hidden-layer*, so erzeugt man einen Anstieg der Freiheitsgrade und die Plastizität des Netzes nimmt zu. Damit kann sich das Netz der Trainingsmenge besser anpassen und die Wahrscheinlichkeit des Erlernens einer Gewichtskonfiguration steigt. Gleichzeitig sinkt mit der verbesserten Approximation jedoch die Generalisierungsfähigkeit. Bei einer zu großen Anzahl an freien Parametern kommt es zu einer Überanpassung (*overfitting*) des Datensatzes, ohne dass das Netz etwas über die Struktur der Aufgabe lernt. Bis jetzt gibt es noch keine analytische Lösung dieses Problems, jedoch eine Reihe von Möglichkeiten die bei der Suche der optimalen Netzstruktur hilfreich sind.

Um den Effekt des *overlearning* (Auswendiglernen der Daten) zu verhindern können folgende Verfahren angewendet werden:

- *Stopped training* Methode
- Addition von Rauschen zum *input*
- *Penalty* Verfahren
- *Pruning* Techniken

6.1 Stopped Training

Der Gedanke des *stopped training* ist, das Lernverfahren zu dem Zeitpunkt abzubrechen, an dem das Netz alle wichtigen und relevanten Informationen aus den Trainingsdaten extrahiert hat. Damit kann das Problem des *overlearnings* verhindert werden. Unter *overlearning* (Auftreten von *overfitting*) spricht man, wenn das Netz individuelle Besonderheiten der Eingabedaten lernt und somit beginnt sich vollständig der Trainingsmenge anzupassen. [vgl. [Mec94]]

Die Vorgehensweise ist folgende: Die zur Verfügung stehenden Daten werden in 3 Teilmengen unterteilt – eine Trainingsmenge, eine Validierungsmenge und eine Testmenge, die ungefähr im Verhältnis 70 %, 20 % und 10 % stehen sollen. Zum Lernen werden ausschließlich die Daten der Trainingsmenge verwendet – die Validierungsmenge dient zur Messung des Übertrainings. Man beobachtet nun parallel die Fehlerabweichungen auf der Trainings- und der Validierungsmenge. Zu Beginn wird der Fehler auf beiden Mengen sinken, doch nach einiger Zeit wird die Fehlerfunktion der Validierungsmenge wieder zu steigen beginnen, während der Fehler auf den Trainingsdaten kontinuierlich weitersinkt. Das ist der geeignete Zeitpunkt, um das Lernverfahren zu stoppen. Würde man das Lernen an dieser Stelle nicht abbrechen, so würde das Netz beginnen, nicht strukturstabile Eigenschaften der Daten abzubilden und die Generalisierungsfähigkeit wäre damit eingeschränkt. [vgl. [Luk05], [Zim94]]

Das *stopped training* führt in vielen Fällen, vor allem wenn die Trainings- und Validierungsmengen klein sind, zu einem relativ frühen Abbruch des Trainings, bereits nach wenigen Epochen. Dadurch entsteht das Problem, dass das Netz nicht die Möglichkeit hat komplizierte Nichtlinearitäten in den Daten zu erlernen und den linearen Anteil somit überschätzt. Ein weiterer Kritikpunkt ist, dass es bis jetzt noch keine formalen Beweise gibt, dass der Fehler auf der Validierungsmenge nach einem Anstieg nicht wieder sinken könnte. Eine Möglichkeit wäre, den Stopp-Zeitpunkt lediglich vorzumerken und das Netz dennoch weiterzutrainieren, um mögliche Verbesserungen zu erzielen bzw. ein neues Training, mit einer neuen Gewichtskonfiguration zu starten. Die Tatsa-

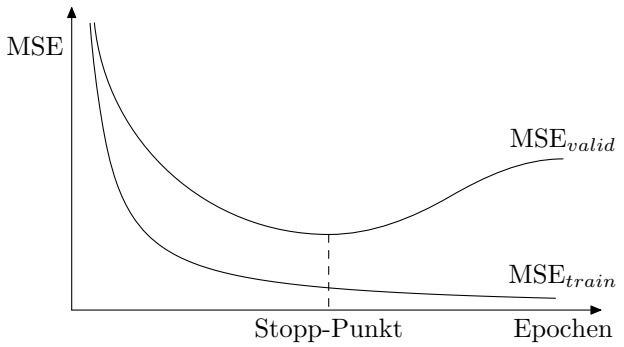


Abbildung 14: Stopped-Training-Punkt

che, dass das Netz in der Lage ist Rauschen zu approximieren, weist außerdem darauf hin, dass das Netz noch überparametrisiert ist. Es sollten also die Parameter reduziert werden, anstelle eines frühzeitigen Abbruchs des Trainings. Man erkennt also, dass das *stopped training* alleine nicht ausreicht, um das jeweilige beste Netz zu identifizieren. Eine gute Möglichkeit ist, das *stopped training* in Verbindung mit anderen Optimierungsverfahren anzuwenden. In den folgenden Kapiteln sollen einige Möglichkeiten zur weiteren Verbesserung der Netztopologie vorgestellt werden. [vgl. [Zim94], [Web00], [And97]]

6.2 Addition von Rauschen zum Input

Beim *training with noise* wird in jedem Trainingsschritt zum Netzinput ein kleiner Zufallsterm addiert. Das führt dazu, dass die Daten gestreut werden und es für das Netz schwieriger wird, eine punktweise Interpolation auszuführen. Es wird nur mehr die Hauptstruktur der Daten aufgenommen. Die beste Lösung ist meistens die Verwendung eines weißen Rauschens. Es stellt sich natürlich die Frage wie hoch der Anteil des Rauschens im Verhältnis zur Varianz gewählt werden soll. Wurden die Daten vorab transformiert, so ist durch die Skalierung eine einheitliche Varianz von Eins gegeben. Man kann nun einen Prozentsatz an Rauschen wählen, der für alle Daten gleich ist – in der Praxis hat sich ein Anteil von 10 % bewährt. [vgl. [Zim94], [Bis95b]]

6.3 Die Penalty Verfahren

Beim *penalty* Verfahren wird die Fehlerfunktion um einen Term erweitert, dem sogenannten *penalty* Term, der ein Maß für die Komplexität des Netzes darstellt. Aus der Überlegung, dass kleine Gewichte annähernd linearen Modellen entsprechen und große Gewichte nichtlineare Strukturen abbilden, bietet sich als Komplexitätsreduktion eine Beschränkung der Gewichte an. [vgl. [Zim94]]

Die verschiedenen Verfahren unterscheiden sich in der Wahl ihres *penalty* Terms – die bekanntesten sind das *weight decay* und die *weight elimination*. Neben diesen beiden, werden auch noch das Verfahren nach Hanson und Pratt sowie das Verfahren nach Chauvin vorgestellt.

6.3.1 Weight decay

Das Verfahren des *weight decay* wurde von P. Werbos erstmals 1988 in seiner Arbeit *"backpropagation: past and future"* [Wer88] beschrieben.

Beim *weight decay* sieht die regularisierte Zielfunktion wie folgt aus:

$$E^{\text{neu}} = E + \frac{\lambda}{2} \sum_{ij} w_{ij}^2 \tag{32}$$

Ziel ist nun nicht mehr die optimale Anpassung an die Trainingsbeispiele, sondern eine Minimierung des Netzwerkfehlers bei gleichzeitiger Minimierung des Betrags der Gewichte. Man erkennt, dass beim *weight decay* der Term, der zum Fehler hinzuaddiert wird, umso größer ist, je größer die Gewichte sind. Das heißt, es findet eine Bestrafung großer Gewichte statt – kleine Gewichte

hingegen werden bevorzugt. Große Gewichte bewirken eine stärkere Zerklüftung der Fehleroberfläche, die das generische Lernen erschwert.

Für die Änderung eines Gewichts ergibt sich:

$$\Delta w_{ij}(t+1) = -\eta o_{pi}\delta_{pj} - \lambda w_{ij} \tag{33}$$

Der Name *weight decay* (Gewichtszerfall) resultiert aus der Auswirkung auf die Gewichtsänderungen: die Gewichte nähern sich dem Wert Null an, wenn sie nicht durch den Gradienten der Fehlerabweichungsfunktion vergrößert werden. Der Parameter λ (*penalty* Term) gibt dabei das Ausmaß des Zerfalls an. Meist wird ein λ zwischen 0,005 und 0,03 vorgeschlagen – in der Praxis hat sich oft ein λ um 0,01 bewährt. [vgl. [Zim94], [Zel94a], [Web00]]

Man kann das *penalty* Verfahren in Verbindung mit dem *stopped training* verwenden. Sobald der Fehler der Validierungsmenge zu steigen beginnt, muss λ vergrößert werden. Das Ziel ist es λ so einzustellen, dass die Lern- und Validierungsmenge ein ähnliches Verhalten der Fehlerverläufe aufweisen. [vgl. [Zim94]]

Beim *weight decay* sollen weniger bedeutende Gewichte zu Null werden – allerdings kann nicht verhindert werden, dass auch relevante Gewichte zerfallen, wenn sie nicht ständig durch den Fehlergradienten ausgeglichen werden. Ein anderes *penalty* Verfahren, das dieses Problem beheben soll ist die sogenannte *weight elimination*.

6.3.2 Weight elimination

Die Methode *weight elimination* wurde 1990 in der Arbeit *"Predicting the future: A connectionist approach"* von A. S. Weigend, D. E. Rumelhart und G. E. Hinton vorgestellt.

Die Fehlerfunktion bei *weight elimination* ist folgende:

$$E^{\text{neu}} = E + \lambda \sum_{ij} \frac{\omega_{ij}^2}{1 + (\frac{\omega_{ij}}{\omega_0})^2} \tag{34}$$

Für kleine Gewichte verhalten sich *weight decay* und *weight elimination* ähnlich. Große Gewichte bleiben bei der *weight elimination* eher unberührt. Der Parameter ω_0 ist eine Konstante, die frei gewählt werden kann. Sie gibt den Übergang zwischen den verschiedenen Verhaltensbereichen an.

Für die Änderung der Gewichte ergibt sich hier:

$$\Delta\omega_{ij} = -\eta \frac{\partial E}{\partial \omega_{ij}} - \lambda \frac{\omega_{ij}}{(1 + (\frac{\omega_{ij}}{\omega_0})^2)^2} \tag{35}$$

Ein Nachteil der *weight elimination* ist ihre hohe Empfindlichkeit gegenüber Veränderungen des Parameters λ, dessen Bestimmung hier aufwendiger ist. Grundsätzlich ist allen *penalty* Verfahren gemein, dass sie lineare Modelle bevorzugen. Durch die Verwendung eines *penalty* Terms wird die Fehlerfunktion verzerrt. Daher sollten Gewichte, die am Ende des Trainings Werte nahe Null aufweisen aus dem Netz entfernt werden – danach ist ein erneutes Trainieren des Netzes mit einer nicht regularisierten Netzfunktion notwendig. [vgl. [Zim94], [Web00], [And97]]

6.3.3 Kostenfunktionen für die Gewichte von hidden-units

Neben den beschriebenen Kostenfunktionen (Gleichungen 32 und 34), gibt es auch noch Kostenfunktionen für die Elimination von Zellen. Das Verfahren wurde von S. J. Hanson und L. Y. Pratt in der Arbeit *"Comparing biases for minimal network construction with back-propagation"* [HP89] vorgestellt. Die Herleitung basiert auf dem *weight decay* Verfahren (siehe dazu Kapitel 6.3.1). Es wird eine *objective function* definiert, die sich aus der Fehlerfunktion E (*error term*) und einem Kostenterm zusammensetzt – Hanson und Pratt bezeichnen den Kostenterm als *bias function* B:

$$O = E + B \tag{36}$$

Das Ziel ist nun O zu minimieren:

$$\frac{\partial O}{\partial w_{ij}} = \frac{\partial E}{\partial w_{ij}} + \frac{\partial B}{\partial w_{ij}} \tag{37}$$

Damit sich der Kostenterm nicht auf ein Gewicht, sondern auf eine *unit* bezieht, werden sämtliche Beträge der Verbindungsgewichte, die zu dieser *unit* führen aufsummiert:

$$w_i = \sum_j \mid w_{ij} \mid \tag{38}$$

Hanson und Pratt haben zwei *bias functions* untersucht. Einerseits den *hyperbolic bias*:

$$B = \frac{w_i}{1+\lambda w_i} \quad \text{mit der Ableitung} \quad -\frac{\partial B}{\partial w_{ij}} = -\frac{\lambda sgn(w_{ij})}{(1+w_i)^2} \tag{39}$$

und andererseits den *exponential bias*:

$$B = (1 - e^{-\lambda w_i}) \quad \text{mit der Ableitung} \quad -\frac{\partial B}{\partial w_{ij}} = -\frac{sgn(w_{ij})}{(e^{\lambda w_i})} \tag{40}$$

In den Untersuchungen wurden mit der Exponentialfunktion bessere Ergebnisse als mit der hyperbolischen Funktion erzielt.

Die Verwendung einer Kostenfunktion zur Elimination von *hidden-units* kann in manchen Fällen die Generalisierungsfähigkeit des Netzes erhöhen, es kann jedoch auch passieren, dass das Lernverfahren nach dem Ausdünnen nicht mehr konvergiert. Der Grund dafür liegt vermutlich in den, durch die unterschiedliche Gewichtung des Gradienten zur Reduzierung des Gesamtfehlers des Netzes und des Gradienten zur Eliminierung von *hidden-units*, auftretenden Problemen.

6.3.4 Kostenfunktion für die Ausgaben von hidden-units

Eine weitere Möglichkeit der Verwendung einer Kostenfunktion stellte 1989 Y. Chauvin vor [vgl. [Cha89]]. Die Kostenfunktion C (*cost function*), die zu minimieren ist, setzt sich aus der gewichteten Summe eines *error terms* und eines *energy terms* zusammen:

$$C = \mu_{er} E_{er} + \mu_{en} E_{en} \tag{41}$$

Der *error term* wird wie bei *back-propagation* (siehe Kapitel 4.1) berechnet. Für den *energy term* verwendet man eine positive monotone Kostenfunktion e auf die Quadrate der *output*-Werte aller *hidden-units*. Bei h *hidden-layer* ergibt sich der gesamte *energy term* also als Summe aller *energy terms* für jeden *hidden-layer*:

$$E_{en} = \sum_{i}^{h} \sum_{j}^{H_i} e(o_j^2) \tag{42}$$

Für jeden *hidden-layer*, der vor dem betrachteten *layer* h liegt, gilt folgendes:

$$\delta_k^{en} = f_k'(net_k) \sum_j \delta_j^{en} w_{jk} \tag{43}$$

Dies ist dieselbe Formel, wie sie auch bei *back-propagation* zum Einsatz kommt, jedoch mit einem anderen zurückpropagierten Term.

Die gesamte Änderung für die Fehlerrückpropagierung und die Rückpropagierung der Kosten kann man in einem Term, wie folgt, zusammenfassen:

$$\Delta w_{kl} = -\alpha \mu_{er} \delta_k^{er} o_l - \alpha \mu_{en} \delta_k^{en} o_l = -\alpha o_l (\mu_{er} \delta_k^{er} + \mu_{en} \delta_k^{en}) = -\alpha o_l \delta_k^{ac} \tag{44}$$

δ^{ac} bezeichnet hier das summierte Fehlersignal für den eigentlichen Netzfehler nach dem *back-propagation* Verfahren und die Kosten nach dem Verfahren von Chauvin.

$$\delta_k^{ac} = f_k'(net_k) \sum_i (\mu_{er} \delta_i^{er} + \mu_{en} \delta_i^{en}) w_{ik} = f_k'(net_k) \sum_i \delta_i^{ac} w_{ik} \tag{45}$$

Man kann, also die Delta der Fehlersignale des Netzwerkfehlers und der *energy* zusammenfassen und gemeinsam unter Anwendung des *back-propagation* Algorithmus durch das Netzwerk zurückpropagieren.

Durch die Minimierung der Quadrate der *outputs* der *hidden-layer* wird gleichzeitig auch die Anzahl der *hidden-units* minimiert, da eine hohe Anzahl an Neuronen in der verdeckten Schicht gleichzeitig einen hohen Wert der Kostenfunktion impliziert. Man kann das Verfahren außerdem in Kombination mit *weight decay* (siehe dazu Formel 32) einsetzen. In diesem Fall wird zusätzlich ein Summand für die Größe der Gewichte eingeführt. Die zu minimierende Zielfunktion sieht dann wie folgt aus:

$$C = \mu_{er} \sum_{j}^{P} \sum_{i}^{O} (t_{ij} - o_{ij})^2 + \mu_{en} \sum_{j}^{P} \sum_{i}^{H} e(o_{ij}^2) + \mu_w \sum_{ij}^{W} w_{ij}^2 \tag{46}$$

6.4 Pruning Techniken

Der Begriff *pruning* heißt im Englischen so viel wie "Ausdünnen" – bei diesen Verfahren wird das neuronale Netz nach dem Training um *units* oder Gewichte reduziert. Man unterscheidet grundsätzlich zwischen *weight pruning* und *unit pruning*. Bekannte Verfahren des *weight prunings* sind das *optimal brain damage*, sowie dessen Verallgemeinerung das *optimal brain surgeon*. Das *unit pruning* unterteilt sich wiederum in zwei Kategorien: einerseits können *input-units* und andererseits *hidden-units* aus dem Netz entfernt werden.

Die folgenden Kapitel sollen einen Überblick über die verschiedenen Möglichkeiten des *prunings* geben.

6.4.1 Löschen der betragsmäßig kleinsten Gewichte

Das Löschen der betragsmäßig kleinsten Gewichte ist das einfachste Verfahren zum Ausdünnen von neuronalen Netzen. Man geht davon aus, dass betragsmäßig kleine Gewichte unwichtigen Strukturen entsprechen. Die Bedeutung eines Gewichts wird in der Literatur häufig mit dem Begriff *saliency* beschrieben. Das Problem dabei ist, dass in manchen Fällen ein linearer Zusammenhang in den Daten gefunden wurde, der durch das Nullsetzen der Gewichte verschwindet. Aus diesem Grund sollte bei der Anwendung dieses Verfahrens eine Netzwerkstruktur mit direkten Konnektoren verwendet werden, die lineare Zusammenhänge abbilden können. Grundsätzlich kann das Verfahren zu jedem Zeitpunkt des Trainings angewendet werden – es erweist sich jedoch als sinnvoll es zum Stopp-Zeitpunkt (siehe dazu Kapitel 6.1) einzusetzen. Nach dem Ausdünnen sollte ein Nachtraining stattfinden. Das Verfahren kann bei sehr großen Netzen zu Beginn recht gute Ergebnisse liefern. Im Allgemeinen wird man jedoch eher Verfahren, wie sie in den nächsten Kapiteln erläutert werden, einsetzen. [vgl. u. a. [Zim94], [Zel94a], [Luk05], [Bis95a]]

6.4.2 Optimal brain damage

Das *optimal brain damage* (OBD) wurde 1990 von Y. Le Chun, J. S. Denker und S. A. Solla in dem Artikel *"Optimal Brain Damage"* [CDS90] zur Ausdünnung von neuronalen Netzen vorgeschlagen. Es basiert auf der Taylorreihenentwicklung der Zielfunktion um ein lokales Minimum:

$$\Delta E = \sum_i g_i \Delta w_i + \frac{1}{2} \sum_i h_{ii} \Delta w_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \Delta w_i \Delta w_j + O(\| \Delta \mathbf{W} \|^3) \tag{47}$$

Für g_i und h_{ij} gilt dabei folgendes:

$$g_i = \frac{\partial E}{\partial w_i} \tag{48}$$

$$h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{49}$$

g_i bilden also die Gradienten von E nach dem Gewichtsvektor $\mathbf{W}$. h_{ij} bilden die Elemente der Hesse Matrix $\mathbf{H}$ (partielle 2. Ableitungen) von E nach dem Gewichtsvektor $\mathbf{W}$. [vgl. u. a. [CDS90], [Luk05]]

Die Hesse Matrix wächst quadratisch mit der Anzahl der Gewichte, die linear mit der Anzahl der *input-units* und der *hidden-units* zusammenhängen. Für größere Netze wird daher auch der Rechenaufwand immer umfangreicher. [vgl. [Luk05]]

Es werden drei vereinfachende Annahmen getroffen [vgl. [Zel94a], [Zim94]]:

1. Die Hesse Matrix wird einer Diagonalmatrix angenähert, indem die zweiten Ableitungen h_{ij} für $i \neq j$ vernachlässigt werden.

2. Man geht davon aus, dass man sich bereits in einem lokalen Minimum der Fehlerfunktion befindet. Dadurch wird die erste Ableitung Null und der Term $\sum_i g_i \Delta w_i$ fällt weg.

3. Man nimmt an, dass die Fehlerfunktion in der Nähe des Minimums lokal quadratisch ist. Deshalb können Terme höherer Ordnung vernachlässigt werden.

Gleichung 47 vereinfacht sich also zu:

$$\Delta E = \frac{1}{2} \sum_i h_{ii} \Delta w_i^2 \tag{50}$$

Zusammenfassend ist die Vorgehensweise beim *optimal brain damage* folgende [vgl. u. a. [CDS90], [Luk05]]:

1. Wählen einer geeigneten Netzwerkarchitektur

2. Trainieren des Netzwerks, mittels *back-propagation* oder eines beliebigen anderen Lernalgorithmus, bis die Fehlerfunktion zu einem lokalen Minimum konvergiert

3. Berechnung der zweiten Ableitung für jedes Gewicht k

4. Berechnung der *saliencies* für jedes Element (Relevanz für die Peformance des Netzwerks) gemäß:

$$s_k = h_{kk} \frac{w_k^2}{2} \tag{51}$$

5. Sortieren der Gewichte nach der *saliency* und eliminieren jener Gewichte mit einem geringen Wert der *saliency*, indem man sie auf Null setzt (anstelle des Nullsetzens kann auch bloß eine Reduktion des Gewichts vorgenommen werden)

6. Fortfahren mit Schritt 2

Ziel des OBD ist es also, jene Gewichte zu finden und zu eliminieren, die die geringste *saliency* haben – das heißt, jene Gewichte, die den geringsten Beitrag zur Erklärung der unabhängigen Variablen liefern. [vgl. [And97]]

Je kleiner ein Gewicht bzw. seine zweite Ableitung ist, umso niedriger ist seine *saliency*. Das heißt, ein Entfernen dieses Gewichts hat nur geringfügige Auswirkungen auf die Fehlerfunktion. Ein kleines Gewicht kann jedoch dann im Netz behalten werden, wenn seine zweite Ableitung seine Wichtigkeit betont. Damit ist im Gegensatz zum Ausdünnen der betragsmäßig kleinsten Gewichte keine Bevorzugung nichtlinearer Strukturen gegeben. [vgl. [Zim94]]

Ein Problem dieses Verfahrens, ist die implizite Unterstellung, dass alle Gewichte k voneinander unabhängig sind, was in Wirklichkeit nicht der Fall ist. Durch das Nullsetzen eines Gewichts wird es aus dem Netz entfernt – dadurch können sich die *sacliences* der anderen Gewichte verändern. [vgl. [And97]]

6.4.3 Optimal brain surgeon

Das *optimal brain surgeon* (OBS) ist eine Verallgemeinerung des *optimal brain damage*. Während beim OBD nur die Diagonale der Hesse Matrix verwendet wird, werden beim OBS alle Gewichte der Hesse Matrix berechnet, um jene Gewichte zu bestimmen, die gelöscht werden können. Es wird als im Gegensatz zum OBD nicht von einer Unabhängigkeit der Gewichte voneinander ausgegangen. [vgl. [HSW93], [HS93]]

Den Ausgangspunkt des OBS bildet wie schon beim OBD die Taylorreihen-Approximation für die Fehlerfunktion E (siehe Gleichung 47). Ebenfalls werden die beiden Annahmen, dass das Netz bereits zu einem lokalen Minimum trainiert wurde und dass die Fehlerfläche in der Nähe eines Minimums lokal quadratisch ist vom OBD übernommen. Damit verschwinden wiederum der erste und letzte Term der Taylorreihenentwicklung.

Das Ziel ist nun ein Gewicht w_q Null zu setzen, um den Fehler zu minimieren. Das Eliminieren von w_q kann durch folgende Formel zum Ausdruck gebracht werden:

$$\mathbf{e}_q^{\mathrm{T}} \Delta \mathbf{W} + w_q = 0 \tag{52}$$

$\mathbf{e}_q$ bezeichnet den zu w_q gehörigen Einheitsvektor im Raum der Gewichte. Das Verfahren versucht nun folgendes Minimum zu finden:

$$\min_q \{ \min_{\Delta \mathbf{W}} (\frac{1}{2} (\Delta \mathbf{W}^{\mathrm{T}}) \mathbf{H} \Delta \mathbf{W} | \mathbf{e}_q^{\mathrm{T}} \Delta \mathbf{W} + w_q = 0) \} \tag{53}$$

Zur Lösung dieser Aufgabe wird die Lagrangesche Funktion L gebildet:

$$L = \frac{1}{2} (\Delta \mathbf{W}^T) \mathbf{H} \Delta \mathbf{W} + \lambda (\mathbf{e}_q^{\mathrm{T}} \Delta \mathbf{W} + w_q) \tag{54}$$

λ ist ein zu bestimmender Lagrange Multiplikator. Nach Bildung der funktionalen Ableitungen, Anwendung der Nebenbedingungen und Matrixinversion erhält man:

$$\Delta \mathbf{W} = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} \mathbf{e}_q \tag{55}$$

$\mathbf{H}^{-1}$ bezeichnet die Inverse der Hesse Matrix, die in diesem Fall keiner Diagonalmatrix entsprechen muss. $\mathbf{e}_q$ gibt die i-te Spalte der Hesse Matrix an.

Die *saliences* werden nach folgender Gleichung bestimmt:

$$L_q = \frac{1}{2} \frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}} \tag{56}$$

Die *saliency* gibt den Anstieg des Fehlers bei der Elimination eines Gewichts an.

Der Ablauf beim OBS unterscheidet sich in den ersten beiden Schritten nicht von jenem des OBD: es wird eine geeignete Netzwerkarchitektur gewählt und das Netzwerk bis zu einem lokalen Minimum trainiert. Im dritten Schritt muss allerdings die vollständige Inverse der Hesse Matrix $\mathbf{H}$ der Fehlerfunktion berechnet werden. Danach müssen für jedes Gewicht die *saliences* nach Gleichung 56 berechnet werden. Ist das Gewicht q mit dem niedrigsten Wert der *salieny* wesentlich kleiner als der Gesamtfehler E, so wird das Gewicht aus dem Netz entfernt. Unter Verwendung des gefundenen q werden nach Gleichung 55 alle Gewichte angepasst. Ist der kleinste *saliency* Wert nicht mehr wesentlich kleiner als der Gesamtfehler E des Netzes wird das Verfahren beendet. "Nicht wesentlich kleiner" ist eine sehr vage Bezeichnung – es gibt allerdings sowohl beim *optimal brain damage* als auch beim *optimal brain surgeon* keine genaue Definition, wann das Training tatsächlich abgebrochen werden soll. Es empfiehlt sich daher nach jedem *pruning* eines Gewichts ein sogenanntes Selektionskriterium zu bestimmen (siehe dazu Gleichung 92 in Kapitel 11). Das Training kann dann gestoppt werden, wenn das Selektionskriterium keine Verbesserung mehr anzeigt. [vgl. [And97], [HS93], [HSW93]]

Der Rechenaufwand ist durch die Berechnung der vollständigen Hesse Matrix $\mathbf{H}$ um einiges höher als beim *optimal brain damage*. Die Zeitkomplexität zur Berechnung der Hesse Matrix zum Löschen eines Gewichts liegt hier bei $\mathrm{O}(n_p n_{\mathrm{out}} n_w^2)$, wobei n_p die Zahl der Trainingsmuster, n_{out} die Zahl der *output-units* und n_w die Zahl der Gewichte im Netz angibt. Im Gegensatz dazu hat das *optimal brain damage* eine Zeitkomplexität von $\mathrm{O}(n_p n_w)$ zum Löschen eines Gewichts. Am wenigsten Zeitaufwand erfordert allerdings das Ausdünnen der betragsmäßig kleinsten Gewichte (Zeitkomplexität von $\mathrm{O}(n_w)$). Mit zunehmenden Zeitaufwand werden die Verfahren jedoch auch besser. OBS reduziert bei gleichem Fehler auf der Trainingsmenge mehr Gewichte als die anderen Verfahren. [vgl. [Zel94a],[HS93]]

6.4.4 Statistische Gewichtsausdünnung

Bei den meisten Verfahren wird davon ausgegangen, dass Verbindungen mit kleinen Gewichte automatisch ein Hinweis auf eine geringe Bedeutung der entsprechenden *units* ist. Es besteht jedoch die Möglichkeit, dass die *unit* zwar eine geringe Relevanz hat, aber dennoch einen wichtigen Beitrag für das Modell liefert, so dass bei ihrem Entfernen der Fehler des Netzes ansteigen würde.

Für diese Fälle haben W. Finnoff und H. G. Zimmermann ein Testkriterium entwickelt. Während des Trainings des Netzes findet eine ständige Änderung der Gewichte statt – es stellt sich dabei die Frage, welche dieser Änderungen

tatsächlich einen Erklärungsbeitrag liefern und welche nur zu einem Rauschen führen.

$$\text{test(w)} = \frac{\bar{w}^2}{\sum_{t=1}^{T}(w_t - \bar{w})^2} \tag{57}$$

Der Test gibt die mittlere Größe des Gewichts zur Schwankungsbreite der Fluktationen an. Ist der Wert der Testgröße hoch, so hat die *unit* eine hohe Relevanz, andernfalls liefert die *unit* keine strukturrelevanten Informationen. In der Praxis ist eine klare Trennung zwischen dem Rauschen und dem strukturstabilen Anteil nicht möglich. Im Kapitel 6.1 wurde bereits der Stopp-Punkt als jener Zeitpunkt charakterisiert, an dem das Netz am besten die Struktur der *inputs* aufgenommen hat, ohne die Daten bereits auswendig zu lernen. Es ist sinnvoll die statistische Gewichtsausdünnung an jenem Punkt einzusetzen.

Für bereits ausgedünnte Netze sieht das Testkriterium wie folgt aus:

$$\text{test(w)} = \frac{(\sum_t \Delta w_t)^2}{\sum_t (\Delta w_t - \Delta \bar{w}_t)^2} \tag{58}$$

Der Test gibt hier das Verhältnis des kumulativen Gradienten der Trainingsmenge zur Schwankungsbreite des Gradienten an. Es können die Testverfahren für aktive und inaktive Gewichte miteinander verglichen werden und bei Bedarf Verfahren zur Wiederbelebung integriert werden.

Der Vorteil des Finnoff/Zimmermann Tests ist die Erhaltung kleiner Gewichte im Netz – dadurch wird verhindert, dass lineare Strukturen im Netz unterrepräsentiert sind. Besonders bei Netzen mit einer hohen Anzahl an Freiheitsgraden und einem großen Rauschanteil erzielt dieses Verfahren gute Ergebnisse. Außerdem erlaubt es im Gegensatz zum *optimal brain damage* eine Wiederbelebung von bereits eliminierten *units*. Das ist insofern ein wichtiger Punkt, da am Beginn des Trainings das Netz noch überdimensioniert ist und daher die Struktur der Daten noch nicht gut wiedergegeben wird – aus diesem Grund kann es zum Ausdünnen von *units* kommen, die zu einem späteren Zeitpunkt relevant sein können.

6.4.5 Hidden-unit pruning

Das *hidden-unit pruning* wird erst nachdem das Netz einige Zeit gelernt hat eingesetzt – würde man gleich zu Beginn einen relativ kleinen *hidden-layer* verwenden, würde es zu einer Überschätzung der Linearitäten in den Daten kommen. Beim *weight pruning* verlieren durch das Nullsetzen von Gewichten, einige *hidden-units* bereits an Bedeutung – es kann dabei auch zur Elimination von ganzen *units* kommen. Das *weight pruning* alleine liefert jedoch meist noch nicht die bestmögliche Neuronenanzahl – in der Praxis zeigt sich, dass sich durch die Anwendung eines *unit prunings* die Netzkomplexität weiter verringern lässt.

Eine Möglichkeit besteht darin auf einem bereits fertig trainierten Netz eine Korrelationsanalyse der Neuronenaktivitäten durchzuführen. Zu Beginn des Trainings ist es üblich, dass einige Neuronen im *hidden-layer* sehr stark korreliert sind – das gibt einen Hinweis darauf, dass der *hidden-layer* zu diesem Zeitpunkt noch zu groß ist. Im fortlaufenden Training nimmt die Stärke dieser Kor-

relationen jedoch ab. Sind nach einer gewissen Trainingszeit zwei *hidden-units* noch immer stark korreliert, so können die entsprechenden Neuronen zu einem einzigen Neuron zusammengefasst werden. Wenn eine *unit* während des Trainings immer annähernd denselben *output* liefert, so trägt sie ebenfalls nicht zur Findung einer Lösung bei und kann vollständig entfernt werden. [vgl. [Zim94], [Luk05]]

6.4.6 Input-unit pruning

Nach der Durchführung eines *weight decays* sowie eines *hidden-unit pruning* kann man zusätzlich eine Optimierung der *input-units* vornehmen. Dazu kann ein Rangordnungstest durchgeführt werden. Dabei wird für jede *input-unit* eine Testgröße berechnet, die die Änderung der Zielfunktion angibt, wenn der *input* durch seinen Mittelwert ersetzt wird.

$$\text{test}(x_i) = \text{Zielfunktion}(\cdots, \bar{x}_i, \cdots) - \text{Zielfunktion}(\cdots, x_i, \cdots) \tag{59}$$

Ist bereits ein lokales Minimum gefunden worden, so ist die Zielfunktion unter Nutzung aller *inputs* gleich der Zielfunktion bei Nutzung des verkleinerten Inputvektors. In diesem Fall ist die Testgröße größer bzw. gleich Null und ihr Wert würde den Erklärungsbeitrag der *unit* beschreiben. Das Nullsetzen des zu entfernenden *inputs* würde den Eingangswert an den Knoten verschieben und zu einer *bias*-Verschiebung im ersten *layer* führen. Daher wird die *input-unit* durch den Mittelwert ersetzt.

Man kann mit diesem Verfahren jene *inputs* eliminieren, die nur einen minimalen Erklärungsbeitrag liefern. Bei Zeitreihenanalysen kann man zusätzlich den Test auf verschiedenen Zeitintervallen anwenden und auch jene *inputs* entfernen, die über die Zeit hinweg inkonsistent sind.

Grundsätzlich wird der Rangordnungstest für das *input-pruning* genutzt – man kann ihn jedoch auch für das Aussortieren von *hidden-units* anwenden. Dabei wird der *output* einer *hidden-unit* durch ihren Mittelwert ersetzt und wie beschrieben ihr Erklärungsbeitrag untersucht. [vgl. [Zim94]]

6.4.7 Skelettierung

M. C. Mozer und P. Smolensky haben in der Arbeit *"Skeletonization: A technique for trimming the fat from a network via relevance assessment"* [MS89] ein weiteres Verfahren zur Reduzierung von *units* vorgestellt.

Bei der *skeletonization* wird für jede *unit* ein zusätzlicher Parameter α_i eingeführt, der die *attentional strength* angibt. Es gilt:

$$o_j = f\left(\sum_i w_{ij}\alpha_i o_i\right) \tag{60}$$

Wenn $\alpha_i = 0$ ist, hat die *unit* keinerlei Einfluss auf das Netzwerk, bei $\alpha_i = 1$ verhält sie sich wie eine herkömmliche *unit*. Die Relevanz ist wie folgt definiert:

$$\rho_i = E_{\alpha_i=0} - E_{\alpha_i=1} \tag{61}$$

ρ_i kann durch die Ableitung von E nach α_i an der Stelle $\alpha_i = 1$ approximiert werden:

$$\lim_{\gamma \to 1} \frac{E_{\alpha_i=\gamma} - E_{\alpha_i=1}}{\gamma - 1} = \left.\frac{\partial E}{\partial \alpha_i}\right|_{\alpha_i=1} \tag{62}$$

Unter der Annahme, dass Gleichung 62 auch für $\gamma = 0$ gütig ist, lässt sich näherungsweise schreiben:

$$\frac{E_{\alpha_i=0} - E_{\alpha_i=1}}{-1} \approx \left.\frac{\partial E}{\partial \alpha_i}\right|_{\alpha_i=1} \tag{63}$$

ρ_i lässt sich approximieren durch:

$$\hat{\rho}_i = -\frac{\partial E}{\partial \alpha_i} \tag{64}$$

Bei der Approximation wird immer von $\alpha_i = 1$ ausgegangen – da α_i also nie geändert wird, handelt es sich um keinen wirklichen Parameter des Systems, sondern nur um einen Parameter, der die Notation vereinfacht.

Mozer und Smolensky haben herausgefunden, dass $\frac{\partial E}{\partial \alpha}$ in der Praxis stark mit der Zeit schwankt. Daher wird für die Schätzung der Relevanz folgende Gleichung empfohlen:

$$\hat{\rho}_i(t+1) = 0{,}8\hat{\rho}_i(t) + 0{,}2\frac{\partial E(t)}{\partial \alpha_i} \tag{65}$$

Es ist noch anzumerken, dass der Algorithmus mit einer linearen Fehlerfunktion $E = \sum \mid t_{pj} - o_{pj} \mid$ verwendet werden soll. Eine quadratische Fehlerfunktion führt dazu, dass die Relevanz bei ähnlichen Werten von Ausgabe und tatsächlichem Wert nur schwer geschätzt werden kann. Eine Möglichkeit ist, das Netz beim Training mit einer quadratischen Fehlerfunktion lernen zu lassen, jedoch für die Bestimmung der Relevanz eine lineare Funktion zu verwenden.

7 Auswahl der Inputprädikatoren

Bei Prognosen ist die Bestimmung der richtigen Inputprädikatoren von Bedeutung für den Erfolg. Vor allem dann, wenn eine große Anzahl an Daten zur Verfügung steht muss eine sinnvolle Auswahl getroffen werden. Eine einfache lineare Korrelationsanalyse ist dafür aber nicht ausreichend, da sie nichtlineare Zusammenhänge unberücksichtigt lässt. Stattdessen sollten bereits bei der Vorauswahl der Prädikatoren neuronale Netze zum Einsatz kommen – selbstorganisierende Netze oder *expert council* Topologien sind dafür geeignet [vgl. [Web00]]. Abgesehen von diesen Verfahren kann im Rahmen des *input prunings* eine weitere Reduktion unwichtiger *input-units* vorgenommen werden (siehe dazu Kapitel 6.4.6).

7.1 Expert Council Topologien

Bei dem Verfahren der *expert council* Topologie werden mehrere Teilnetze (Subnetze) gebildet, die jeweils getrennt voneinander mit nur einem Teil der zur Verfügung stehenden Prädikatoren trainiert und optimiert werden. Wenn eines der Teilnetze ein schlechtes Trainingsverhalten aufweist, so kann man davon ausgehen, dass die Prädikatoren, die in diesem Netz verwendet wurden, wenig Relevanz für das Lösen der betreffenden Aufgabe haben. Das Problem dabei ist, dass man zu Beginn festlegen muss, welche Inputvariablen zu *clustern* zusammengefasst werden sollen, was ein weitergehendes Verständnis für das Problem erfordert. [vgl. [Web00]]

Eine weitere Anwendung der *expert council* Topologie ist folgende: es können Teilnetze gebildet werden, die jedoch alle mit denselben Inputdaten trainiert werden. Durch die seperate Handhabung der Daten verfolgen die Subnetze unterschiedliche Suchpfade und produzieren folglich auch unterschiedliche *outputs*. Die verschiedenen *outputs* können durch ein *supervisor* Netz zusammengefasst werden. Dieses bildet dann ein gewichtetes Mittel der Teilnetze. Der Vorteil dabei ist, dass man gleichzeitig, durch eine Analyse der Abweichungen der Submodelle voneinander eine Kenngröße für die Stabilität des Gesamtmodells erhält. [vgl. [Zim94]]

7.2 Selbstorganisierende Netze

Eine weitere Möglichkeit die Dimension der Eingabedaten zu reduzieren, ist der Einsatz von selbstorganisierenden Netzen. Selbstorganisierende Netze (*self-organizing maps*) werden auch als *kohonen maps* bezeichnet, benannt nach ihrem Entwickler T. Kohonen. Sie werden mittels eines unüberwachten Lernverfahrens trainiert – das heißt, der *output* ist nicht bekannt und somit wird auch keine Fehlerfunktion definiert. Selbstorganisierende Netze sind in der Lage sowohl lineare als auch nichtlineare Gemeinsamkeiten in den Daten ausfindig zu machen und abzubilden. [vgl. [Web00], [Roj93], [NKK94]]

Im Grunde besteht ein selbstorganisierendes Netz aus einem *input-layer* und einem *hidden-layer*, der die eigentliche Karte (Kohonen Schicht) darstellt. Jede *input-unit* ist mit sämtlichen *units* im *hidden-layer* verbunden. Die *units* im *hidden-layer* weisen untereinander ebenfalls gewichtete Verbindungen auf. [vgl. u. a. [Web00], [Zel94a]]

Das Lernen erfolgt nun in folgenden Schritten: zu Beginn muss ein *winner*-Neuron bestimmt werden. Dazu werden zunächst die Gewichtsvektoren m_i der Neuronen per Zufall bestimmt und mit einem n-dimensionalen Eingabevektor x in einer bestimmten Metrik verglichen. Meist wird hier der euklidische Abstand verwendet. Das Gewinnerneuron ist jenes, mit dem minimalen euklidischen Abstand zwischen x und m_i:

$$\| x - m_c \| = \min_i \{ \| x - m_i \| \} \tag{66}$$

Im nächsten Schritt werden alle Neuronen in der Nachbarschaft nach folgender Vorschrift angepasst:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{67}$$

h_{ci} ist dabei die sogenannte Nachbarschaftsfunktion (*neighborhood kernel*). Es ist wichtig, dass die Nachbarschaftsfunktion für $t \to \infty$ gegen Null geht, damit eine Konvergenz möglich ist. Normalerweise gilt:

$$h_{ci}(t) = h(\parallel r_c - r_i \parallel) \tag{68}$$

wobei r_c und r_j die Positionsvektoren der Neuronen c und j im Gitter angeben. In [Koh00] werden zwei Nachbarschaftsfunktionen empfohlen. Eine Möglichkeit ist die Definition einer Nachbarschaftsumgebung N_c um das *winner*-Neuron:

$$h_{ci}(t) = \begin{cases} \alpha(t) & , \quad i \epsilon N_c(t) \\ 0 & , \quad \text{sonst} \end{cases} \tag{69}$$

$N_c(t)$ gibt die r-te Nachbarschaft des Siegerneurons an. $0 < \alpha(t) < 1$ wird Lernrate genannt. Sowohl die Lernrate, als auch der Radius von N_c nehmen mit der Zeit ab.

Eine zweite Möglichkeit ist die Verwendung der Gauss Funktion:

$$h_{ci} = \alpha(t) \exp\left(-\frac{\parallel r_c - r_i \parallel^2}{2\sigma^2(t)}\right) \tag{70}$$

Abhängig von der Anzahl der bereits durchgeführten Schritte muss schließlich bestimmt werden, ob das Training beendet oder fortgesetzt werden soll. Beim Fortsetzen werden die Lernrate α und der Radius r nach Plan reduziert. Somit werden die Inputprädikatoren als *cluster* auf der Kohonen Schicht abgebildet – man erhält eine topologieerhaltende Abbildung.

Durch die Aktualisierung der Neuronengewichte nach Gleichung 67 werden die Gewichtsvektoren in Richtung des Eingabevektors x gezogen. Der Nachbarschaftsradius r bewirkt, dass jede Gewichtsänderung auch die Nachbarneuronen beeinflusst. Dieser Einfluss nimmt jedoch aufgrund der Reduktion von r im Laufe des Trainings ab. Die Reduzierung der Lernrate α führt außerdem dazu, dass die Gewichtsanpassung am Anfang des Trainings größer ist, und gegen Ende abnimmt. [vgl. [NKK94], [Roj93]]

Man kann nun ein *pattern* anstelle von p Inputprädikatoren mit $d < p$ relevanten Merkmalen charakterisieren und dieses für die tatsächliche Prognose in einem *feed-forward* Netz mit beispielsweise *back-propagation* einsetzen. [vgl. [Web00]]

7.3 Bestimmung der Prognoseunsicherheit

Man unterscheidet drei Verfahren zur Bestimmung der Konfidenzintervalle bei der Prognose mit neuronalen Netzen [vgl. [Web00]:

- Klassische asymptotische Methoden
- Verfahren der Bayes'schen Statistik

- *Resampling* Methoden

Da *resampling* Methoden im Fall von neuronalen Netzen am häufigsten zur Anwendung kommen, sollen sie in den folgenden Abschnitten kurz erläutert werden.

7.4 Resampling Methoden

Den *resampling* Verfahren ist gemein, dass aus einer Stichprobe systematisch neue Stichproben (mit oder ohne Zurücklegen) gezogen werden – daher auch der Name *resampling*. Zu den *resampling* Methoden zählen unter anderem der *bootstrap*-Algorithmus und der Jackknife-Ansatz. [vgl. [BD02]]

7.4.1 Bootstrap-Algorithmus

Beim *bootstrap*-Algorithmus werden aus den Trainingsdaten B unabhängige Stichproben (*bootstrap*-Stichproben) mit Zurücklegen gezogen, bis sie die gleiche Größe wie der ursprüngliche Datensatz haben – anhand dieser Stichproben kann jeweils der interessierende Parameter $\rho_{(b)}$ berechnet werden. Hat man schließlich für alle B Stichproben die Berechnung durchgeführt lässt sich die Verteilung der entsprechenden Statistik $\rho_{(.)}$ bestimmen. [vgl. [And97]]

Der Mittelwert der Verteilung der Schätzer ergibt sich zu:

$$\bar{\rho} = \frac{1}{B} \sum_{b=1}^{B} \rho_{(b)} \tag{71}$$

Die Varianz von $\rho_{(.)}$ ist wie folgt definiert:

$$s^2 = \frac{1}{B-1} \sum_{b=1}^{B} (\rho_{(b)} - \bar{\rho})^2 \tag{72}$$

Es lassen sich nun Prognoseintervalle bestimmen, indem die Verteilung der Prognosen für den Zeitpunkt $(T+1)$ erstellt wird. Im Normalfall benötigt man für eine ausreichend gute Abschätzung der Prognoseverteilung eine Ziehung von 50-200 Stichproben, wodurch ein hoher Rechenaufwand entsteht.

7.4.2 Jackknife-Ansatz

Beim Jackknife-Ansatz werden die ursprünglichen n Datensätze in k disjunkte Teilmengen der Größe h aufgeteilt, so dass gilt: $n = kh$. In der Regel wird $k = n$ gewählt – das heißt der vorhandene Datenbestand wird in so viele Teilmengen zerlegt, wie Datensätze vorhanden sind – jede Teilmenge besteht nur aus einem einzigen Datensatz. In diesem Spezialfall des Jackknife-Ansatzes spricht man auch von der *one-hold-out*-Methode. [vgl. [Pod94]]

Anschließend können sogenannte Pseudowerte nach folgender Formel berechnet werden:

$$y_{*j} = k y_{\text{all}} - (k-1) y_j \tag{73}$$

Dabei bezeichnet y_{all} die interessierende Größe für die gesamte Stichprobe und y_j den sich ergebenden Wert bei der Auslassung der j-ten Teilmenge. Die Pseudowerte werden k-mal berechnet, wobei jeweils eine andere Teilmenge bei der Berechnung unberücksichtigt bleibt. Der Jackknife-Schätzer ist folgendermaßen definiert:

$$y_* = \frac{1}{k}(y_{*1} + y_{*2} + \cdots + y_{*k}) \tag{74}$$

Für die Varianz gilt:

$$s^2 = \frac{\sum y_{*j}^2 - \frac{1}{k}(\sum y_{*j})^2}{k-1} \tag{75}$$

$$s_*^2 = \frac{s^2}{k} \tag{76}$$

Die Jackknife-Schätzer approximieren eine t-Verteilung mit $k-1$ Freiheitsgraden, so dass Konfidenzintervalle geschätzt werden können.

8 Gütekriterien für die Prognose

Trotz der besprochenen Möglichkeiten die Generalisierungsfähigkeit neuronaler Netze zu optimieren, werden immer Abweichungen zwischen den tatsächlichen und den durch das Netz ermittelten Werten bestehen. Es ist daher wichtig die Prognosegüte zu bestimmen. Man unterscheidet grundsätzlich zwischen *ex-ante* und *ex-post* Prognosen. Von *ex-post* Prognosen spricht man, wenn die Prognose für einen Zeitraum der Vergangenheit vorgenommen wird, von dem man die eingetretenen Zustände zum jetzigen Zeitpunkt bereits kennt. Eine *ex-ante* Prognose ist eine Vorhersage für die Zukunft – die tatsächlichen Werte sind daher in diesem Fall unbekannt. Es existiert außerdem der Begriff Pseudo *ex-ante* Prognose. Darunter versteht man Prognosen, wo auf einen Teil der vorliegenden Information verzichtet wird. Das heißt aus Sicht des Modells handelt es sich um eine *ex-ante* Prognose, obwohl die Daten für den Zeitraum für den prognostiziert wird bereits bekannt sind.

Außerdem kann eine Unterscheidung in eine vertikale und eine horizontale Prognoseverteilung der Prognosewerte getroffen werden. Betrachtet man die vertikale Prognoseverteilung, so können Prognoseintervalle anhand der Kovarianzmatrix der Modellresiduen geschätzt werden. Man sollte versuchen ein neuronales Netz möglichst sparsam zu parametrisieren, da sonst eine zu große Anzahl an redundanter Information die Kovarianzmatrix stark vergrößert. Es gilt, dass ein kleines Prognoseintervall eine bessere Prognosegüte liefert. Da die analytische Herleitung teilweise sehr komplex ist können *resampling*-Verfahren eingesetzt werden, um die Konfidenzintervalle zu approximieren (siehe dazu Kapitel 7.4). *Ex-ante* Prognosen können nur mit Hilfe von Konfidenzintervallen quantifiziert werden.

Bei der horizontalen Prognoseverteilung können Kennwerte berechnet werden, indem die prognostizierten Werte mit den bekannten, tatsächlichen Werten

verglichen werden. Da bei diesem Verfahren also beobachtete Werte vorliegen müssen ist sie ausschließlich für *ex-post* Prognosen einsetzbar. [vgl. u. a. [Web00]]

Eine detaillierte Beschreibung der auftretenden Fehler bei Fernsehprognosen und der einzusetzenden Gütemaße findet man in [Web00].

Man unterscheidet drei Arten der Fehlprognose [vgl. [Web00]]:

- Niveaufehlprognose: die Werte werden stets um einen konstanten Betrag über- bzw. unterschätzt.
- Abweichungsfehlprognose: die Werte werden sowohl im Vorzeichen, als auch im Betrag falsch geschätzt.
- Veränderungsfehlprognose: falsche Voraussage eines An- oder Abstiegs der Werte (Spezialfall: Wendekursfehlprognose).

Veränderungsfehlprognosen sind vor allem bei der Vorhersage von Aktienkursen von Interesse – für unseren Fall, der Prognose der Fernsehnutzung, sind in erster Linie Abweichungsfehler relevant.

Es gibt eine Reihe von Gütemaßen, die für die Bestimmung der Prognosegenauigkeit eingesetzt werden können. Verschiedene Gütemaße weisen unterschiedliche Charakteristika auf – daher reicht ein Gütemaß zur Beurteilung einer Prognose nicht aus. Die beste Möglichkeit ist die Verwendung eines Gütemaß-Mixes, um möglichst unterschiedliche Prognosequalitäten zu erfassen.

Allgemein gilt, dass die Gütemaße auf Daten angewendet werden sollen, die nicht zur Schätzung des Modells herangezogen werden. Es ist daher nötig einen Teil der Daten als Prognosemenge zur Bestimmung der Prognosgüte zurückzubehalten. Weiters ist eine Beschränktheit als Charakteristika der Gütemaße von Vorteil – sie erleichtert die Interpretierbarkeit und die Vergleiche verschiedener Prognosen miteinander.

Bei der Prognose von Reichweiten für Fernsehsender macht es keinen Unterschied ob die tatsächlichen Werte über- oder unterschätzt werden, daher sollte man Gütemaße wählen, die eine symmetrische Verlustfunktion besitzen. Weiters sollen Gütemaße mit quadratischen und linearen Verlustfunktionen zum Einsatz kommen, da es nicht plausibel begründbar ist, dass große Abweichungen stärker zu gewichten sind als kleine. Praktiker sind sich jedoch größtenteils einig, dass das Niveau der Daten eine Rolle bei der Prognose spielt – Fehlprognosen bei Sendungen mit einer hohen Nutzung sind weniger schwerwiegend, als Fehlprognosen bei Sendungen mit einer geringen Nutzung. Außerdem sollte man auch ein monotones Gütemaß zur Beurteilung heranziehen, um das Ausmaß der Unterschiede zweier Prognosen miteinander vergleichen zu können. Monotonie bedeutet, dass sich das Gütemaß um einen konstanten Faktor ändert, wenn sich die Differenz zwischen prognostiziertem Wert und gewünschtem Wert um einen konstanten Faktor verändert – die beiden Faktoren müssen dabei nicht gleich sein. [vgl. [Web00], [Web95]]

Unter den eben beschriebenen Gesichtspunkten sollen einige der wichtigsten Gütemaße für die Prognose der Fernsehnutzung vorgestellt werden:

1. Der mittlere absolute Fehler

$$MAE = \frac{1}{N} \sum \mid O_t - T_t \mid \tag{77}$$

Der MAE gibt an, inwieweit der vom Netz generierte *output* O_t im Durchschnitt vom gewünschten *target* T_t abweicht. N steht für die Anzahl der Beobachtungen. Es handelt sich beim MAE um ein symmetrisches Gütemaß – das heißt negative und positive Abweichungen werden gleich stark bewertet. Der MAE zeichnet sich außerdem durch Monotonie aus.

2. Der mittlere absolute prozentuale Fehler

$$MAPE = \frac{1}{N} \sum \mid \frac{O_t - T_t}{T_t} \mid * 100\,\% \quad [T_t \neq 0] \tag{78}$$

Der MAPE ist ein relatives (d. h. der Prognosefehler wird in Beziehung zum tatsächlichen Wert gesetzt), quantitatives Gütemaß und zeigt, um wie viel Prozent die Prognose im Mittel von den beobachteten Werten abweicht. Die Prognosefehler werden linear und abhängig vom Niveau der Beobachtungen bewertet. Ein Nachteil ist, dass zu niedrige Prognosen begünstigt werden.

3. Die Wurzel aus dem quadratischen mittleren Fehler

$$RMSE = \sqrt{\frac{1}{N} \sum (O_t - T_t)^2} \tag{79}$$

Der RMSE gibt ebenfalls an, inwieweit die prognostizierten Werte von den gewünschten Werten abweichen, ohne dass es zu einem Ausgleich zwischen positiven und negativen Abweichungen kommt – größere Abweichungen werden durch das Quadrieren jedoch stärker gewichtet als kleine. Dadurch kann es auch passieren, dass Ausreißer in den Daten den RMSE stärker beeinflussen, als dies erwünscht ist.

4. Bias-, Varianz- und Kovarianz-Anteil

$$BIAS = \frac{(\bar{T} - \bar{O})^2}{MSE} \quad [MSE \neq 0] \tag{80}$$

$$VAR = \frac{(s_T - s_O)^2}{MSE} \quad [MSE \neq 0] \tag{81}$$

$$KOV = \frac{2(1-R) s_T s_O}{MSE} \quad [MSE \neq 0] \tag{82}$$

wobei MSE den mittleren quadratischen Fehler angibt:

$$MSE = \frac{1}{N} \sum (O_t - T_t)^2 \tag{83}$$

und R für die Korrelation steht:

$$R = \frac{1}{N} \frac{\sum (O_t - \bar{O})(T_t - \bar{T})}{s_O s_T} \tag{84}$$

R ist ein korrelatives, quantitatives Gütemaß für die Stärke des Zusammenhangs zwischen den Prognosewerten und den Zielwerten. Der Korrelationskoeffizient lässt sich auch als die Wurzel des Bestimmtheitsmaßes definieren (siehe Kapitel 11). [vgl. [HS02]]

s_T und s_O stehen für die Streuungen der Zielwerte und der beobachteten Werte.

Der BIAS, die Varianz und die Kovarianz werden verwendet, um den RMSE hinsichtlich seiner systematischen und unsystematischen, also zufälligen, Fehlerkomponenten transparent zu machen. Ein hoher Bias-Anteil bedeutet, dass ein hoher Niveau-Fehler vorhanden ist – ist hingegen der Varianz-Anteil hoch, so handelt es sich um eine systematische Abweichungsfehlprognose. Das bedeutet, dass Schwankungen der Beobachtungswerte um den Mittelwert zwar erfasst werden, diese Erfassung jedoch zu stark oder zu schwach ist. Der Kovarianz-Anteil setzt sich zusammen aus $1-(VAR+BIAS)$ und gibt den Anteil des unsystematischen Fehlers an. Da sich die Kovarianz aus den anderen beiden Maßen ergibt reicht es aus, nur den Bias- und den Varianz-Anteil zu bestimmen. Alle drei Gütemaße sind auf den Bereich von Null bis Eins beschränkt. Eine gute Prognose ist dann erreicht, wenn BIAS- und Varianz-Anteil möglichst gering sind und der Kovarianz-Anteil nahe Eins liegt – das bedeutet, der Gesamtfehler resultiert nur noch aus einem unsystematischen Anteil.

5. Die Wurzel des mittleren quadratischen prozentualen Fehlers

$$RMSP = \sqrt{\frac{1}{N}\sum \frac{(O_t - \bar{T}_t)^2}{T_t^2}} * 100\,\% \tag{85}$$

Beim RMSP handelt es sich um ein kombiniertes Gütemaß (Kombination von quadratischem und relativem Gütemaß). Er gibt, genau wie der MAPE, an, um wie viel Prozent die prognostizierten Werte von den beobachteten abweichen. Der RMSP hat verschiedene Eigenschaften: er gewichtet große Werte stärker als kleine, die Gewichtung ist abhängig vom Niveau der Werte und seine Verlustfunktion ist symmetrisch.

6. Das Ball'sche Q^2

$$Q^2 = 1 - \frac{\sum(T_t - O_t)^2}{\sum(T_t - \bar{T}_{(A)})^2} \tag{86}$$

Q^2 gibt an, um wie viel die Ergebnisse der durchgeführten Prognose besser sind, im Vergleich zu einer naiven Prognose (z. B. arithmetisches Mittel, Median oder Modus). Da die Mittelwerte der Prognosedaten normalerweise im vorhinein nicht bekannt sind empfiehlt es sich, den Mittelwert der bekannten Daten in der Anpassungsmenge heranzuziehen. Negative Werte bedeuten ein schlechteres Abschneiden, Werte nahe Eins ein besseres Abschneiden der Prognose im Vergleich zur Trivialprognose.

9 Die Sensitivitätsanalyse

Neuronale Netze werden zunächst oft als *black-box* Verfahren bezeichnet, da man von der Größe der Gewichte nicht zwangsläufig auf die tatsächliche Bedeutung der einzelnen Prädikatoren für die Erzielung des *outputs* schließen kann. Es gibt jedoch Möglichkeiten mittels einer Relevanz- bzw. Sensitivitätsanalyse die Zusammenhänge innerhalb des Netzes ersichtlich zu machen. Das größte Interesse besteht meist darin herauszufinden, welchen Einfluss die Änderung einzelner Inputprädikatoren auf den *output* hat. [vgl. u. a. [Web00]]

Bei der Relevanzanalyse wird ein Relevanzmaß definiert, das angibt, wie sich der Fehler des Netzes ändert, wenn die betrachtete Variable durch ihren Mittelwert, bzw. einen anderen festgelegten Wert ersetzt wird:

$$\mathrm{REL}_i = \frac{\mathrm{MSE}_{x_i=\bar{x}_i}}{\mathrm{MSE}} \tag{87}$$

Je größer der Wert von REL_i ist, umso größer ist die Bedeutung des entsprechenden Prädikators. Bei $\mathrm{REL}_i = 1$ oder sogar darunter ist die Variable zur Findung der Lösung wenig relevant und kann entfernt werden. Anschließend muss das Netz erneut trainiert werden. [vgl. [And97]]

Eine andere Möglichkeit die Wirkung eines Inputprädikators auf den *output* sichtbar zu machen, besteht darin, den zu analysierenden Prädikator über seinen Definitionsbereich zu variieren, während sämtliche andere Prädikatoren konstant gehalten werden. Damit zeigt sich welchen Einfluss die Änderung eines *inputs* auf das Ergebnis nimmt.

Bei der Prognose der Fernsehnutzung kann die Anwendung der Sensitivitätsanalyse hilfreich sein, um die Auswirkung einzelner Prädikatoren auf die Reichweite abzuschätzen. Man kann die Sensitivitätsanalyse auch außerhalb der Trainings- oder Validierungsmenge einsetzen – dann spricht man von *case* Szenarien. Mittels *case* Szenarien lässt sich also untersuchen, wie gewisse Konstellationen von Inputvariablen den *output* beeinflussen. [vgl. [Web00], [Web01]]

10 Informationskriterien

Bei der Auswahl eines passenden Modells können auch sogenannte Informationskriterien zur Beurteilung herangezogen werden. Am bekanntesten sind dabei das Akaike-Informationskriterium (engl. *Akaike's information criterion* [AIC], 1973) und das Schwarz-Bayes'sche-Informationskriterium (engl. *Bayesian information criterion* [BIC], 1978). [vgl. [And97]]

Im Grunde setzen sich Informationskriterien aus zwei Teilen zusammen: ein Term für den Fehler des Netzes zu dem ein Term für die Netzwerkkomplexität strafend addiert wird. Ist das Netzwerk zu simpel, so ist der Fehler hoch und das Kriterium liefert einen hohen Wert. Für den Fall, dass das Netz zu komplex ist, wird aufgrund des Komplexitätsterms ebenfalls ein hoher Wert ausgegeben. [vgl. [Bis95a]]

$$AIC = \ln\hat{\sigma}^2 + \frac{2K_p}{T} \tag{88}$$

$$BIC = \ln\widehat{\sigma}^2 + \frac{K_p \ln(T)}{T} \tag{89}$$

K_p steht für die Anzahl der Parameter im Netz und T für die Anzahl der Beobachtungen. Es wird also der Fehler des Netzes gegen die Netzwerkkomplexität abgewogen – man erkennt, dass umso niedriger der Wert ist, den das AIC bzw. das BIC liefern, umso besser ist das Modell. Eine Erweiterung des Netzes durch Hinzunahme einer *hidden-unit* soll nur dann erfolgen, wenn dadurch der Fehler genügend minimiert wird – das heißt, wenn der Wert des Informationskriteriums entsprechend kleiner wird. Das BIC favorisiert Netze mit weniger Parametern – man erkennt, dass es im Vergleich mit dem AIC zusätzliche Parameter in Abhängigkeit der Anzahl der Beobachtungen stärker bestraft. [vgl. [And97]]

Im Grunde sind Informationskriterien nur unter der Voraussetzung einer asymptotischen Normalverteilung und bei korrekt spezifizierten Modellen anwendbar – allerdings hat sich in der Praxis gezeigt, dass die Kriterien auch bei Modellspezifikationen gute Ergebnisse erzielen. [vgl. [Web00]]

11 Selektionskriterien

Neben den im vorigen Kapitel beschriebenen Informationskriterien können zur Modellwahl auch Selektionskriterien zum Einsatz kommen. Das bekannteste und am häufigsten verwendete ist das Bestimmtheitsmaß R^2, das wie folgt definiert ist [vgl. [And97]]:

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{T}(y_t - \bar{y}_t)^2} \tag{90}$$

Das Bestimmtheitsmaß R^2 kann Werte im Bereich von Null bis Eins annehmen: die Anpassung der Regression ist umso besser, je näher R^2 bei Eins liegt und umso schlechter, je näher sich der Wert bei Null befindet. Ein Nachteil dieses Selektionskriteriums liegt darin, dass jeder zusätzlich eingeführte Parameter den Wert von R^2 erhöht und somit zu einer Verbesserung führt. [vgl. [And97]]

Beim justierten Bestimmtheitsmaß $\bar{R}^2$ wird dieses Problem behoben, indem die Anzahl der verwendeten Parameter in die Berechnung einfließt:

$$\bar{R^2} = 1 - \frac{T-1}{T-K}(1 - R^2) \tag{91}$$

T und K geben auch hier die Anzahl der Beobachtungen und die Freiheitsgrade im Netz an. Das $\bar{R}^2$ bestraft die Freiheitsgrade jedoch weniger stark als beispielsweise das BIC.

Ein weiteres Selektionskriterium stellt das Amemiyas *prediction criterion* (PC) (auch Akaikes *finite prediction criterion*) dar. Mit dem PC versucht man den zu erwartenden Prognosefehler abzuschätzen. [vgl. [And97]]

$$PC = \left(\frac{T+K}{T}\right) \frac{\sum_{t=1}^{T}(y_t - \hat{y}_t^2)}{T-K} \tag{92}$$

12 Begriffe aus der Medienforschung

In Österreich wird seit 1991 der sogenannte TELETEST zur Ermittlung der Reichweiten von Fernsehprogrammen im Auftrag des ORF vom unabhängigen Marktforschungsinstitut Fessel-GfK durchgeführt. Der Test ist folgendermaßen aufgebaut: ein Panel von 1 500 österreichischen Haushalten (3 537 Personen, ab einem Alter von drei Jahren) ist mit Telecontrol-Messgeräten ausgestattet, die die Nutzung des Fernsehgeräts personenbezogen erfassen. Dieses Panel ist repräsentativ für die 3 361 000 TV-Haushalte in ganz Österreich. Haushalte bezeichnet in diesem Zusammenhang alle Privathaushalte mit mindestens einem Fernsehgerät, deren Haushaltsvorstand die österreichische Staatsbürgerschaft besitzt. [vgl. [orf]]

Die folgenden Definition sind von [orf], [agf] und [med] übernommen. Man unterscheidet zwischen verschiedenen Arten von Reichweiten:

- Durchschnittsreichweite

  Die Durchschnittsreichweite wird auch als durchschnittliche Sehbeteiligung bzw. als *rating* bezeichnet. Sie gibt an wie viele Personen innerhalb eines gewissen Zeitabschnitts, einer Sendung oder eines Werbeblocks ferngesehen haben. Dazu wird der Quotient aus der tatsächlichen Sehdauer aller Personen zur gesamten Dauer der Sendung oder des Zeitintervalls gebildet.

$$\text{DRW in } \% = \frac{\text{tatsächliche Sehdauer aller Personen}}{\text{mögliche Sehdauer aller Personen}} * 100 \tag{93}$$

- Bruttoreichweite

  Die Bruttoreichweite in Prozent wird auch als *gross rating points* (GRPs) bezeichnet und stellt ein Maß für den Werbedruck dar (engl. *advertising impact*). Den GRP kann man auf zwei verschiedene Arten definieren.

$$\text{GRP in } \% = \text{Nettoreichweite in } \% * \text{Durchschnittskontakte} \tag{94}$$

  wobei der Durchschnittskontakt (engl. *opportunity to see*) angibt, wie oft eine Person durchschnittlich Kontakt mit einer Werbung hatte. Es werden dabei nur jene Personen erfasst, die mindestens einen Kontakt (Berührung mit dem Medium) hatten.

  Die zweite Möglichkeit der Definition von GRP ist:

$$\text{GRP} = \frac{\text{Bruttoreichweite}}{\text{Zielgruppenpotenzial}} * 100 \tag{95}$$

  Unter Zielgruppe versteht man eine Gruppe von Personen, die sich durch bestimmte Merkmale (z. B. soziodemografische Merkmale wie Alter, verfügbares Einkommen ect.) auszeichnet. Sendungen sowie Werbungen sind meist auf eine bestimmte Zielgruppe ausgerichtet.

Der GRP gibt also die Summe der Kontakte der Zielgruppe mit der Schaltung eines Mediaplans an. Dabei werden Mehrfachkontakte berücksichtigt (d. h. hat eine Person mehrfachen Kontakt wird jeder einzelne zu den GRPs hinzugerechnet). Der *cost per GRP* gibt die Aufwendungen an, die nötig sind um 1 % der Zielgruppe zu erreichen.

- Nettoreichweite

  Unter Nettoreichweite versteht man die Anzahl jener Personen, die in einem festgelegten Zeitabschnitt unter einem bestimmten Seherkriterium ferngesehen haben. Das Seherkriterium ist dabei abhängig von der Dauer der Sendung – bei einer Dauer von mehr als 10 Minuten muss der Kontakt mit dem Medium eine Minute ohne Unterbrechung gedauert haben, andernfalls eine Sekunde. Mehrfachkontakte treten in der Zählung nur einmal auf – sie finden also keine Berücksichtigung.

- Tagesreichweite

  Die Tagesreichweite gibt an wie viele Personen an einem Tag eine bestimmte Zeit lang (beim TELETEST eine Minute) durchgehend einen Sender gesehen haben. Mehrfachkontakte bleiben auch hier unberücksichtigt.

- Kumulierte Reichweite

  Die Kumulierte Reichweite gibt die Summe aller mit einer Sendung erzielten Kontakte ohne Berücksichtigung von Mehrfachkontakten an. Sie ist wie folgt definiert:

$$\text{Kumulierte Reichweite in \%} = \frac{\text{GRPs in \%}}{\text{Durchschnittskontakte}} \tag{96}$$

- Technische Reichweite

  Die Technische Reichweite gibt an, wie viele Haushalte innerhalb eines festgesetzten Gebiets theoretisch ein bestimmtes Medium (in unserem Fall Fernsehsender) empfangen können. Dabei bleibt jedoch unberücksichtigt, wie viele Haushalte tatsächlich ein Empfangsgerät, also einen Fernseher, besitzen. Die Technische Reichweite ist nicht zu verwechseln mit dem sogenannten Empfangspotenzial. Dieses gibt an, wie viele der Haushalte, die einen Fernseher besitzen den Sender tatsächlich empfangen können. Das Empfangpotenzial ist somit immer kleiner als die technische Reichweite, da hier eine kleinere Anzahl an Haushalten zugrunde gelegt wird.

Die Begriffe Reichweite, Einschaltquote und Marktanteil werden in der Umgangssprache häufig synonym verwendet. Es handelt sich dabei jedoch um drei verschiedene Parameter. Der Begriff Reichweite wurde in diesem Kapitel bereits erläutert. Unter Einschaltquote wird die durchschnittliche Sehbeteiligung der Haushalte verstanden – das heißt sie gibt an wie viele Fernsehgeräte während einer Sendung eingeschalten waren (ein Fernsehgerät gilt ab einer Dauer von einer Sekunde als eingeschaltet). Diese Größe ist also nicht personenbezogen.

Der Marktanteil wiederum gibt den relativen Anteil der Sehdauer eines Senders an der gesamten Sehdauer aller Sendungen in einem bestimmten Zeitintervall an. Das Zeitintervall kann dabei beliebig gewählt werden (Tag, Monat ...).

$$\text{Marktanteil eines Senders in \%} = \frac{\text{Nutzungszeit des Senders}}{\text{Nutzungszeit TV gesamt}} * 100 \qquad (97)$$

13 Simulatoren für neuronale Netze

Es gibt eine Vielzahl an sowohl proprietären (der Quellcode ist nicht zugänglich) als auch freien (der Quellcode wird zu Verfügung gestellt) Software-Simulatoren für neuronale Netze. Im folgenden Kapitel sollen einige der derzeit populärsten Simulatoren kurz vorgestellt werden – dabei werden die Unterschiede hinsichtlich der *key-features*, wie inkludierte Netzwerkarchitekturen, Lernalgorithmen und Möglichkeiten des Einlesens der Daten aufgezeigt. Es werden einige Netzwerktypen und Lernverfahren erwähnt, auf die in dieser Arbeit nicht näher eingegangen wird. Bei Interesse findet man nähere Beschreibungen dazu in der Literatur, beispielsweise in [Roj93], [Zel94a] oder [Bis95a]. Die aufgelisteten Simulatoren stellen jedoch nur einen kleinen Ausschnitt der am Markt erhältlichen *software* dar – die Auflistung ist daher keinesfalls als vollständig anzusehen.

13.1 Freie Software

Im folgenden Kapitel werden die Eigenschaften der Simulatoren Joone und SNNS bzw. JavaNNS erläutert. Für den praktischen Teil der Arbeit, die Prognose der Fernsehnutzung mittels neuronalen Netzen (siehe Kapitel 15), wurde hauptsächlich der Simulator JavaNNS eingesetzt (siehe Kapitel 13.1.2). Bei einigen Netzwerkspezifikationen kam außerdem SNNSv4.2 zum Einsatz. Ausschlaggebend für die Entscheidung diese *software* zu benutzen war die große Anzahl an inkludierten Lernalgorithmen sowie die ausführliche Dokumentation.

13.1.1 Joone-Java Objected Oriented Neural Engine

Joone ist ein auf JAVA basierender Simulator für neuronale Netze. Joone 1.2.1 ist auf der homepage gratis downloadbar. Die *software* kann sowohl für überwachtes, als auch für unüberwachtes Lernen eingesetzt werden. Inkludierte Netze sind unter anderem Elman-Netze, Jordan-Netze, *time delay neural networks* (TDNN), *feed-forward networks* (FFN) und selbstorganisierende Karten (SOMs). Als Lernalgorithmen stehen *on-line back-propagation, batch back-propagation* und Rprop zur Verfügung. Wie bei EasyNN-plus sind die *pattern* auch hier aus Excel Datein einlesbar. Das Projekt ist frei – das heißt jeder Benutzer hat die Möglichkeit die *software* selbst weiterzuentwickeln und neue Algorithmen oder Architekturen zu implementieren. [vgl. [joo]]

13.1.2 SNNS

SNNS ist die Abkürzung für *Stuttgarter Neural Network Simulator*. Das Programm wird am Institut für "Parallele und Verteilte Höchstleistungsrechner"

an der Universität Stuttgart unter der Leitung von Andreas Zell seit 1989 entwickelt. Es ist als *freesoftware* erhältlich und kann kostenlos von der homepage der Universität Stuttgart bezogen werden. Mittels SNNS ist es möglich neuronale Netze zu erstellen, zu trainieren, zu testen und auch zu visualisieren. Es sind eine Vielzahl an Lernalgorithmen inkludiert, wie beispielsweise *back-propagation*, Counterpropagation, Rprop und Quickprop. JavaNNS ist der Nachfolger von SNNS und wurde am "Wilhelm-Schickard-Institute for Computer Science" (WSI) der Universität Thübingen entwickelt – basierend auf dem Kernel von SNNS wurde ein benutzerfreundlicheres graphisches *user interface* in JAVA aufgesetzt. Die beiden Programme sind zum größten Teil kompatibel. [vgl. [snn]]

13.2 Proprietäre Software

13.2.1 EasyNN-plus

EasyNN-plus ist ein Neuronaler Netzwerksimulator der Firma Neural Planner Software für das Betriebssystem Windows, der auf EasyNN basiert. Die aktuelle Version EasyNN-plus 8.0c (*release date* 4.August 2006) ist auf der homepage für 30 Tage gratis herunterzuladen (*shareware*). EasyNN-plus ist für die Vorhersage, Klassifikation und Zeitreihenprognose anwendbar. *Multi-layer networks* können mit minimalem Benutzeraufwand aus importierten Datein (csv, Excel, plain text) generiert werden. Zusätzliche *features* sind unter anderem *early-stopping*, das Hinzufügen einer *noise* zum *input* und Möglichkeiten zum *pruning*. Außerdem können die Ergebnisse auf verschiedene Art graphisch veranschaulicht werden. [vgl. [eas]]

13.2.2 STATISTICA Neural Networks

STATISTICA Neural Networks ist eine Simulator, der von StatSoft, Inc. entwickelt wurde. Er ist vollständig integriert in das ebenfalls von StatSoft erstellte Software Programm STATISTICA. Es stellt eine Reihe von Netzwerkarchitekturen wie *multi-layer-perceptrons* (MLP), *radial basis functions* (RBF), SOMs ect., sowie unterschiedliche Lernalgorithmen – z. B. *back-propagation*, *quickpropagation*, Kohonen Training – zur Verfügung. Es enthält außerdem *features* wie das Training einzelner Netzwerksegmente oder eine Reduzierung der Inputdimension. [vgl. [sta]]

13.2.3 NeuroSolutions

NeuroSolutions ist ein Netzwerksimulator, der von NeuroDimension, Inc. entwickelt wurde. Die *software* lässt sich für Vorhersagen, Klassifikationen, Funktionsapproximationen und Clusterungen einsetzen. Es wird eine Vielzahl verschiedener Netzarchitekturen angeboten – darunter beispielsweise MLP, RBF, *time lagged recurrent networks* (TLRN), SOM's ect. Ein großer Vorteil von NeuroSolutions ist das graphische *user interface*, das die Möglichkeit bietet selbst Netzwerke zu erstellen. Das *developer level* erlaubt dem Benutzer außerdem

eigene Transferfunktionen und Lernmethoden in C++ zu programmieren. Dadurch erhält die *software* ein hohes Maß an Flexibilität.

Von NeuroDimension sind außerdem die Produkte NeuroSolutions for MATLAB und NeuroSolutions for Excel erhältlich. NeuroSolutions for MATLAB ist ein *tool*, dass die Möglichkeit bietet NeuroSolutions innerhalb von MATLAB einzusetzen. Es werden unter anderem MLP, *recurrent neural networks* und *generalized feed-forward networks* (GFFN) unterstützt, die mittels Quickprop, *back-propagation* mit Momentum oder *conjugate gradient* Verfahren trainiert werden können. NeuroSolutions for Excel ist ein *add-in* für Excel, mit dessen Hilfe der Benutzer neuronale Netze erstellen und trainieren kann ohne die Excel Umgebung verlassen zu müssen. [vgl. [dim]]

13.2.4 NeuroShell Predictor

NeuroShell Predictor ist ein Produkt der Ward System Group, Inc. Der Simulator arbeitet mit zwei verschiedenen Methoden – einerseits mit einem Algorithmus namens TurboCall 2 und andererseits mit einer erweiterten Variante von *general regression neural networks* (GRNN). Als Zielfunktionen können der RMSE, der MSE, maximale Korrelation ect. eingesetzt werden. Außerdem ermöglicht das Programm die Erstellung verschiedener Graphiken. [vgl. [neu]]

13.2.5 BrainMaker

BrainMaker ist eine *software* der Firma CaliforniaScientifics, die für das Betriebssystem Windows geeignet ist. Es gibt neben BrainMaker auch noch eine Professional Version, die mit zusätzlichen *features* (*case scenarios*, größere Anzahl an *units*) ausgestattet ist. Die Daten können mit einer Reihe unterschiedlicher Formate eingelesen werden (darunter wieder Excel und ASCII). BrainMaker bietet graphische Unterstützung und hat *data analysis features* inkludiert. [vgl. [bra]]

13.2.6 NeuroIntelligence

NeuroIntelligence ist ein Netzwerksimulator der Alyuda research company. Er stellt die bekanntesten Trainingsalgorithmen, wie *back-propagation*, Quickprop, *conjugate gradient descent* ect. zur Verfügung. Außerdem sind verschiedene Techniken, darunter beispielsweise *early stopping* und *jogging weights* inkludiert. Die Daten können sowohl aus Excel files als auch von ASCII-Formaten eingelesen werden. Es bietet die Möglichkeit die Resultate in Form von Graphen oder Tabellen auf einfache Weise zu visualisieren. [vgl. [int]]

14 Bisherige Prognosen bei Fernsehnutzung

In der Vergangenheit wurden verschiedene statistische Verfahren zur Vorhersage der Fernsehnutzung eingesetzt [Web00]:

- Modelle mit gewichteten Aggregatmittelwerten

- Lineare Modelle
- Nichtlineare univariate Verteilungs- und Saisonmodelle
- Nichtlineare multivariate Modelle
- Tree-Modelle
- Hybride Modelle

In den folgenden Abschnitten soll die Vorgehensweise der einzelnen Methoden kurz erläutert werden. Die Erwähnung konkreter Studien findet man in [Web00].

14.1 Modelle mit gewichteten Aggregatmittelwerten

Bei diesem Modell werden die arithmetischen Mittelwerte der Intensität der Fernsehnutzung für verschiedene Aggregate berechnet. Als Aggregate gelten beispielsweise Programme eines Monats, Programme eines bestimmten Genres ect. Soll nun für eine bestimmte Fernsehsendung die voraussichtliche Nutzung prognostiziert werden, so wird ein gewichtetes Mittel über alle Aggregate, denen die Sendung angehört, gebildet.

Die Höhe der Gewichtung muss aufgrund der Erfahrung eines Mediaplaners vorgenommen werden und ist daher zu einem großen Teil subjektiv. Ohne genaue Analyse der Daten kann es bei diesem Verfahren leicht zu Fehlschlüssen kommen. [vgl. [Web00], [BD02]]

14.2 Lineare Modelle

Es gibt die Möglichkeit die Fernsehreichweite, abhängig von einer einzelnen Variablen, beispielsweise der Zeit, zu prognostizieren. Dies geschieht mit Hilfe einer einfachen linearen Regression. [vgl. u. a. [HS02]]

$$\hat{y} = bx + c \tag{98}$$

wobei $\hat{y}$ die abhängige Variable (auch erklärte Variable) und x die unabhängige Variable (Prädiktor) beschreiben. Die Gewichtung b (Regressionskoeffizient) und die Regressionskonstante c werden mit der Methode der kleinsten Quadrate (*ordinary least squares*) ermittelt.

$$\sum (y_j - \hat{y}_j)^2 = \text{Min} \tag{99}$$

Die quadrierten Abstände zwischen den tatsächlichen Werten y_j und den geschätzten Werten $\hat{y}_j$ sollen minimiert werden.

Eine weitere Möglichkeit ist die Verwendung eines allgemeinen linearen Modells (ALM).

$$y = \mathbf{X}\beta + \varepsilon \tag{100}$$

Der Vektor y (abhängige Variable) hat dabei die Dimension Tx1 (mit t=1...T Beobachtungen) und $\mathbf{X}$ hat die Dimension TxK (wobei K die Koeffizienten und

nicht die Prädikatoren angibt). Daraus ergeben sich für β und ε die Dimensionen Kx1 bzw. Tx1.

Beim ALM geht man also davon aus, dass die zu prognostizierende Variable aus einer Linearkombination unabhängiger Variablen plus einem Fehler, der mögliche Zufallseinflüsse oder unberücksichtigte Variablen miteinbezieht, vorhergesagt werden kann.

Im klassischen ALM wird angenommen, dass der Fehler ε normalverteilt ist, mit konstanter Varianz und einem Erwartungswert von Null. Die Fehler zu verschiedenen Zeiten sollen außerdem voneinander unabhängig sein.

$$\epsilon \sim N(0, \omega_\epsilon^2 \mathbf{I}_T) \tag{101}$$

$\omega_\epsilon^2 \mathbf{I}_T$ gibt die Varianz-Kovarianz des Fehlers wieder – dabei steht $\mathbf{I}_T$ für die Einheitsmatrix der Dimension T. Wie bei der linearen Regression wird auch hier die Methode der kleinsten Quadrate angewendet:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{102}$$

Der Vorteil des ALM ist, dass Inputprädikatoren mit verschiedenen Skalenniveaus verwendet werden können, indem man beispielsweise nicht intervallskalierte Variablen *dummy*-kodiert. Für diesen Fall muss das ursprüngliche ALM angepasst werden, da Gleichung 101 in dieser Weise nicht bestehen bleiben kann. Die Anpassung geschieht durch die Einführung einer Transformationsmatrix $\mathbf{Z}$ in das ursprüngliche System.

René Weber hat in seiner Dissertation "Prognosemodelle zur Vorhersage der Fernsehnutzung" [Web00] die Prognose von Fernsehreichweiten und Reichweiten-Veränderungen mittels ALM, neuronalen Netzen, Tree-Modellen und hybriden Modellen untersucht. In diesem Kapitel sollen die Ergebnisse, die mit dem ALM erzielt wurden erläutert werden. Die anderen Prognoseverfahren werden in den Kapiteln 14.4, 14.5 und 14.6 beschrieben.

Die Prognose der Reichweite (in Tausend) für den Fernsehsender SAT1 erfolgt, basierend auf dem Programmgeschehen im Zeitraum 1.1.95 bis 30.4.97, für die Monate Mai/Juni 97 in der Zeit von 18:00 bis 23:00 Uhr. Als Zielgruppe dienen Erwachsene ab 14 Jahren (einige einfachere Analysen wurden auch für die Zielgruppe Erwachsene von 30-39 Jahren durchgeführt). Wir wollen an dieser Stelle nur die Ergebnisse der Reichweitenprognose für die Erwachsenen ab 14 Jahren anführen.

Nicht unproblematisch erwies sich die Identifikation eines geeigneten ALM. Aufgrund des umfangreichen Datensatzes konnte die Identifikation nur etappenweise durchgeführt werden, d. h. es wurde immer nur ein Teil der Inputprädikatoren berücksichtigt. Das optimale Modell schreibt das größte Gewicht den SAT1-Programmgenres innerhalb des aktuellen Programms, des Vorlauf- und des Nachlaufprogramms zu. Bei langfristigen Prognosen konnte ein MAPE von 28,8 %, ein R von 81,5 % und ein R^2 von 66,4 % erzielt werden. Bei den kurzfristigen Prognosen wurde ein niedrigerer MAPE erzielt (21,8 %), das R^2 sinkt jedoch auf 65,8 %.

In [Web01] wird die Prognose mittels einer einfachen Regression mit einer Prognose mit Hilfe von neuronalen Netzen verglichen (siehe dazu auch Ka-

pitel 14.4). Die Datenbasis wurde aus einer Arbeit von Gehrau übernommen ("Der Erfolg von Spielfilmen im Fernsehen. Eine Frage der richtigen Rezeptur?") – sie besteht aus 209 Spielfilmen die auf den Sendern ARD, ZDF, RTL, SAT1, PRO7, RTL2, KAB1 und VOX vom 30.3.96 bis 26.4.96 in der Zeit von 19:00 Uhr bis 24:00 Uhr ausgestrahlt wurden. Als Einflußgrößen für das Modell wurden die Inhaltsmerkmale Action, Spannung, Humor, Erotik und Anspruch (entnommen aus dem Rating dreier deutscher Programmzeitschriften) sowie der Sendebeginn, die technische Reichweite des Senders, der durchschnittliche Marktanteil im vorhergehenden Quartal und die Information, an welchem Wochentag die Sendung ausgestrahlt wurde (Freitag/Samstag oder Sonntag bis Donnerstag), gewählt. Als die beiden wichtigsten Prädikatoren bei der linearen Regression stellten sich der Marktanteil des Senders und der Sendebeginn mit β-Werten von 0,57 und $-0,32$ heraus. Bei den Inhaltsmerkmalen hatten nur die Prädikatoren Humor und Action einen relevanten Einfluss. Daher wurde ein Prognosemodell mit diesen vier Prädikatoren als optimales Modell eingesetzt. Im Schätzzeitraum und im Prognosezeitraum ergaben sich ein MAPE von 70,3 % bzw. 67 % und ein R^2 von 55,2 % bzw. 63 %. Der Grund für das schlechtere Abschneiden im Schätzzeitraum liegt an der größeren Anzahl an Aussreissern, die nur schlecht prognostiziert werden können. Im Vergleich mit den Ergebnissen, die mit neuronalen Netzen erzielt wurden (siehe dazu Kapitel 14.4) sieht man, dass die lineare Regression wesentlich schlechtere Ergebnisse erzielt.

14.3 Nichtlineare univariate Verteilungs- und Saisonmodelle

Bei Saisonmodellen liegen als Inputprädikatoren nur Informationen, die sich aus der Saison ergeben, zugrunde. Dazu gehören der Wochentag, der Jahrestag, die Tageszeit, das Monat und das Quartal. Zur Prognose verwendet man daher zeitreihenanalytische Methoden, wie beispielsweise die Fouriersynthese oder das ARIMA Modell.

ARIMA steht für *auto regressive integrated moving average model* – die Erweiterung für Saisonmodelle wird als SARIMA bezeichnet. Mit Hilfe eines ARIMA(p, d, q)-Modells kann die Systematik einer Zeitreihe vollständig beschrieben werden. Die Anzahl der autoregressiven Anteile in den Daten wird durch den Parameter p wiedergegeben, die Anzahl der Differenzierungen, die notwendig sind, um die Zeitreihe stationär zu machen gibt der Parameter d an und q gibt die Anzahl der Gleitmittelkomponenten wieder.

Ein ARIMA-Prozeß setzt sich aus einem autoregressiven Prozeß (AR(p)-Prozeß) und einem *moving-average*-Prozeß (MA(q)-Prozeß) zusammen. Um ein Zeitreihe beschreiben zu können, muss sie zumindestens schwach stationär sein. Das bedeutet, dass alle Zufallsvariablen denselben Mittelwert und dieselbe Varianz haben. Außerdem darf die Korrelation zwischen zwei Messpunkten nicht von der Position innerhalb der Zeitreihe abhängen. Der erste Schritt zur Bestimmung einer Zeitreihe durch einen ARIMA-Prozeß ist also die Überführung der urspründlichen in eine schwach-stationäre Zeitreihe. Dies kann durch eine einfache Differenzenbildung geschehen. Weiters müssen die mit Hilfe der kleinsten Quadrate Methode die Modellparameter geschätzt werden. Im Anschluß

findet eine Diagnose des Modells statt – hält das Modell dieser Prüfung stand, kann die Prognose anhand folgender Gleichung vorgenommen werden:

$$y^*_{t+1} = \varphi^*_0 + \varphi^*_1 y_t + \varphi^*_2 y_{t-1} + \cdots + \varphi^*_p y_{t-p+1} + \varepsilon^*_1 \gamma^*_t + \varepsilon^*_2 \gamma^*_{t-1} + \cdots + \varepsilon^*_q \gamma^*_{t-q+1} \quad (103)$$

y^*_{t+1} ist der geschätzte Wert für einen Zeitpunkt $(t+1)$, φ^* und ε^* sind die geschätzten Koeffizienten für den AR(p)-Prozeß bzw. den MA(q)-Prozeß. γ^* gibt die Störgröße zum Zeitpunkt (t) an, mit $\gamma^* = y_t - y^*_t$. [vgl. [Pod94]]

Die Fouriersynthese ist auch unter dem Begriff Spektralanalyse bekannt – die Betrachtung erfolgt nun nicht mehr im Zeitbereich, sondern im Frequenzbereich. Das bedeutet, dass die Zeitreihe in Form harmonischer Schwingungen dargestellt wird. Es handelt sich dabei um eine andere Art der Darstellung, die jedoch dieselbe Information enthält. Saisonalitäten in den Daten können als Kombination einzelner Sinus- und Cosinusschwingungen aufgefasst werden – diese lassen sich für die Prognose zukünftiger Reichweiten anwenden.

In [Web95] wurde sowohl mit Hilfe der Spektralanalyse, als auch durch den Einsatz eines ARIMA Modells versucht, die Einschaltquoten der "Tagesschau" in der ARD um 20:15 Uhr zu prognostizieren. Die Daten standen für den Zeitraum vom 1.1.85 bis 31.12.87 zur Verfügung. Die letzten drei Monate dienten als Validierungsmenge. Es wurden drei Prognosen durchgeführt: eine kurzfristige (10 Tage), eine mittelfristige (30 Tage) und eine langfristige (3 Monate). Die Reichweiten konnten bei der Spektralanalyse durch die ersten 5 bis 6 harmonischen Schwingungen recht gut erklärt werden. Für die kurzfristige Prognose konnte ein MAPE von 8,64 % und ein R^2 von 69 % erzielt werden. Mit dem SARIMA Modell konnte ein MAPE von 10,45 % und ein R^2 von 72 % erreicht werden. In den längerfristigen Prognosen lieferte das ARIMA-Modell gegenüber der Spektralanalyse bessere Ergebnisse.

Neben dem ARIMA Modell und der Spektralanlayse wurde auch noch ein Versuch der Prognose mit neuronalen Netzen unternommen. Eine kurze Erläuterung zu den Ergebnissen findet man im nächsten Kapitel.

14.4 Nichtlinare multivariate Modelle

Die Funktionsweise eines neuronalen Netzes wurde im theoretischen Teil der Arbeit bereits besprochen, weswegen in diesem Kapitel gleich einige vorhandene Studien zu dem Thema angesprochen werden sollen.

Interessante Ergebnisse, auch in Hinblick auf den praktischen Teil der Arbeit, liefert eine Studie aus dem Jahr 1996 von I. Garland (*"On the possibility of predicting ratings reliably"*), bei der die Fernsehnutzung in Sydney untersucht wurde. Dabei wurde ein *feed-forward*-Netz mit logistischen Funktionen eingesetzt. Als Determinanten wurden die Saisonalitäten der Reichweiten (Wochentag, Tageszeit) und die "Anzahl der fernsehenden Haushalte" herangezogen. Die Ergebnisse wurden nach 6 australischen Fernsehsendern differenziert – es stellte sich heraus, dass die Prognosen bezüglich zukünftiger Daten bei kleinen Fernsehsendern mit geringer Reichweite schlechter ausfielen. Ähnliches konnte auch im praktischen Teil dieser Arbeit beobachtet werden (siehe Kapitel 15). Für kleine Sender wie GoTV, MTV oder PulsTV, die im Allgemeinen sehr

niedrige Reichweiten haben, konnten bei verschiedensten Netzkonfigurationen keine brauchbaren Ergebnisse erzielt werden, so dass auf eine Prognose mittels neuronaler Netze verzichtet werden musste.

Einige wichtige aktuellere Untersuchungen zum Thema Fernsehnutzung mit Hilfe von neuronalen Netzen stammen von René Weber. An dieser Stelle sollen zwei seiner Arbeiten kurz vorgestellt werden.

1. Aufbauend auf einer Studie zum Erfolg von Spielfilmen im deutschen Fernsehen (Gehrau 1997), die mittels Regressionsanalyse durchgeführt wurde, untersucht René Weber dieselben Daten mit dem Einsatz von neuronalen Netzen. Derselbe Datensatz wurde außerdem mittels linearer Regression untersucht – die Ergebnisse dazu wurden bereits in Kapitel 14.2 aufgezeigt. [vgl. [Web01]]

   Die Datenbasis besteht aus 209 Spielfilmen, die auf den Fernsehkanälen ARD, ZDF, RTL, SAT1, PRO7, RTL2, KAB1 und VOX im Zeitraum vom 30.3.96 bis zum 26.4.96 mit Sendebeginn zwischen 19:00 und 24:00 Uhr ausgestrahlt wurden. Der Erfolg der Spielfilme wurde mittels der Reichweite der Zuschauer ab 3 Jahren gemessen. Weber hat die Daten in einen Trainingsdatensatz (101 Spielfilme), einen Validierungsdatensatz (47 Spielfilme) und einen Prognosedatensatz (61 Spielfilme) unterteilt. Es wird ein *feed-forward*-Netz mit einem *hidden-layer* verwendet. Als Lernalgorithmus dient der *back-propagation* Algorithmus. Nach der genetischen Topologieoptimierung erweist sich ein Netz, bestehend aus 5 *input-*, 9 *hidden-* und einer *output-unit* als optimal. Als Prädikatoren für die *input-units* dienen der Marktanteil des Senders, die Sendezeit sowie die Inhaltsvariablen Humor, Action und Spannung (entnommen aus drei deutschen Programmzeitschriften). Mit dem neuronalen Netz erhielt man für den Schätzzeitraum einen MAPE von 39,7 % und ein R^2 von 78,7 %. Für den Prognosezeitraum ergab sich ein MAPE von 44,7 % und ein R^2 von 78,5 %. Es zeigt sich, dass das neuronale Netz in beiden Zeiträumen bessere Ergebnisse erzielt als die lineare Regression. Die Ergebnisse der linearen Regression sind in Kapitel 14.2 zu finden.

2. In [Web95] wurde eine Prognose der "Tagesschau" für den Sender ARD um 20:15 Uhr unter anderem auch mit neuronalen Netzen durchgeführt (zu den anderen Modellen siehe Kapitel 14.2).

   Als Lernverfahren kam die konjugierte Gradientenmethode mit *batch-learning* und ausschließlich sigmoiden Aktivierungsfunktionen zum Einsatz. Die besten Netze wurden mit Hilfe einer Kreuzvalidierung bestimmt – jenes Netz mit dem geringsten RMSE bei der Prognose wurde als ideales Netz angenommen. Außerdem wurde ein Multi-Start-Verfahren angewendet, um einem "gefangen" werden in einem Minimum vorzubeugen.

   Es hat sich gezeigt, dass Netze mit sehr kleinen *hidden-layern* eine schlechtere Prognose liefern, aber kaum Unterschiede hinsichtlich des RMSE bei Netzen mit 8 bis 20 *hidden-units* zu bemerken sind. Eine Vergrößerung des *input-layers* verbessert die Prognose.

Die besten Netze für die mittel- und langfristige Prognose wurden aus den Prognosenetzen für die kurzfristige Prognose abgeleitet, da aus Kapazitätsgründen eine umfassende Netzsuche nicht möglich war. Bei der kurzfristigen Prognose lieferte ein 30-8-10 Netz das beste Ergebnis (RMSE von 1,91 %), bei der mittelfristigen ein 7-10-30 Netz (RMSE von 2,73 %) und bei der langfristigen ein 7-7-92 Netz (RMSE von 2,82 %). Der MAPE und das R^2 wurden nur für die kurzfristige Prognose angegeben – sie betragen hier 15,06 % und 10 %. Im Vergleich mit den anderen Verfahren hat das neuronale Netz hier am schlechtesten abgeschnitten und war auch nicht deutlich besser als die Trivialprognose (MAPE von 18,46 %).

3. Wie bereits in Kapitel 14.2 erwähnt, hat René Weber in [Web00] die Vorhersage der Fernsehnutzung mit dem Einsatz verschiedener Prognosemodelle untersucht. Im folgenden sollen die erzielten Ergebnisse, in Hinblick auf die neuronalen Netze dargestellt werden. Die Prognose mit ALM wurde bereits in Kapitel 14.2 beschrieben. Auf die Prognosen mittels hybriden Modellen wird in Kapitel 14.6 näher eingegangen.

   Die Prognose der Reichweite (in Tausend) für den Fernsehsender SAT1 erfolgt, basierend auf dem Programmgeschehen im Zeitraum 1.1.95 bis 30.4.97, für die Monate Mai/Juni 1997 in der Zeit von 18:00 bis 23:00 Uhr. Als Zielgruppe dienen Erwachsene ab 14 Jahren (einige einfachere Analysen wurden auch für die Zielgruppe Erwachsene von 30-39 Jahren durchgeführt).

   Es werden verschiedene originäre, abgeleitete und ergänzende Variablen im Modell berücksichtigt. Zu den originären Variablen zählen Variablen zur zeitlichen Identifikation (Datum, Beginnzeit/Endezeit der Sendung, Dauer der Sendung), die Art des Programms (zusammengefasst in einer Gesamt-Genre-Variablen, die die Variablen Programmsparte, Sendungsform, Non-Fiction-Thema und Fiction-Thema enthält) und Fernsehkanäle (ARD, ZDF, SAT1, RTL, PRO7, KAB1, VOX und RTL2).

   Zu den abgeleiteten Variablen zählen das Programmumfeld (Konkurrenzprogramm, Vorlauf- und Nachlaufprogramm und konkurrierende Vorlauf- und Nachlaufprogramme), Variablen zur Erfassung saisonaler Effekte (Gewichtungsvariablen), Attraktivität der Programme (residuenbasierte Attraktivitätsindizes), Positionsvariablen (absolutes/relatives Positionsmaß, Sendungsteil-Nummer und Gesamtanzahl der Sendungsteile, um Effekte aufgrund von Werbeunterbrechungen zu erfassen) sowie Interaktionen (Dauer x Beginn, Position der Minute x Art des Programms).

   Als ergänzende Variablen werden noch besondere Ereignisse (sportliche Ereignisse, Beginn/Ende der Sommerzeit) und Feiertage sowie Schulferienzeiten berücksichtigt.

   Zur Identifikation eines optimalen Prognosenetzes wurde eine genetische Optimierung vorgenommen: *selection* (50 % der besten Netze), *refill* (Zufallsauswahl und *cloning*), *crossover* (*one-cut-swap*) und Mutation (*random-exchange* und *selection-reverse*). Als Basistopologie wurde ein *feed-forward*-Netz mit jeweils einem *input-, hidden-,* und *output-layer* gewählt.

Als Lernverfahren wurde der *back-propagation* Algorithmus in Verbindung mit dem "Quasi-Newton-Verfahren" eingesetzt. Die Daten wurden zu Beginn einer z-Transformation unterzogen und mittels Multi-Start-Verfahren in Form von *batch-learning* dem Netz präsentiert. Zur Vermeidung des *overfitting* wurde ein *weight decay* sowie eine Kreuzvalidierung in Verbindung mit einem *stopped training* eingesetzt.

Als optimales Netz für die Prognose der Reichweite hat sich ein Netz erwiesen, das aus 68 *input-units*, 54 *hidden-units* und einer *output-unit* besteht. Der *hidden-layer* setzt sich aus 21 Neuronen mit einer logistischen, aus 18 Neuronen mit einer tangentialen und aus 15 Neuronen mit einer linearen Aktivierungsfunktion zusammen.

Mittels Sensitivitätsanalysen wurden für das Netz jene Prädikatoren bestimmt, die den größten Einfluss auf das Ergebnis haben. Bei der Prognose der Reichweite waren vor allem das SAT1-Programm, das SAT1-Vorlaufprogramm, die SAT1-Sendungsteilnummer, der Wochentag und die Beginnzeiten ausschlaggebend. Außerdem stellte sich heraus, dass die ARD und RTL die stärksten Konkurrenzprogramme sind.

Mittels eines Gütemaß-Mixes wurde ein Vergleich zwischen dem linearen Modell, dem Tree-Modell und dem hybriden Modell vorgenommen. Dabei stellte sich heraus, dass ein neuronales Netz mit genetischer Topologieoptimierung und einer inhaltlich plausiblen Prädikatorenauswahl zu sehr guten Vorhersagen für die Reichweiten führte. Bei langfristigen Prognosen wurde ein MAPE von 25,7 % und ein R^2 von 75,8 % erreicht, bei kurzfristigen Prognosen ein MAPE von 19,3 % und ein R^2 von 69,8 %. Gleich gute, teils bessere Ergebnisse konnten nur mit hybriden Modellen erzielt werden (näheres dazu findet man in Kapitel 14.6).

14.5 Tree-Modelle

Tree-Modelle teilen einen Datensatz anhand von Entscheidungsregeln in Subgruppen. Die Unterteilung (*split*) erfolgt an sogenannten Knoten – der erste Knoten wird auch *root* genannt, die untersten Knoten heißen *leaves*.

Bei der Generierung eines optimalen *trees* wird jede Prädikatoren-Subgruppe in zwei Subgruppen unterteilt, so dass die Differenz der Abweichungen maximal ist. Wie bei neuronalen Netzen gibt es auch beim Tree-Modell Möglichkeiten zur Optimierung der Struktur. Beispielsweise kann eine weitere Aufteilung eines Knoten nur dann vorgenommen werden, wenn dadurch die Modellanpassung merklich verbessert wird – diese Methode stellt ein Pendant zum *weight decay* bei neuronalen Netzen dar. Es gibt auch die Methode des *shrinking*, bei der eine Glättung der Datenwerte und eine Übertragung auf die vorhergehenden Knoten stattfindet.

Einige Vorteile von Tree-Modelle sind, dass sie keine Vorverarbeitung der Inputdaten erfordern, ihre Unempfindlichkeit gegenüber fehlenden Daten und die gleichzeitige Einbeziehungen nominal- und intervallskalierter Merkmale. Ein Nachteil ist hingegen, dass Interaktionseffekte zwischen einzelnen Determinanten in einem Zweig des *trees* zwar lokal erfasst werden können, eine globale

Erfassung aber oft schwierig ist.

Die Prognose in einem Tree-Modell wird durchgeführt, indem für bekannte Prädikatoren ein Pfad des *trees* durchlaufen wird – die *leaves* geben schließlich die Prognose der abhängigen Variablen an. Nähere Beschreibungen zum Verfahren sind in [Web00] zu finden.

In [Web00] wurde, wie bereits erwähnt, auch versucht die Fernsehreichweiten des Senders SAT1 mit dem Einsatz eines Tree-Modells zu prognostizieren. Der dabei verwendete Datensatz wurde bereits in Kapitel 14.4 beschrieben und wird an dieser Stelle daher nicht noch einmal erläutert.

Bei der Erstellung des optimalen Modells wurde darauf geachtet, dass jeder Hauptknoten mindestens 10 und jeder Endknoten mindestens 5 Beobachtungen enthält. Als optimales Modell ergibt sich ein *tree* mit 203 Pfaden. Aufgrund der Aufteilung, kann man auf die Bedeutung der einzelnen Prädikatoren rückschließen. Die *splits* im oberen Bereich des *trees* haben die größte Relevanz – steigt man in dem *tree* herab, so nimmt auch die Wichtigkeit der *splits* ab.

Der erste Knoten teilt das Modell anhand des Programmangebots des Senders ARD – für die Prognose des SAT1 Programms wird eine Aufteilung in ein Programm vor der 20-Uhr-Tagesschau und nach der 20-Uhr-Tagesschau vorgenommen. Der MAPE bei den kurzfristigen Prognosen liegt bei 25,4 % und das R^2 bei 52,1 %. Für langfristige Prognosen liefert das Tree-Modell einen MAPE von 39,6 % und ein R^2 von 61 %. Im Vergleich mit dem ALM, den neuronalen Netzen und dem hybriden Modell (mehr dazu im folgenden Kapitel) prognostiziert das Tree-Modell am schlechtesten.

14.6 Hybride Modelle

Unter hybriden Modellen versteht man Modelle, die die Ergebnisse verschiedener anderer Modelle heranziehen um Prognosewerte zu definieren. Dadurch, dass unterschiedliche Prognosemodelle auch unterschiedliche Variablen berücksichtigen bzw. Variablen unterschiedlich berücksichtigen, kann es durch eine Kombination verschiedener Modelle zu Synergieeffekten kommen. Die wichtigste Frage, die sich dabei stellt ist welche Prognosemodelle zusammen verwendet werden sollen.

Die Kombination der Prognoseverfahren erfolgt meist über die gewichtete Summe der Mittelwerte der Einzelprognosen. Es besteht jedoch auch die Möglichkeit eine einfache Kombination der Verfahren vorzunehmen, indem das arithmetische Mittel gebildet wird.

Die einfache Kombination von Prognoseverfahren hat R. Weber in [Web95] getestet. Es wurden die Prognoseergebnisse des SARIMA-Modells, eines neuronalen Netzes und des Frequenzmodells durch Bildung des arithmetischen Mittels kombiniert (zu den Ergebnissen der Einzelprognosen siehe Kapitel 14.3 und Kapitel 14.4). Bei einer kurzfristigen Prognose (10 Tage) wurde ein MAPE von 10,2 % und ein R^2 von 64 % erreicht. Damit ist das hybride Modell dem Frequenzmodell unterlegen. Bezüglich mittelfristiger Prognosen (30 Tage) konnten mit dem hybriden Modell jedoch die besten Ergebnisse erzielt werden. Langfristig war zwar das SARIMA-Modell nicht zu übertreffen – doch auch hier hat das hybride Modell durchaus gute Prognosen geliefert.

In seiner Dissertation [Web00] hat Weber erneut Fernsehreichweiten mit hybriden Modellen prognostiziert. Es wurden dazu wiederum drei Modelle miteinander verknüpft: neuronale Netze, Tree-Modelle und ALM (zu den Einzelergebnissen wird auf die Kapitel 14.4, 14.5 und 14.2 verwiesen). Einmal wurde eine Kombination der Modelle in Form eines einfachen arithmetischen Mittels und einmal über gewichtete Mittel vorgenommen. In beiden Versuchen war die Kombination aller drei Modelle erfolgreicher als eine Kombination von nur zwei Modellen. Die Versuche mit gewichteten und ungewichteten Mittelwerten ergaben im Prognosezeitraum ähnliche Ergebnisse – als optimales hybrides Modell wurde ein Modell mit ungewichteten arithmetischen Mitteln gewählt, da sich dabei die Berechnung einfacher gestaltet. Das hybride Modell liefert bessere Ergebnisse als die Einzelprognosen. Bei der kurzfristigen Prognose wurde ein MAPE von 19,3 % und ein R^2 von 83,5 % erreicht, langfristig ein MAPE von 25,7 % und ein R^2 von 87,1 %.

15 Beispiel: Prognose von Fernsehreichweiten mittels neuronaler Netze

15.1 Einleitung

Im praktischen Teil der Arbeit wurde versucht mit neuronalen Netzen die Reichweite von Werbesendungen zu prognostizieren. Es wurden die Sender ORF1, ORF2, ORF, ATV, RTL, SAT1, PRO7, RTL2, VOX, SRTL, KAB1, GoTV, MTV und PulsTV untersucht. Der ORF umfasst jene Werbungen, die in der Zeit von 19:13 bis 20:37 zeitgleich auf den Sendern ORF1 und ORF2 ausgestrahlt wurden – im weiteren Verlauf dieser Arbeit werden die Daten dieses Zeitraums, wie die Daten eines Senders behandelt, weswegen ORF der Einfachheit halber auch als Sender bezeichnet wird. Sämtliche für das Training verwendete Daten stammen aus dem Zeitraum vom 1.1.05 bis 30.4.06. Zur Prognose wurden zusätzlich Daten vom 1.5.06 bis 31.7.06 hinzugezogen. Die Prognosen beziehen sich auf die Reichweiten bei der Zusehergruppe der 12-49-jährigen.

Für jeden Sender wurde ein individuelles Netz trainiert, mit Ausnahme von ORF1, ORF2 und ORF. Diese Sender wurden in einem Netz zusammengefasst. Als Inputprädikatoren wurden bei allen Sendern die Temperatur (eine *unit*), der Wochentag (Dummy-Variable, 7 *units*), das Monat (Dummy-Variable, 12 *units*) und die Information, ob es sich um einen Feiertag handelt oder nicht Dummy-Variable, eine *unit*) verwendet. Ebenfalls geht die Beginnzeit des Werbespots als *input* ein – die Anzahl dieser Dummy-Variablen ist jedoch bei den einzelnen Sendern unterschiedlich, da verschiedene Zeitintervalle prognostiziert werden. Für sämtliche Sender, abgesehen von ORF1, ORF2 und ORF, wurden stündliche Intervalle für die Beginnzeiten verwendet. Bei den österreichischen Sendern wurden die Intervalle auf 15 Minuten reduziert.

Es wurden ausschließlich 3-schichtige Netze mit *forward connection* verwendet. Beim Testen stellte sich heraus, dass die Netze durchwegs bessere Ergebnisse bei der Prognose erzielen, wenn zusätzlich *shortcut connections* zwischen dem ersten und dem dritten *layer* eingebaut werden. Die im folgenden angeführten Ergebnisse beziehen sich daher nur mehr auf Netze mit drei *layern, forward connection* und *shortcuts* (vergleiche hierzu Kapitel 2). Außerdem hat sich, wie erwartet, herausgestellt, dass die Aktivierung der Funktion "Shuffle" in JavaNNS (die Daten werden dabei in jedem Zyklus in einer anderen Reihenfolge präsentiert, um ein Auswendiglernen zu verhindern) die Ergebnisse verbessert. Folglich wurde jedes Training mit "geshuffelten" Daten durchgeführt.

Um das für den jeweiligen Sender beste Netz zu identifizieren, wurde der Datenbestand in drei disjunkte Teilmengen zerlegt: eine Trainingsmenge (zufällig gewählte 70 % der Daten), eine Validierungsmenge (zufällig gewählte 20 %) und eine Prognosemenge (zufällig gewählte 10 %).

Die besten Ergebnisse wurden bei einem Training mittels *resilient propagation* erzielt (siehe dazu Kapitel 4.4), weswegen sämtliche im weiteren angeführten Ergebnisse aufgrund eines Trainings mit diesem Verfahren zustande kamen. Es wurden die in JavaNNS *per default* gesetzten Parameter beibehalten: Δ_0 (Startwert für alle Δ_{ij}) ist 0,1, $\Delta_{\max}$ (die obere Schranke der *update values*) ist 50 und α (*weight decay parameter*, der im Exponenten steht)

ist 4. Als Aktivierungsfunktion wurde die in JavaNNS standardmäßig verwendete Funktion Act_Logistic (Gleichung 104) beibehalten, als Ausgabefunktion die Identity_Function. Die Gewichte wurden vor Beginn des Trainings mit Zufallszahlen zwischen -1 und 1 initialisiert. Versuche mit Verwendung einer Tangens-Hyperbolicus Funktion als Aktivierungsfunktion haben die Prognosen eher verschlechtert. Ebenso konnten mit anderen Einstellungen der einzelnen Parameter keine merklichen Verbesserungen erzielt werden.

$$a_j = \frac{1}{1 + e^{-sum_j - bias_j}} \tag{104}$$

Da Act_Logistic nur Outputwerte im Bereich zwischen Null und Eins liefert mussten die Reichweiten bei einigen Sendern vor dem Training skaliert werden und nach der Prognose wieder rückskaliert.

Als Gütekriterien der trainierten neuronalen Netze wurden die Korrelation R, das Gütemaß R^2, sowie der MAPE (*mean average percentage error*) herangezogen (siehe dazu Kapitel 8). Sie wurden immer für die Prognosemengen berechnet, die nicht zum Training eingesetzt wurden und somit neue Daten für das Netz darstellen.

15.2 Ergebnisse

In den folgenden Kapiteln sollen die Ergebnisse, die bei der Prognose mit neuronalen Netzen erzielt wurden erklärt werden. Da, wie bereits in der Einleitung erwähnt, für die verschiedenen Sender jeweils eigene Netze trainiert wurden, sind die Ergebnisse auch nach Sendern unterteilt.

Für jeden Sender wird zu Beginn ein Überblick über die verwendeten Datensätze gegeben. Anschließend wird der Netzaufbau erklärt und die besten Ergebnisse, die erzielt wurden aufgezeigt.

Eine nähere Analyse wurde für die Sender ORF, ORF1 und ORF2 vorgenommen, da hier die besten Ergebnisse entstanden.

15.2.1 ORF1, ORF2 und ORF

Ursprünglich wurde für jeden der drei Sender, wie für alle anderen Sender, jeweils ein eigenes Netz generiert. Da die Sender aber durch die ORF-Daten verknüpft sind wurden sie schließlich in einem Netz zusammengefasst.

Der Sender ORF umfasst, wie bereits erwähnt, jene Werbungen, die in der Zeit zwischen 19:13 Uhr und 20:37 Uhr auf den Sendern ORF1 und ORF2 zeitgleich ausgestrahlt werden. Der Datensatz besteht aus 912 Einträgen, davon 219 aus dem Jahr 2006.

Die Abbildungen 15, 16, 17 zeigen die Reichweiten des ORF abhängig vom Monat, der Temperatur und dem Wochentag. Man erkennt einen Rückgang der Reichweiten im Sommer, was mit einem Rückgang der Reichweiten bei steigenden Temperaturen einhergeht. Weiters lässt sich ein Abfall der Fernsehnutzung von Montag bis Freitag erkennen. Die größten Streuungen sind am Wochenende zu finden.

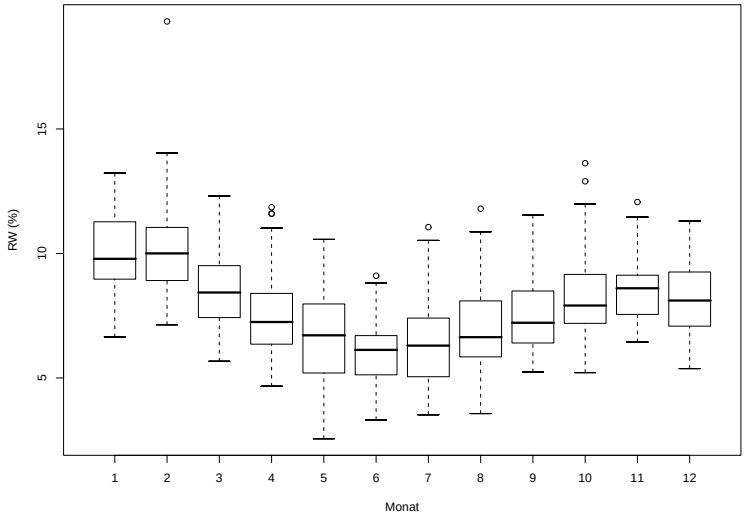


Abbildung 15: Reichweiten vom Sender ORF abhängig vom Monat

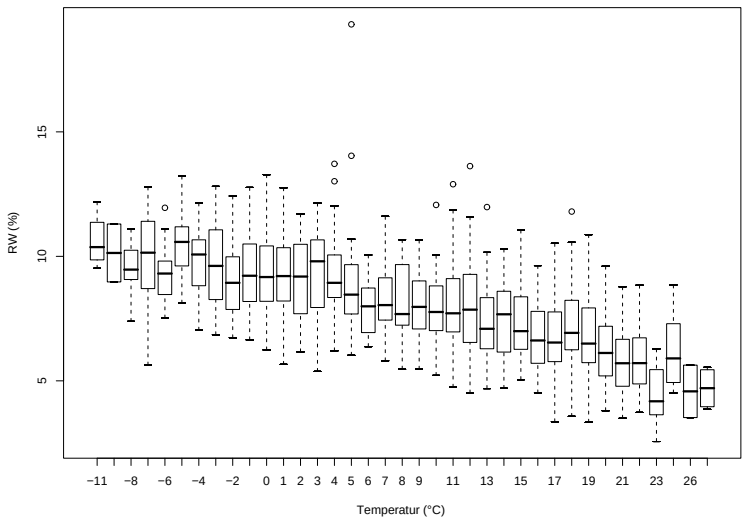


Abbildung 16: Reichweiten vom Sender ORF abhängig von der Temperatur

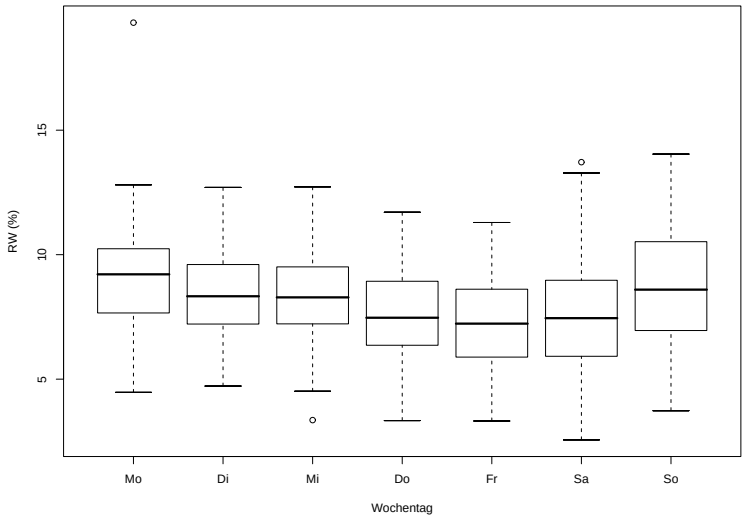


Abbildung 17: Reichweiten vom Sender ORF abhängig vom Wochentag

Der Datensatz des Senders ORF1 besteht aus 6 031 Einträgen, wobei 1 562 aus dem Jahr 2006 stammen. Auf die Beginnzeiten von 1:00 Uhr bis 9:00 Uhr entfallen lediglich 43 der vorhandenen Daten, wobei für die Beginnzeiten zwischen 2:00 Uhr und 4:00 Uhr überhaupt keine Messungen vorliegen. Da dies einen ziemlich kleinen Anteil am gesamten Datenbestand darstellt und außerdem für eine aussagekräftige Prognose zu wenige Trainingsbeispiele liefert, wurden diese 43 Daten entfernt.

Die Verteilung der Reichweiten abhängig vom Monat, der Temperatur, der Beginnzeit und dem Wochentag sind in den Abbildungen 18, 19, 20 und 21 zu finden. Auch hier erkennt man eine Abnahme der Reichweiten im Sommer – sie ist allerdings nicht so stark ausgeprägt, wie beim Sender ORF. Abhängig von der Ausstrahlungszeit des Werbespots sieht man zwei *peaks* – einerseits um die Mittagszeit von 11:00 bis 13:00 Uhr und andererseits am Abend von 19:00 bis 22:00 Uhr. Die Streuung ist jedoch zu Mittag am größten.

Der Datensatz des Senders ORF2 besteht aus 5 662 Einträgen, davon 1 436 aus dem Jahr 2006. Für die Beginnzeiten von 1:00 Uhr bis 9:00 Uhr sind keine Daten vorhanden, ebenso nicht für die Beginnzeiten zwischen 10:00 Uhr und 11:00 Uhr und 12:00 Uhr bis 13:00 Uhr. Da lediglich zwei Werte für den Beginn zwischen 9:00 Uhr und 10:00 Uhr sowie zwischen 11:00 Uhr und 12:00 Uhr angegeben sind, wurden diese 4 Einträge aus der Datenmenge eliminiert.

Die Abbildungen 22, 23, 24, 25 zeigen wiederum die Reichweiten abhängig vom Monat, der Temperatur, der Beginnzeit und dem Wochentag. Die höchsten Reichweiten sind am Abend zwischen 19:00 Uhr und 22:00 Uhr zu finden – allerdings ist um diese Zeit auch die größte Streuung in den Daten erkennbar. Der Ausreißer, von 12 % wurde am 19.4.05 gemessen und kam durch die Papstwahl zustande. Man sieht auch, dass beim Sender ORF2 weniger Unterschiede in der Reichweite bezüglich der Wochentage vorliegen.

In Summe sind also 12 558 Datensätze vorhanden – davon bilden 8 790 die Trainingsmenge, 2 511 die Validierungsmenge und 1 257 die Prognosemenge. Die Daten wurden für den Zeitraum von 10:00 Uhr bis 1:00 Uhr prognostiziert – damit ergeben sich 15 Dummy-Variablen für die Beginnzeiten. Außerdem geht der Sender, auf dem die Werbung ausgestrahlt wird als Prädikator ein – dafür werden 3 Dummy-Variablen benötigt. Die ersten Prognoseversuche wurden mit stündlichen Intervallen durchgeführt. Es hat sich aber gezeigt, dass sich dadurch vor allem beim Sender ORF Probleme ergaben. Beim ORF gibt es zwei Werbeblöcke – einer wird vor der "Zeit im Bild" (ZIB) ausgestrahlt, der andere vor dem Wetterbericht. Diese beiden Werbeblöcke liegen innerhalb einer Stunde und man würde somit für beide Blöcke eine Prognose erhalten. Die Reichweiten variieren in Wirklichkeit jedoch stark, wie wir bei der detaillierten Betrachtung der Ergebnisse noch sehen werden. Um dieses Problem zu minimieren wurden zusätzlich 4 Dummy-Variablen eingeführt, die gemeinsam mit den Dummy-Variablen der Stunden viertelstündige Intervalle der Beginnzeiten liefern. In Summe werden also 43 *input-units* verwendet.

Die Maxima und Minima der Reichweiten betragen beim ORF 19,3 % und 2,6 %, beim ORF1 23,6 % und 0,1 % und beim ORF2 12 % und 0,1 %. Um die Werte auf ein Intervall von Null bis Eins zu bringen wurden sie vor dem Training mit 0,01 multipliziert und anschließend rückskaliert. Die höheren Reichweiten

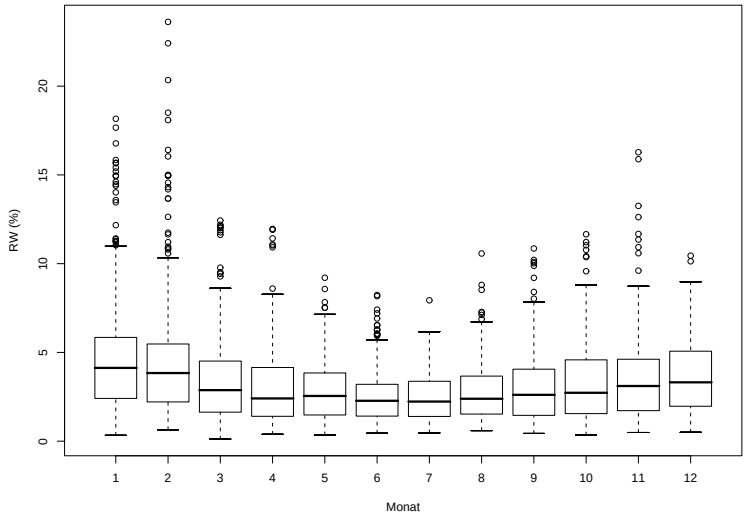


Abbildung 18: Reichweiten von ORF1 abhängig vom Monat

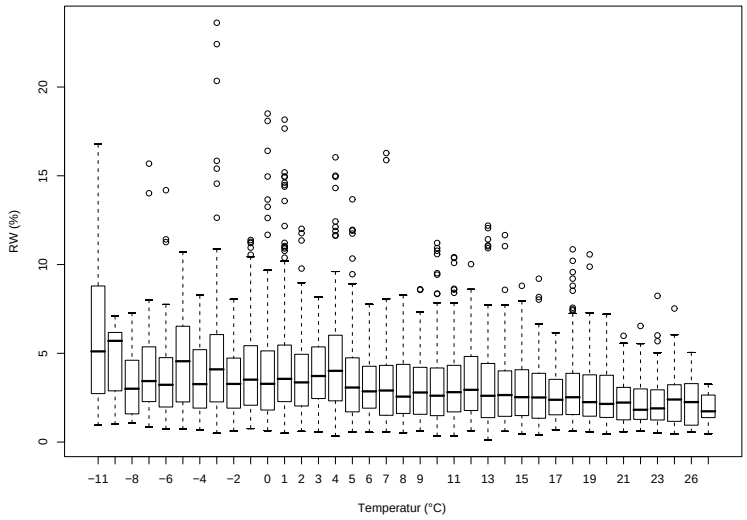


Abbildung 19: Reichweiten von ORF1 abhängig von der Temperatur

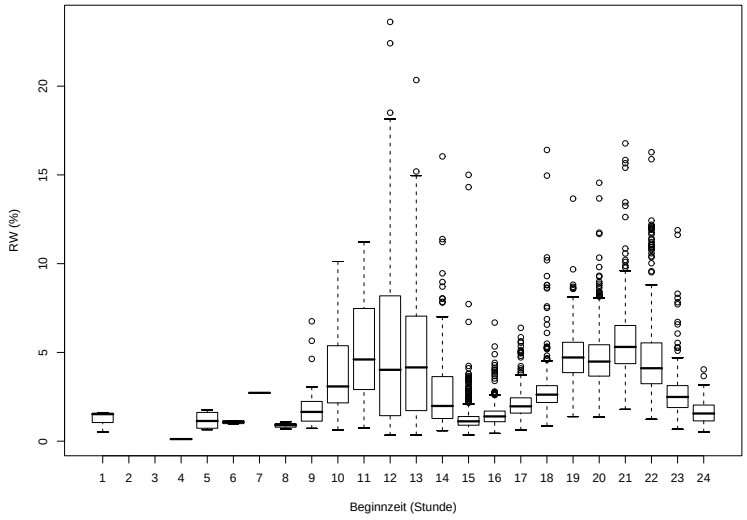


Abbildung 20: Reichweiten von ORF1 abhängig von der Beginnzeit

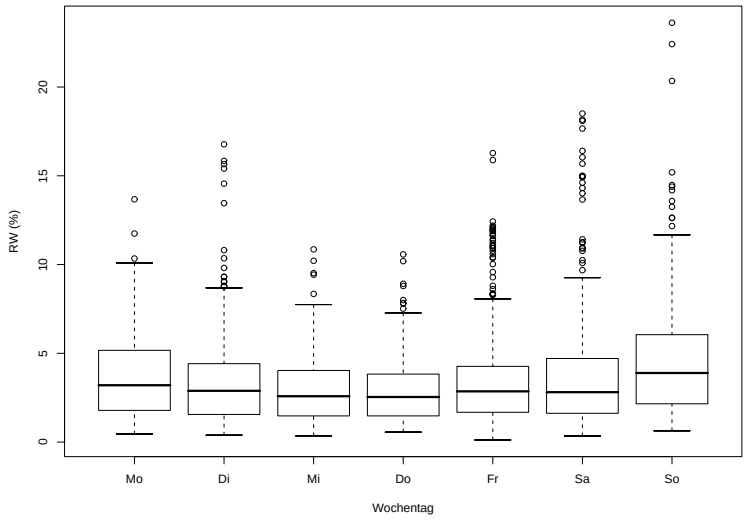


Abbildung 21: Reichweiten von ORF1 abhängig vom Wochentag

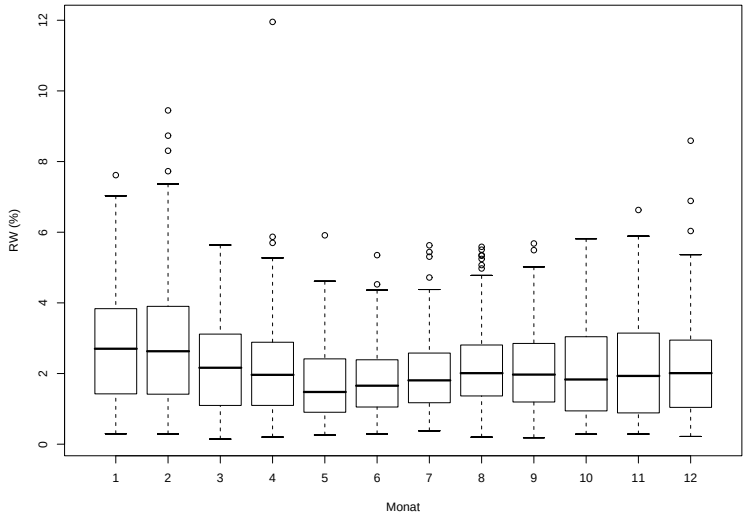


Abbildung 22: Reichweiten vom Sender ORF2 abhängig vom Monat

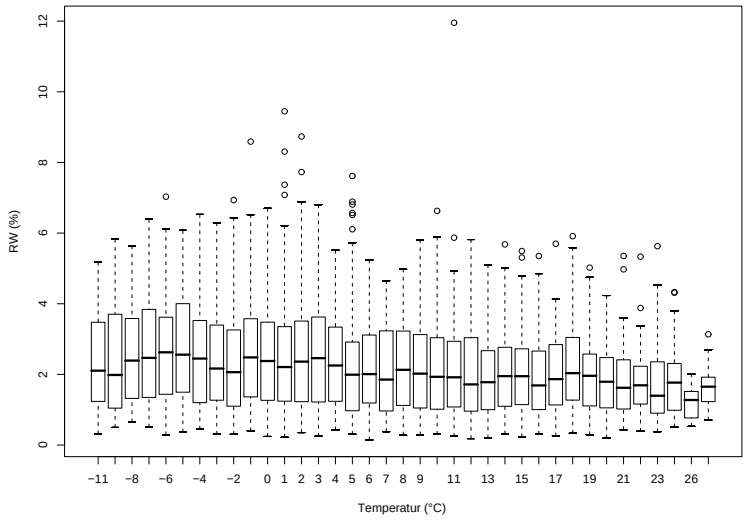


Abbildung 23: Reichweiten vom Sender ORF2 abhängig von der Temperatur

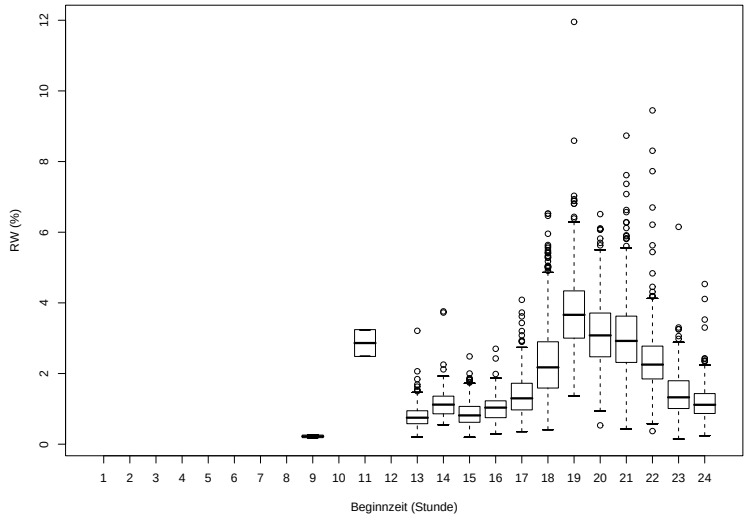


Abbildung 24: Reichweiten vom Sender ORF2 abhängig von der Beginnzeit

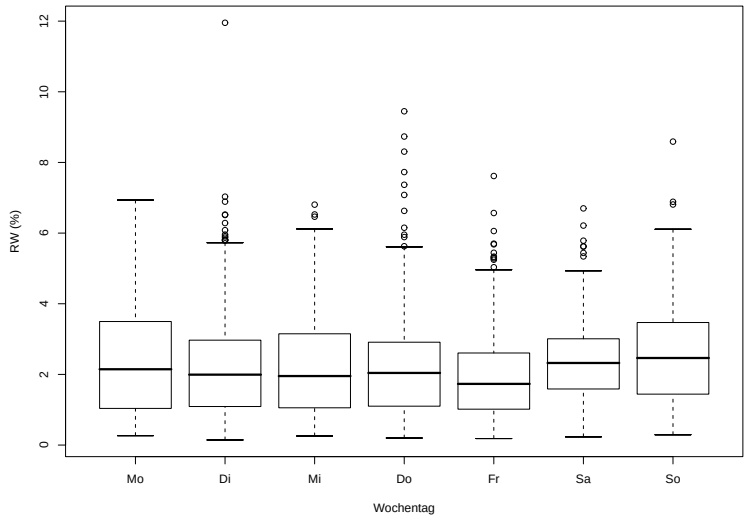


Abbildung 25: Reichweiten vom Sender ORF2 abhängig vom Wochentag

beim ORF ergeben sich verständlicherweise, da hier die Zuseher von zwei Sendern (ORF1 und ORF2) erfasst werden.

In Tabelle 1 sind jene Netze, die hinsichtlich MAPE und R^2 die besten Ergebnisse liefern zusammengefasst. Die Angabe des Netzes ist wie folgt zu verstehen: die erste Ziffer bezeichnet die Anzahl der *input-units*, die zweite die Anzahl der *hidden-units*, und die letzte Ziffer die *output-unit*, die stets Eins ist. "Zyklen" gibt an, wie oft die Trainingsdaten dem Netz vollständig präsentiert wurden.

Aufgrund dieser Information wurde das beste Netz – ein 43-25-1 Netz (geringster MAPE und höchste Korrelation) – ausgewählt und mit dem kompletten Datensatz (12 558 Einträge) 300 Epochen trainiert. Anschließend wurden sowohl *ex-post*, als auch *ex-ante* Prognosen (wobei es sich eigentlich um sogenannte Pseudo *ex-ante* Prognosen handelt (vgl. Kapitel 8)) durchgeführt.

Für die *ex-post* Prognosen wurde der Zeitraum Februar 2006 gewählt – dabei wurden beim Sender ORF1 und ORF2 die Ergebnisse in der Zeit zwischen 20:00 Uhr und 21:00 Uhr genauer untersucht. Beim Sender ORF beziehen sich die Prognosen auf die Werbespots vor dem Wetterbericht um ca. 19:50 Uhr. Wie wir bereits gesehen haben ist zu dieser Zeit im Allgemeinen eine hohe Reichweite zu verzeichnen, weshalb die Prognose von Werbespots hier besonders interessant ist. Diese Daten wurden dem Netz während des Trainings bereits präsentiert. In der Abbildung 26 sieht man die Ergebnisse der Prognose für den ORF. Die blaue Linie gibt die tatsächlichen Reichweiten an, die rote Linie, die vom Netz generierten *outputs*. Der Grund warum in einigen Abbildungen ein paar Tage des Monats ausgelassen wurden ist jener, dass an manchen Tagen keine tatsächlichen Messungen zur Verfügung standen. Prognosen für diese Tage wurden, da es keine Vergleichswerte gibt, bei der Erstellung der Abbildungen entfernt.

Abbildung 27 zeigt die Prognosen für den Februar im ORF1. Auch hier gibt die blaue Linie die tatsächlichen Messungen und die rote Linie die prognostizierten Reichweiten an. Man sieht, dass das Netz nicht in der Lage ist Ausreißer, wie beispielsweise am 14.2, annähernd richtig vorherzusagen. Ein Grund für die Ausreißer kann in der Übertragung der Olympischen Winterspiele auf ORF1 liegen. Am 14.2 fand die Ski Alpin Kombination der Herren statt.

In Abbildung 28 sieht man die Reichweiten für den Februar im ORF2. Die schwarze und die blaue Linie geben die tatsächlichen Messungen an, wobei die schwarze Linie die Reichweiten am Hauptabend, um ca. 20:11 Uhr und die blaue Linie die Reichweiten zur Generalansage um etwa 20:05 Uhr darstellen.

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
43-25-1	300	27,54	90,26	81,47
43-26-1	300	27,89	89,66	80,39
43-28-1	400	28,79	89,64	80,35

Tabelle 1: Gütekriterien für die Sender ORF1, ORF2 und ORF bei verschiedenen Netzwerkkonfigurationen

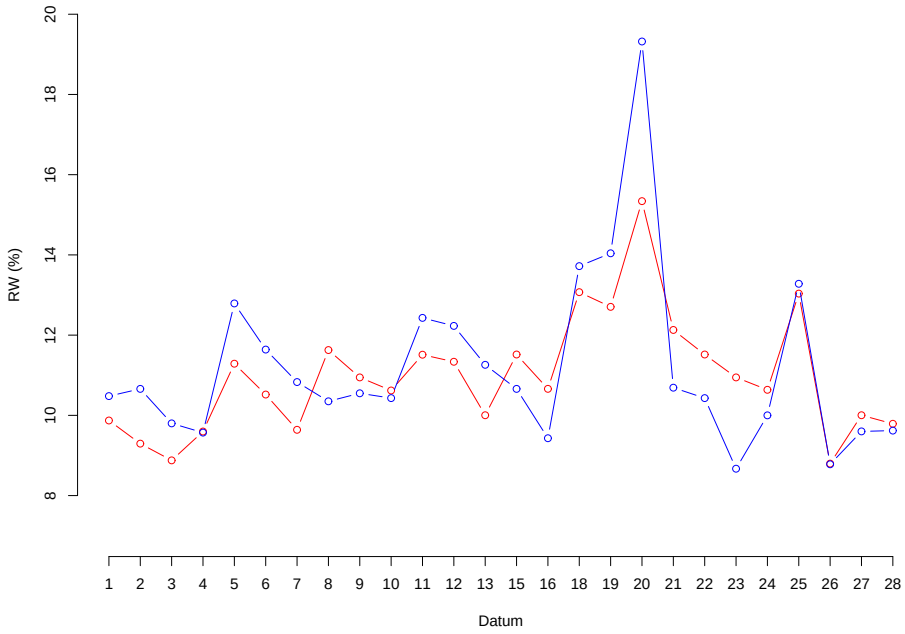


Abbildung 26: Prognose für den Sender ORF für den Februar 2006

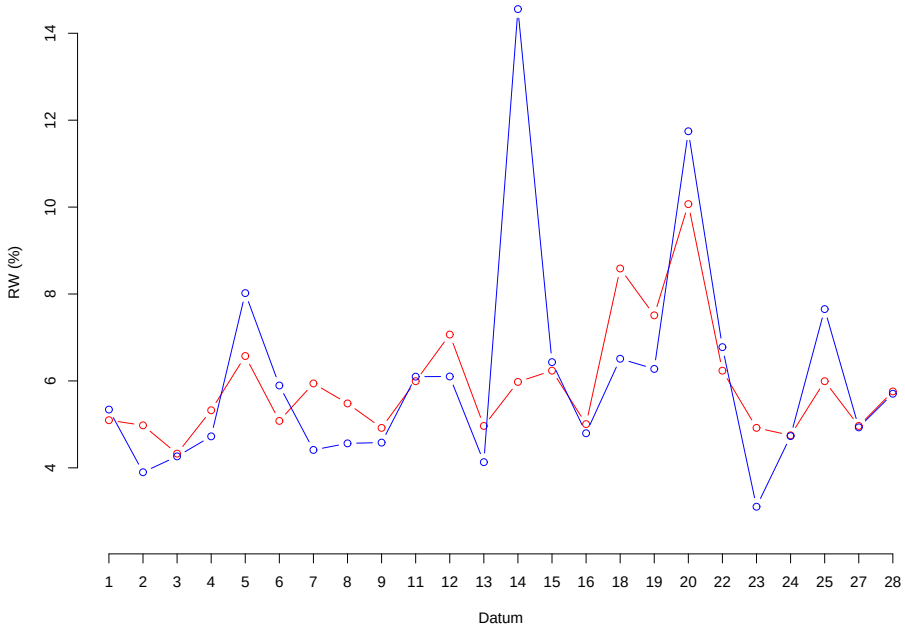


Abbildung 27: Prognose für den Sender ORF1 für den Februar 2006

Da die Ausstrahlungszeiten der beiden Werbeblöcke in dasselbe Zeitintervall (20:00 Uhr bis 20:15 Uhr) fallen, erhält man nur eine Prognose. Man sieht jedoch, dass die tatsächlichen Reichweiten von Generalansage und Hauptabend annähernd gleich sind, weswegen eine Prognose für beide Werbeblöcke vertretbar ist. Auch hier sieht man wieder, dass Ausreißer – in diesem Fall handelt es sich um Ausreißer nach unten – vom Netz nicht erfasst werden können.

Neben den Prognosen für den Februar, wurden auch Prognosen in die Zukunft unternommen – im Detail wurden die Monate Mai und Juli des Jahres 2006 untersucht. Die Daten für diese beiden Monate wurden dem Netz beim Training nicht präsentiert. Da zum Zeitpunkt der Prognose die zukünftigen Temperaturen nicht bekannt waren, wurden die Mitteltemperaturen der letzten Jahre für den jeweiligen Tag angenommen.

Die Abbildungen 29, 30 und 31 zeigen die Prognosen für den Monat Mai. Die schwarzen und blauen Linien geben auch hier die Reichweiten am Hauptabend und zur Generalansage an. Der ORF bildet hier eine Ausnahme – die schwarze und die blaue Linie geben die tatsächlichen Reichweiten des Werbeblocks vor der ZIB um ca. 19:25 Uhr und vor dem Wetterbericht um etwa 19:50 Uhr an. Die rote Linie und die grüne Linie geben die jeweiligen Prognosen an – rot die Prognose für die Werbung vor der ZIB und grün die Prognose für die Werbung vor dem Wetterbericht.

Während die Reichweiten von der Generalansage und dem Hauptabend auf ORF1 und ORF2 nur geringe Unterschiede aufweisen, sind beim Sender ORF deutliche Unterschiede zwischen den Reichweiten vor der ZIB und dem Wetter zu sehen. Die Reichweiten vor dem Wetter sind meist höher – man sieht anhand der unterschiedlichen Prognosen für die beiden Zeiträume, dass dieser Unterschied auch vom Netz erfasst wurde.

Die Abbildungen 32, 33 und 34 zeigen die Prognosen für den Juli. Man sieht, dass die Prognosen bei allen drei Sendern über den tatsächlich eingetretenen Reichweiten liegen. Ein Grund dafür ist in den hohen Temperaturen im Juli zu finden. Bei hohen Temperaturen nimmt der Anteil der Zuseher gewöhnlich ab, da anderen Freizeitaktivitäten nachgegangen wird – da das Netz diesen Zusammenhang gelernt hat und von niedrigeren Temperaturen ausgegangen wurde kommen durchgehend zu hohe Reichweiten zustande.

Um diesen Effekt auch zu veranschaulichen wurde der Sender ORF2 noch einmal prognostiziert – diesmal wurden jedoch die Mitteltemperatur an jedem Tag um 5 Grad Celsius erhöht. Abbildung 35 zeigt die Ergebnisse für dieses *case scenario*. Man erkennt, dass die prognostizierten Reichweiten niedriger sind und jetzt mit den tatsächlichen Werte im selben Reichweitenintervall liegen.

Der Versuch, die Temperatur nicht als Inputprädikator einzubeziehen und mit einem 42-25-1 Netz zu arbeiten bringt in diesem Fall zwar eine leichte Verbesserung, die Reichweiten werden im Allgemeinen jedoch noch immer überschätzt (siehe dazu Abbildung 36). Der Grund dafür liegt in den höheren Reichweiten im Juli 2005 die beim Training gelernt werden. Um diesen Fehler auszugleichen müssten weitere Faktoren als Inputdaten berücksichtigt werden. Einige Ideen dazu werden noch in Kapitel 15.3 angesprochen.

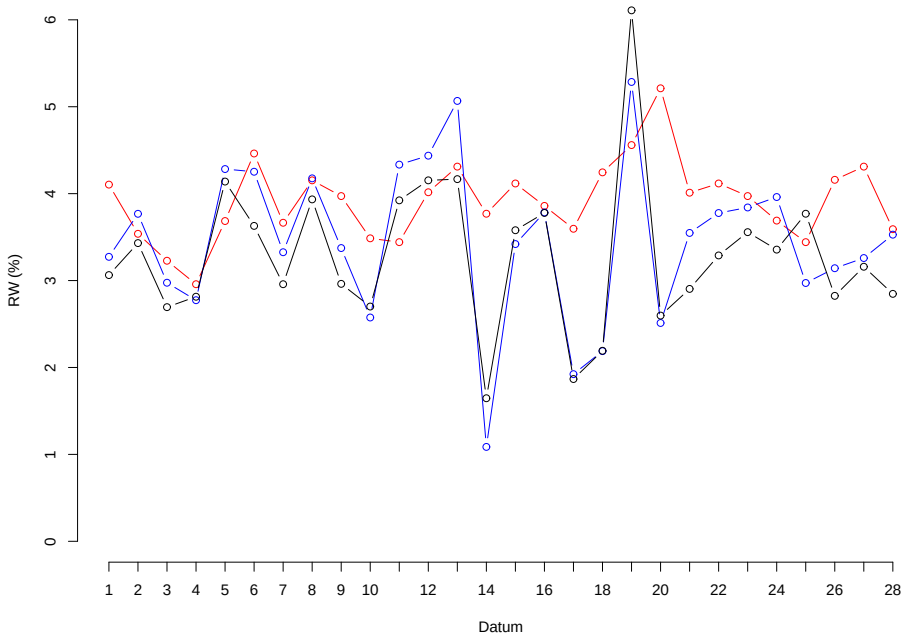


Abbildung 28: Prognose für den Sender ORF2 für den Februar 2006

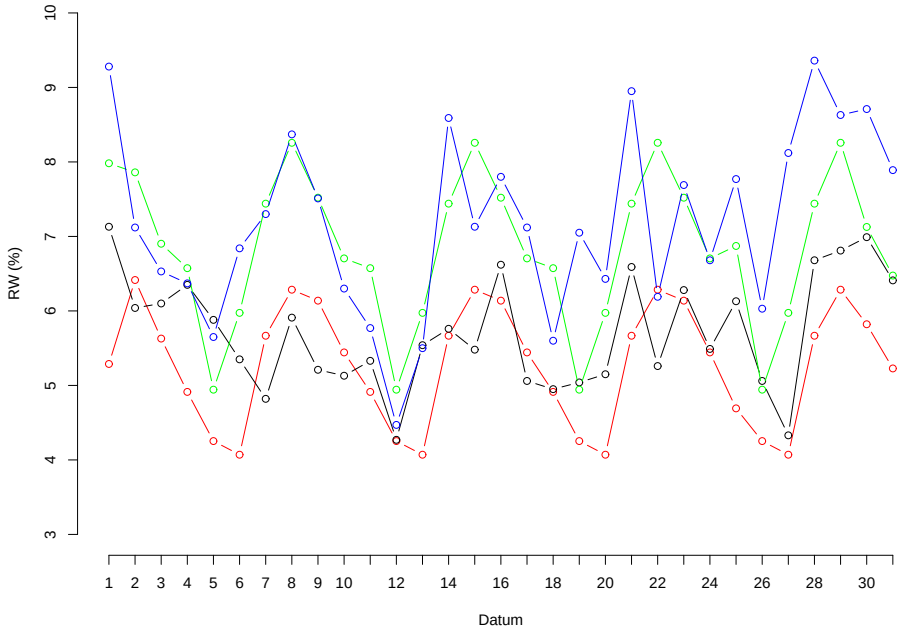


Abbildung 29: Prognose für den Sender ORF für den Mai 2006

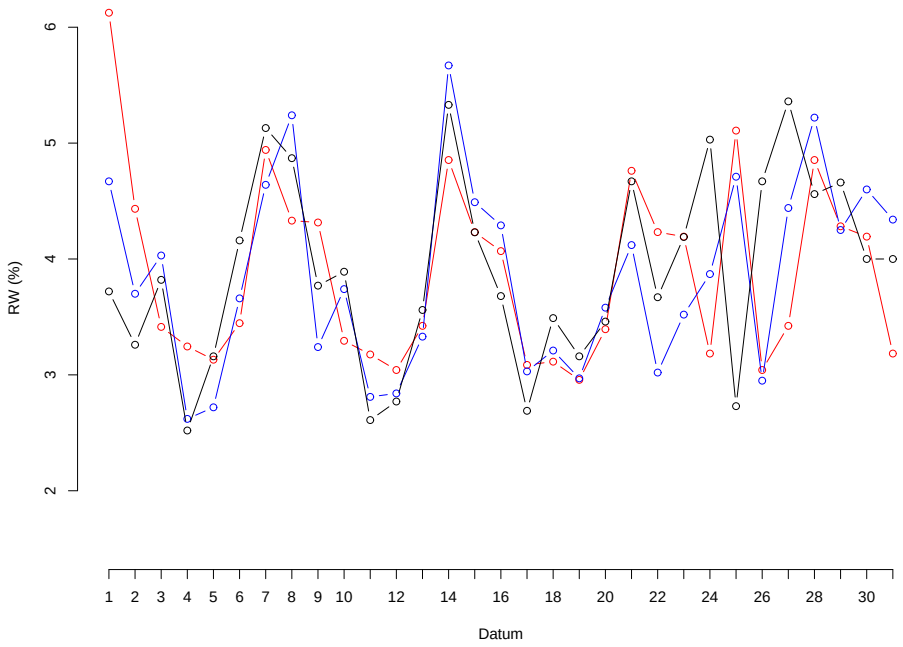


Abbildung 30: Prognose für den Sender ORF1 für den Mai 2006

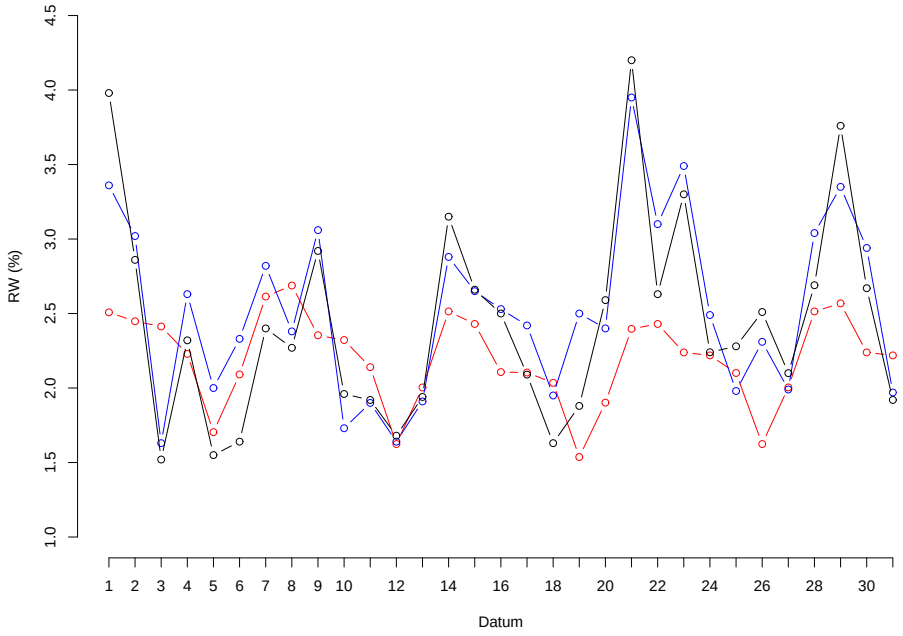


Abbildung 31: Prognose für den Sender ORF2 für den Mai 2006

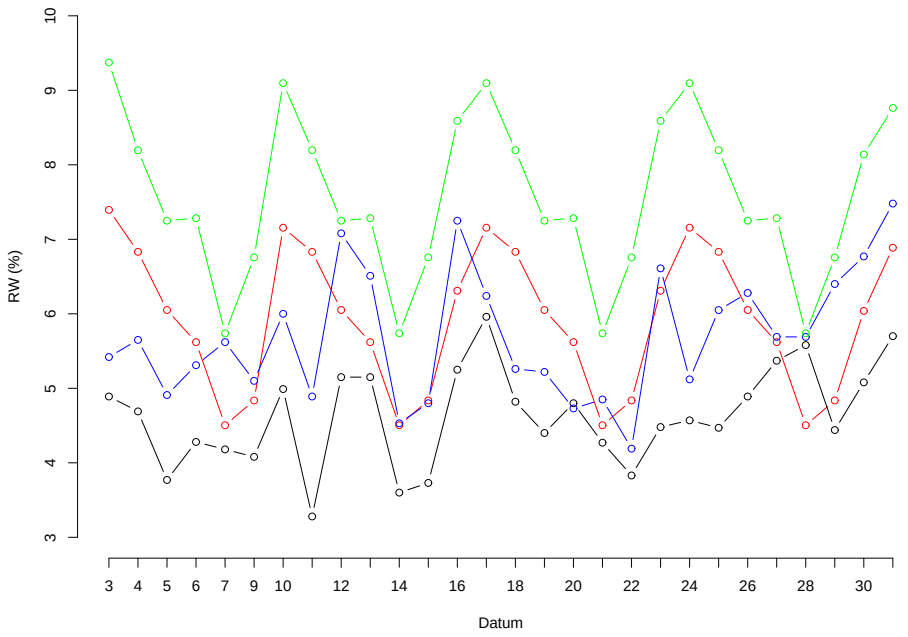


Abbildung 32: Prognose für den Sender ORF für den Juli 2006

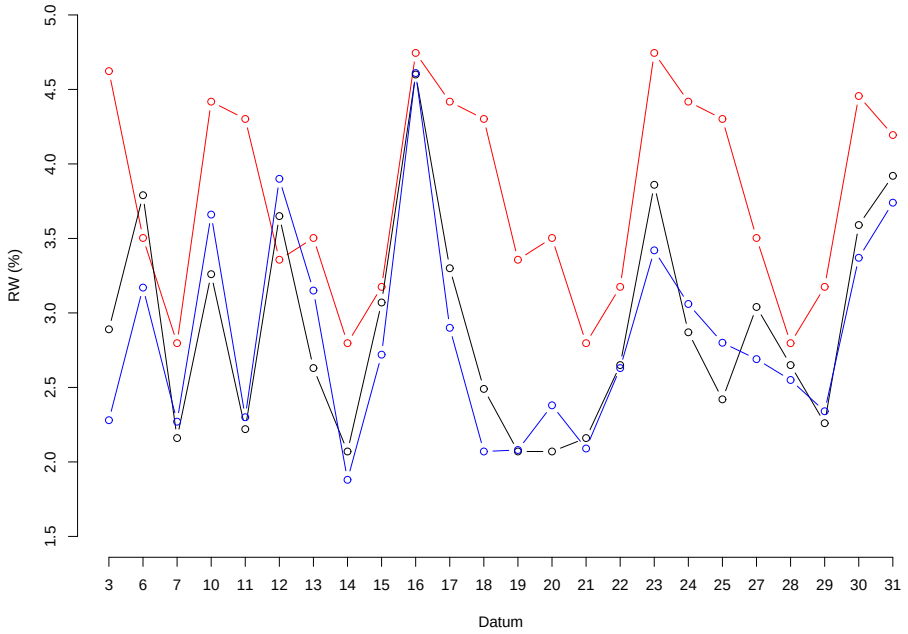


Abbildung 33: Prognose für den Sender ORF1 für den Juli 2006

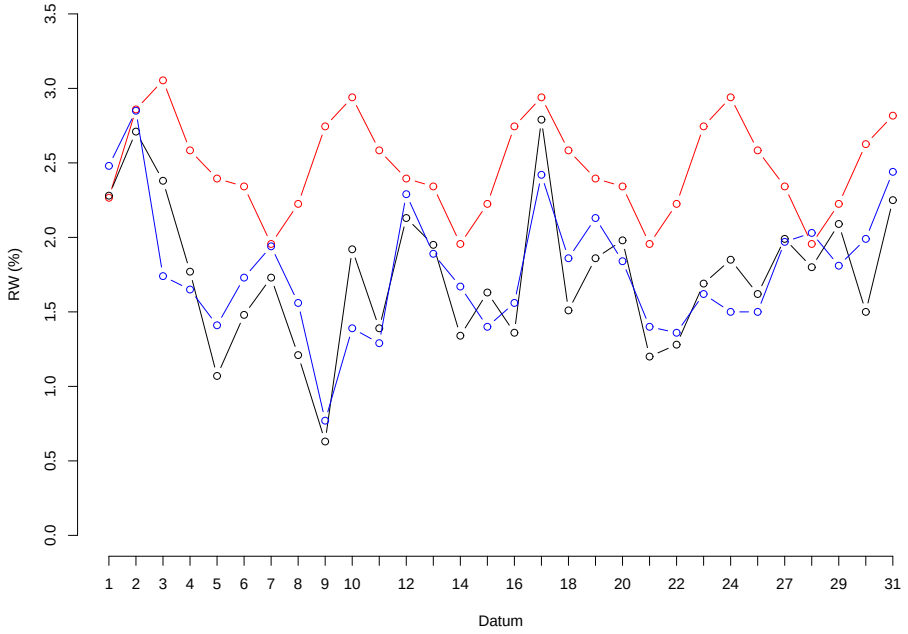


Abbildung 34: Prognose für den Sender ORF2 für den Juli 2006

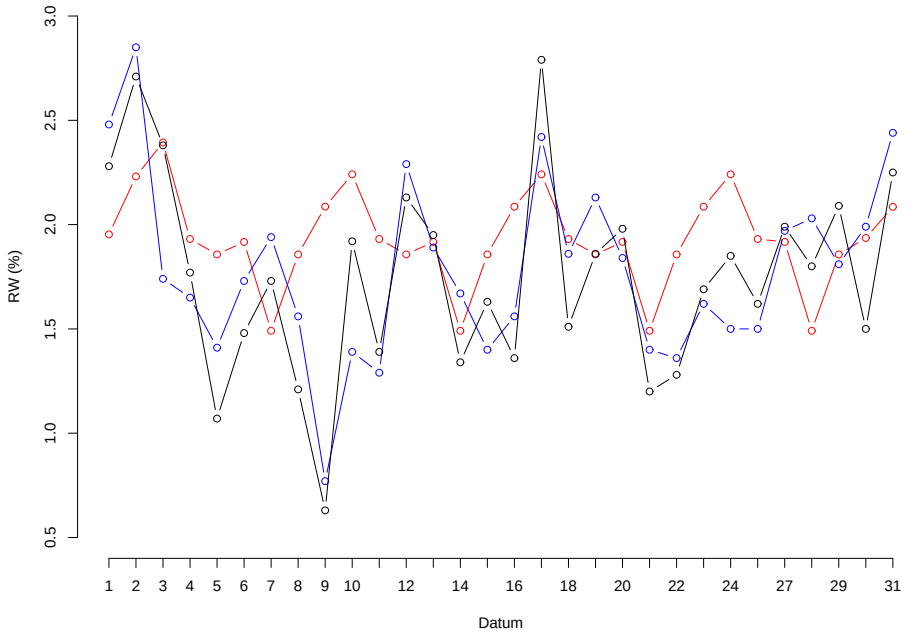


Abbildung 35: Prognose für den Sender ORF2 für den Juli 2006 bei Erhöhung der Mitteltemperatur um 5 Grad Celsius

Abbildung 37 zeigt die Prognose für das Jahr 2006. Dabei gibt die rote Linie die Prognosen für den ORF1, die blaue Linie die Prognosen für den ORF2 und die grüne Linie die Prognosen für den ORF an. Die Abbildung zeigt die Reichweiten jeweils für den Sonntag zur Generalansage um 20:05 Uhr bzw. beim ORF für den Werbeblock vor der ZIB um 19:26 Uhr. Es wurde hier die Zeit und der Tag absichtlich konstant gehalten, um den saisonalen Trend über die Monate ersichtlich zu machen. Ebenfalls wurden für das gesamte Jahr die Mitteltemperaturen verwendet um größere Abweichungen zwischen den *ex-post* und *ex-ante* Prognosen zu verhindern. Ab dem Monat September 2006 handelt es sich hier um tatsächliche *ex-ante* Prognosen, da zum jetzigen Zeitpunkt (Ende August 2006) für das letzte Quartal dieses Jahres klarerweise noch keine Informationen bezüglich der Reichweite vorhanden sind. Man sieht bei allen drei Sendern eine Abnahme der Reichweite im Sommer – die minimalsten Reichweiten findet man im August – und einen Anstieg ab Herbst der seine Höhepunkt im Februar erreicht. Dieser Trend ist beim ORF2 am schwächsten ausgeprägt. Die stärkste Ausprägung dieses Trends findet man beim ORF.

Bei den Gütemaßen aus Tabelle 1 handelt es sich um die Ergebnisse einer *in-sample* Prognose. Das heißt die Daten wurden aus derselben Grundgesamtheit, wie die Daten der Trainingsmenge entnommen, nämlich aus dem Zeitraum vom 1.1.05 bis 31.4.06. Da *in-sample* Prognosen in der Regel bessere Ergebnisse liefern als *out-of-sample* Prognosen, sollen an dieser Stelle noch die erreichten Gütemaße einer *out-of-sample* Prognose dargestellt werden. Von einer *out-of-sample* Prognose spricht man, wenn die Daten für die Prognose aus einer anderen Population wie jene für das Training stammen. Man wählt dabei im Normalfall eine direkt an den Trainingszeitraum angrenzende Zeitspanne. In diesem Fall wurde der Zeitraum vom 1.5.06 bis 31.7.06 gewählt. Es stehen 498 Datensätze zur Verfügung, die die Reichweiten der Werbeblöcke der Generalansage, des Hauptabends, der ZIB und des Wetters enthalten. Als Gütemaße ergeben sich ein MAPE von 27,08 %, ein R von 86,16 % und ein R^2 von 74,23 %.

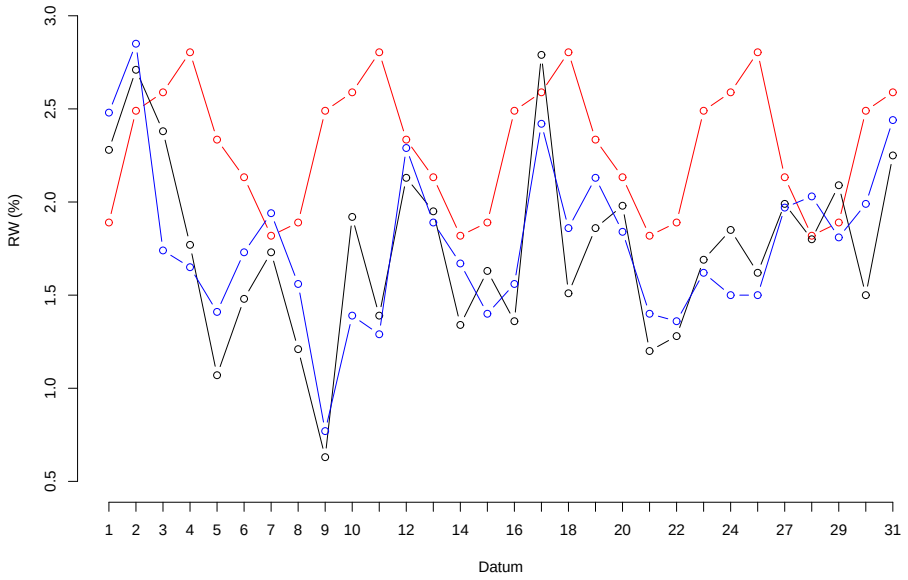


Abbildung 36: Prognose für den Sender ORF2 für den Juli 2006 ohne Einbeziehung der Temperatur

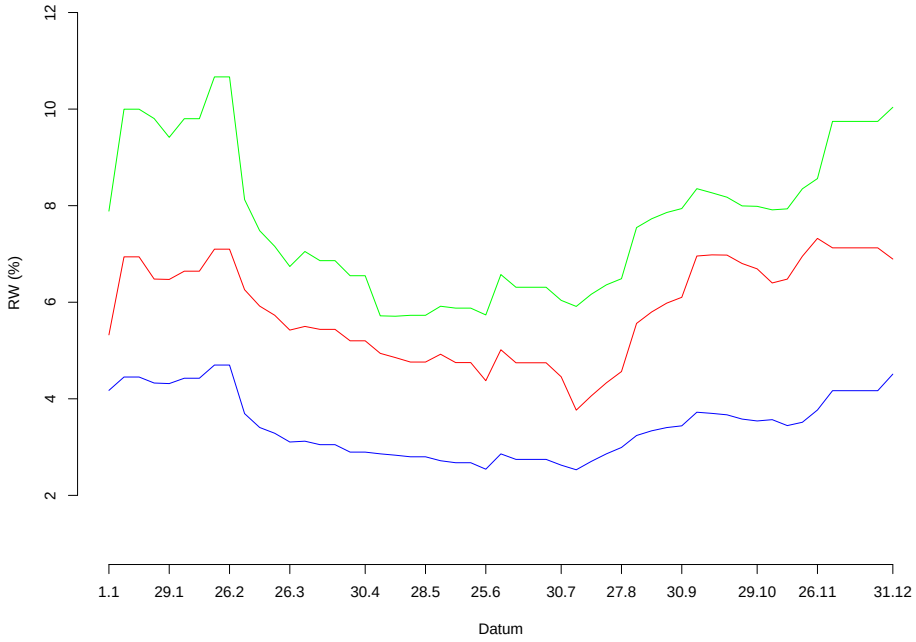


Abbildung 37: Prognose für das Jahr 2006

15.2.2 ATV

Der Datensatz besteht aus 19 675 Einträgen, wobei 5 053 aus dem Jahr 2006 stammen. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 5,02 %. 1 055 Daten haben eine Reichweite größer 1 % – davon haben wiederum 83 eine Reichweite größer 2 %, 5 eine Reichweite größer 3 %, 3 eine Reichweite größer 4 % und einer eine Reichweite größer 5 %. Die 5 Spitzenreichweiten stammen vom 12.9.05 in der Zeit zwischen 20:00 Uhr und 23:00 Uhr, als die Losung der Gruppen für die Fussball-WM 2006 ausgestrahlt wurde.

Abbildung 38 zeigt die Verteilung der Reichweiten abhängig von der Beginnzeit.

Zur Identifizierung eines Netzes wurden die Daten des Jahres 2005 verwendet. Es wurden nur Daten verwendet, die eine Reichweite größer 0,03 % aufweisen (kleinere Reichweiten sind von keinem besonderen Interesse). Jene Einträge, deren Beginnzeit zwischen 2:00 Uhr und 16:00 Uhr lagen wurden dabei entfernt, da hier im Schnitt über 30 % der Daten eine Reichweite kleiner 0,03 % aufweisen. Somit ergeben sich 10 Dummy-Variablen für die Beginnzeiten und in Summe 31 *input-units*. Es stehen nun 8 053 Daten zur Verfügung (Training: 5 637; Validierung: 1 610; Prognose: 806). Die Outputwerte wurden vor Beginn des Trainings mit 0,1 multipliziert.

Welche Netze die besten Ergebnisse liefern ist in Tabelle 2 zu sehen.

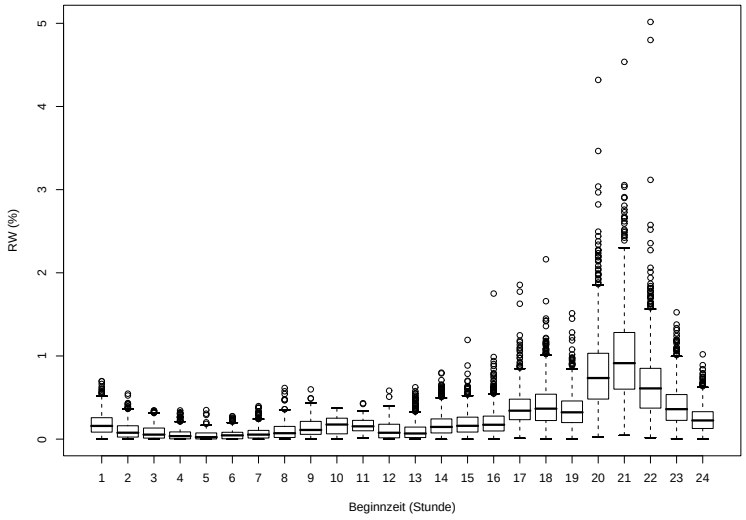


Abbildung 38: Reichweiten vom Sender ATV abhängig von der Beginnzeit

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
31-20-1	600	65,42	76,77	58,93
31-21-1	450	64,93	75,50	57,00
31-22-1	300	68,04	75,22	56,57

Tabelle 2: Gütkriterien für den Sender ATV bei verschiedenen Netzwerkkonfigurationen

15.2.3 RTL

Der Datensatz des Senders RTL besteht aus 27 780 Einträgen, wobei 7 230 aus dem Jahr 2006 stammen. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 3,12 %. In Summe haben 1 217 Daten eine Reichweite größer 1 % (davon haben wiederum 62 Daten eine Reichweite größer 2 % und 2 Daten eine Reichweite größer 3 %).

Die Verteilung der Reichweiten abhängig von der Beginnzeit des Werbespots zeigt Abbildung 39.

Zur Bestimmung der besten Netze wurden zwei Versuche unternommen. Einerseits der Versuch Reichweiten größer 0,1 % und andererseits Reichweiten größer 0,03 % zu prognostizieren. In beiden Fällen wurden die Reichweiten vor dem Training mit 0,1 multipliziert werden.

Im ersten Fall wurde der Datenbestand um die Beginnzeiten der Spots zwischen 0:00 Uhr und 12:00 Uhr gekürzt – damit ergibt sich ein Datensatz mit 11 260 Einträgen (Training: 7 881; Validierung: 2 252; Prognose: 1 127). Der Grund für die Reduzierung liegt darin, dass hier mehr als 50 % der gemessenen Reichweiten unter 0,1 % liegen. Es ergeben sich 12 Dummy-Variablen für die Beginnzeiten und in Summe 33 *input-units*. Tabelle 3 zeigt die besten Netze.

Im zweiten Versuch wurde der Datenbestand um die Beginnzeiten zwischen 1:00 Uhr und 9:00 Uhr verringert. Hier liegt der Anteil der Daten mit Reichweiten unter 0,03 % jeweils bei ca. 15 %. Daher ergeben sich 16 Dummy-Variablen für die Beginnzeiten und in Summe 37 *input-units*. Es stehen insgesamt 15 675 Daten zur Verfügung (Training: 10 972; Validierung: 3 135; Prognose: 1 568). Die Ergebnisse der besten Netze sind in Tabelle 4 zusammengefasst.

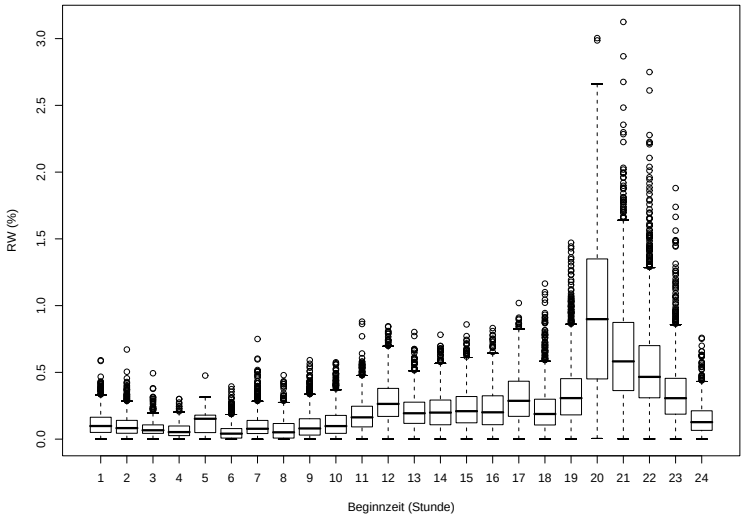


Abbildung 39: Reichweiten vom Sender RTL abhängig von der Beginnzeit

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
33-20-1	300	47,48	72,83	53,05
33-25-1	100	48,87	72,75	52,93
33-18-1	200	48,71	71,93	51,74

Tabelle 3: Gütekriterien für den Sender RTL bei verschiedenen Netzwerkkonfigurationen

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
37-25-1	200	72,88	73,10	53,44
37-26-1	200	73,60	72,68	52,82
37-20-1	100	74,98	72,41	52,43

Tabelle 4: Gütekritierien für den Sender RTL bei verschiedenen Netzwerkkonfigurationen

15.2.4 SAT1

Beim Sender SAT1 stehen 19 408 Daten zur Verfügung, davon stammen 4 363 aus dem Jahr 2006. Die minimale Reichweite beträgt weniger als 0,001 %, die maximale 2,30 %. Insgesamt haben 386 Daten eine Reichweite größer 1 % (davon wiederum 10 eine Reichweite größer 2 %).

Abbildung 40 zeigt die Reichweiten abhängig von der Beginnzeit des Werbespots.

Zur Identifizierung der geeignetsten Netze wurde die Daten des Jahres 2005 herangezogen. Die Netze wurden nur für Reichweiten trainiert, die über 0,03 % liegen. Aus diesem Grund wurden jene Messungen, die eine Beginnzeit zwischen 0:00 Uhr und 12:00 Uhr haben gestrichen – zu diesen Zeiten haben in etwa 50 % der zur Verfügung stehenden Daten eine Reichweite kleiner 0,03 % und könnten von den trainierten Netzen somit nicht korrekt erfasst werden. Es ergeben sich 12 Dummy-Variablen für die Beginnzeiten und in Summe 33 *input-units*. Es stehen insgesamt 9 780 Daten zur Verfügung (Training: 6 846; Validierung: 1 956; Prognose: 978). Die Ergebnisse sind in Tabelle 5 zu finden.

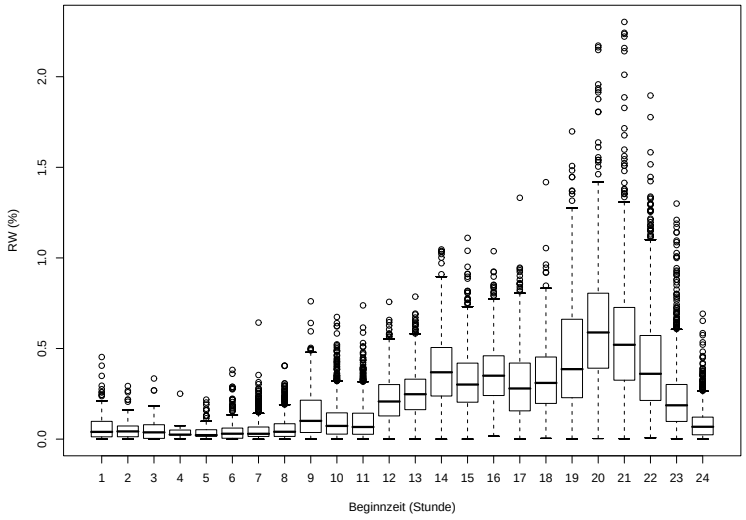


Abbildung 40: Reichweiten vom Sender SAT1 abhängig von der Beginnzeit

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
33-19-1	300	76,25	68,28	46,61
33-18-1	300	74,46	67,42	45,46
33-17-1	300	75,97	67,38	45,40

Tabelle 5: Gütekritierien für den Sender SAT1 bei verschiedenen Netzwerkkonfigurationen

15.2.5 PRO7

Der Datensatz des Senders PRO7 besteht aus 21 611 Daten, wobei 4 792 aus dem Jahr 2006 stammen. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 3,08 %. 689 Messungen haben eine Reichweite größer 1 % ergeben, wovon 18 eine Reichweite größer 2 % aufweisen und einzig das Maximum eine Reichweite größer 3 %.

Abbildung 41 zeigt die Reichweiten in Abhängigkeit von den Beginnzeiten der Werbespots.

Es wurden zwei Prognoseversuche unternommen: einerseits eine Prognose von Reichweiten, größer 0,03 % und andererseits größer 0,1 %. In beiden Fällen mussten einige Daten aus dem vorhandenen Datenmaterial gestrichen werden. Es handelt sich dabei um jene Messungen, im Zeitraum zwischen 1:00 Uhr und 10:00 Uhr. Zu dieser Zeit haben mehr als 30 % der vorhandenen Daten Reichweiten kleiner 0,03 % bzw. mehr als 60 %, Reichweiten kleiner 0,1 %. Somit entstehen 15 Dummy-Variablen für die Beginnzeiten und 36 *input-units*.

Es sind sind in Summe 12 273 Messungen mit Reichweiten größer 0,03 % (Training: 8 591; Validierung: 2 454; Prognose: 1 228) und 10 973 Messungen mit Reichweite größer 0,1 % (Training: 7 681; Validierung: 2 194; Prognose: 1 098) vorhanden.

Die besten Netze für die Prognose der Reichweiten größer 0,03 % und 0,1 % sind in den Tabellen 6 bzw. 7 zu sehen. Man erkennt, dass die Netze bessere Werte hinsichtlich MAPE und Korrelation liefern, die nur mit Reichweiten größer 0,1 % trainiert wurden.

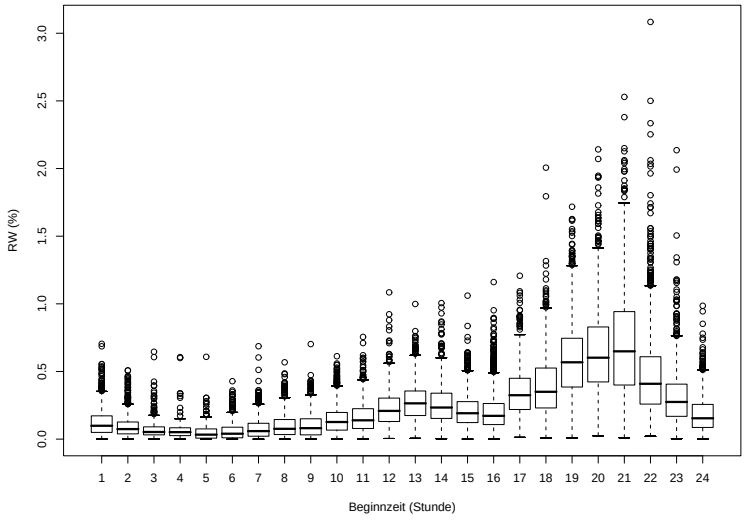


Abbildung 41: Reichweiten vom Sender PRO7 abhängig von der Beginnzeit

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
36-18-1	400	57,85	74,21	55,07
36-20-1	400	59,90	73,28	53,70
36-19-1	400	60,37	72,59	53,22

Tabelle 6: Gütekriterien für den Sender PRO7 bei verschiedenen Netzwerkkonfigurationen

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
36-17-1	700	42,82	77,57	60,18
36-18-1	800	43,46	77,22	59,63
36-19-1	500	43,39	76,70	58,83

Tabelle 7: Gütekriterien für den Sender PRO7 bei verschiedenen Netzwerkkonfigurationen

15.2.6 RTL2

Der Datensatz des Senders RTL2 umfasst 22 424 Daten, wovon 6 098 aus dem Jahr 2006 stammen. Dir minimale Reichweite liegt bei weniger als 0,001 %, die maximale Reichweite ist 1,65 %. Insgesamt haben nur 19 Messungen eine Reichweite größer 1 % ergeben.

Abbildung 42 zeigt die Reichweiten abhängig von den Beginnzeiten des Werbespots.

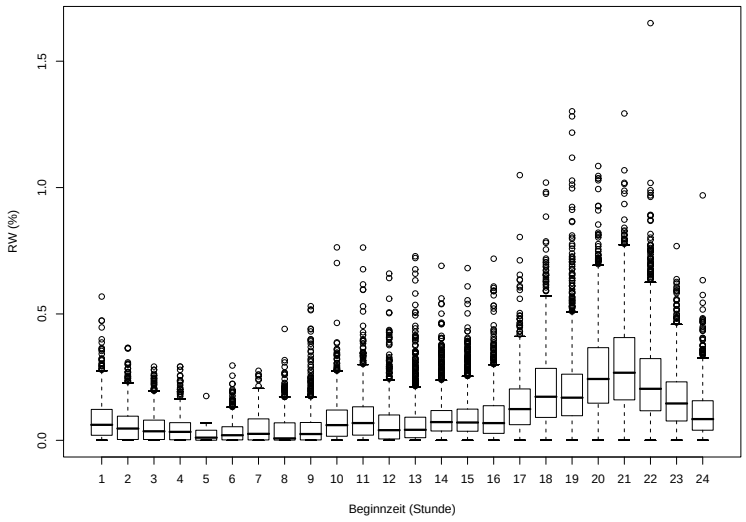


Abbildung 42: Reichweiten vom Sender RTL2 abhängig von der Beginnzeit

Bei der Prognose wurde, wie bereits bei den Sendern ATV, RTL, SAT1 und PRO7, auf Reichweiten kleiner 0,03 % verzichtet. Aus diesem Grund wurden Beginnzeiten eliminiert, bei denen zu 40 % oder häufiger nur sehr kleiner Reichweiten erzielt werden – es handelt sich dabei um die Beginnzeiten von 2:00 Uhr bis 14:00 Uhr. Somit bleiben 12 Dummy-Variablen für die Beginnzeiten und 33 *input-units*. In Summe stehen nun 3 218 Dateneinträge zur Verfügung (Training: 6 102; Validierung: 1 743; Prognose: 873). Die Reichweiten wurden vor dem Training mit 0,1 multipliziert.

Die Netze, die die besten Ergebnisse erzielten sind in Tabelle 8 zu finden.

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
33-22-1	500	63,58	63,56	40,39
33-21-1	300	63,94	62,90	39,56
33-20-1	400	65,02	61,16	37,41

Tabelle 8: Gütekriterien für den Sender RTL2 bei verschiedenen Netzwerkkonfigurationen

15.2.7 VOX

Beim Sender VOX stehen in Summe 22 112 Daten zur Verfügung, davon stammen 5 553 aus dem Jahr 2006. Die minimale Reichweite liegt unter 0,001 %, die maximale beträgt 2, 06 %. 230 Daten haben eine Reichweite größer 1 % (eine Reichweite größer 2 % weist nur das Maximum auf). Die Verteilung, abhängig vom Beginn des Werbespots zeigt Abbildung 43.

Für die Bestimmung des Netzes für Reichweiten größer 0,1 % wurden nur jene Daten herangezogen, deren Beginnzeiten zwischen 16:00 Uhr und 0:00 Uhr liegen – in Summe sind das 7 032 Daten (Training: 4 922; Validierung: 1 406; Prognose: 704). Durch 8 Dummy-Variablen für die Beginnzeiten ergeben sich in Summe 29 *input-units*. Um die Daten auf ein Intervall von 0-1 zu bringen wurden die Reichweiten vorab mit 0,1 multipliziert. Als beste Netze erwiesen sich die in Tabelle 9 aufgelisteten.

Bei der Identifizierung eines geeigneten Netzes für Reichweiten größer 0,03 % wurden sämtliche Daten mit Beginnzeiten zwischen 12:00 Uhr und 1:00 Uhr herangezogen – hier standen 11 679 Daten zur Verfügung (Training: 8 175; Validierung: 2 335; Prognose: 1 169). Auf eine Transformation der Daten wurde verzichtet, zu Gunsten besserer Prognosen für kleine Werte. Man nimmt also in Kauf, dass Reichweiten größer 1 nur annähernd vorhergesagt werden können. Die besten Netze bei diesem Training sind in Tabelle 10 zu finden.

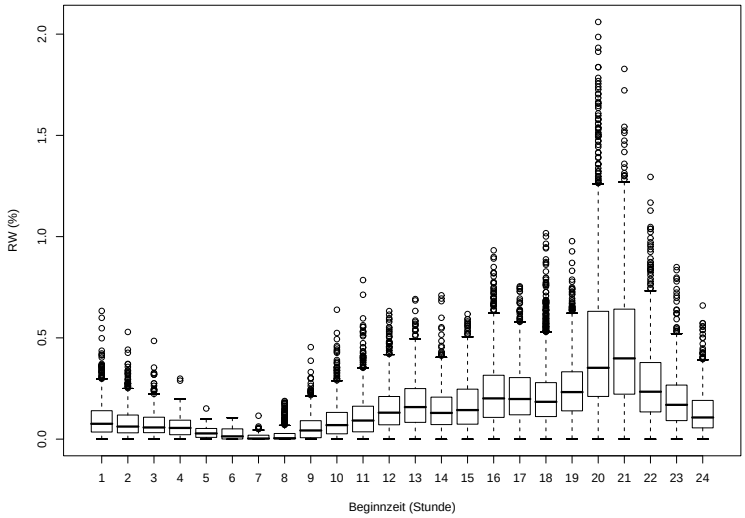


Abbildung 43: Reichweiten vom Sender VOX abhängig von der Beginnzeit

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
29-13-1	700	41,19	84,05	70,64
29-14-1	700	41,49	83,72	70,09
29-12-1	400	43,32	82,19	67,55

Tabelle 9: Gütekriterien für den Sender VOX bei verschiedenen Netzwerkkonfigurationen

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
34-19-1	300	67,07	79,02	62,44
34-16-1	200	67,79	78,98	62,38
34-18-1	200	67,07	78,84	62,15

Tabelle 10: Gütekriterien für den Sender VOX bei verschiedenen Netzwerkkonfigurationen

15.2.8 SRTL

Der Datensatz des Senders SRTL umfasst 20 872 Einträge, davon 4 914 aus dem Jahr 2006. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 1,99 %. 95 Daten weisen eine Reichweite größer 1 % auf.

In Abbildung 44 ist die Verteilung der Reichweiten in Abhängigkeit von der Beginnzeit zu sehen.

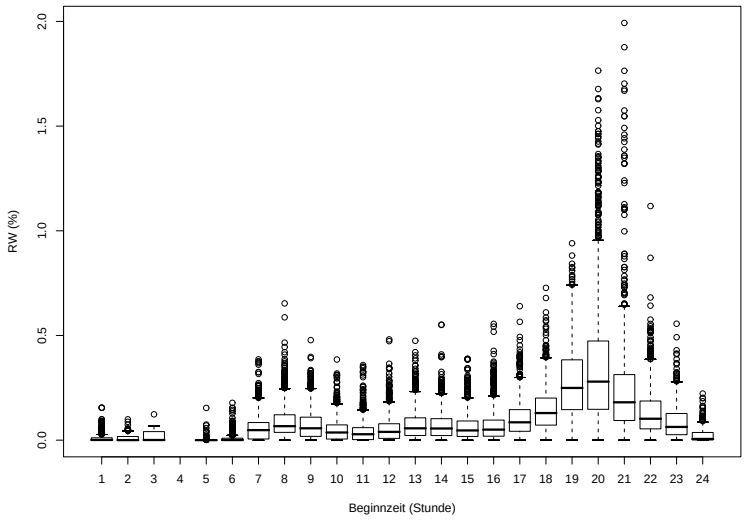


Abbildung 44: Reichweiten vom Sender SRTL abhängig von der Beginnzeit

Für das Training wurden nur Daten mit einer Reichweite größer 0,03 % verwendet. Aus diesem Grund mussten einige Beginnzeiten, bei denen mehr als 30 % aller Messungen eine Reichweite kleiner 0,03 % aufweisen vollständig aus dem Datenbestand eliminiert werden. Die Prognose wurde schließlich nur für Werbespots, die im Zeitraum von 18:00 Uhr bis 24:00 Uhr ausgestrahlt werden vorgenommen. Somit wurden lediglich 6 Dummy-Variablen für die Beginnzeiten benötigt, was in Summe zu 27 *input-units* führt. Der Datensatz bestand nach der Reduzierung aus 6 010 Einträgen (Training: 4 207; Validierung: 1 202; Prognose: 601). Die Reichweiten wurden mit 0,1 multipliziert (die Ergebnisse dazu sind in Tabelle 11 zu sehen).

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
27-15-1	300	86,49	62,85	39,50
27-14-1	600	84,80	62,32	38,84
27-13-1	400	87,86	62,10	38,57

Tabelle 11: Gütekriterien für den Sender SRTL bei verschiedenen Netzwerkkonfigurationen

Überraschenderweise erhält man jedoch bessere Ergebnisse bei der Prognose, wenn man auf diese Skalierung verzichtet und in Kauf nimmt, dass Werte größer 1 % nicht korrekt prognostiziert werden können (Ergebnisse hierzu in Tabelle 12).

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
27-19-1	200	73,66	68,11	46,39
27-15-1	300	75,37	68,05	46,31
27-17-1	500	74,77	67,95	46,17

Tabelle 12: Gütekriterien für den Sender SRTL bei verschiedenen Netzwerkkonfigurationen

15.2.9 KAB1

Der Datensatz des Senders KAB1 umfasst 18 209 Einträge, wobei 4 292 aus dem Jahr 2006 stammen. Die minimale Reichweite liegt unter 0,001 %, die maximale bei 1,29 %. Insgesamt weisen 13 Messungen eine Reichweite größer 1 % auf.

Abbildung 45 zeigt die Verteilung der Reichweiten abhängig von der Beginnzeit der Werbespots.

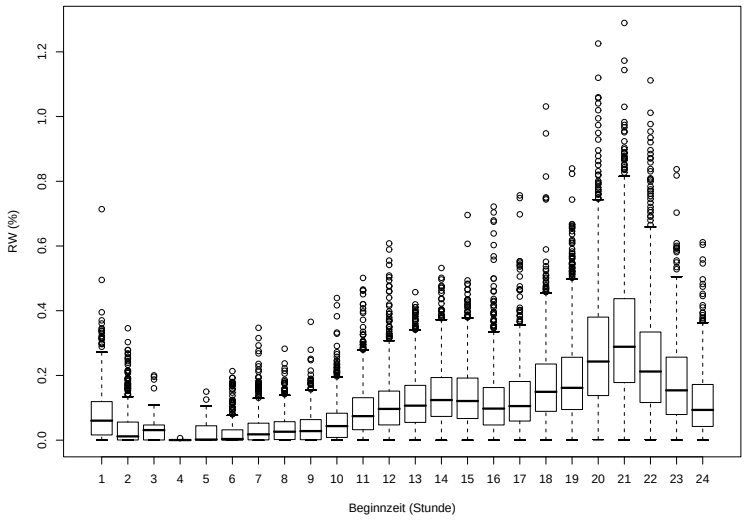


Abbildung 45: Reichweiten vom Sender KAB1 abhängig von der Beginnzeit

Es wurden auch hier nur Reichweiten größer 0,03 % berücksichtigt. Aus diesem Grund musste die Prognose auf die Beginnzeiten der Spots von 11:00 Uhr bis 1:00 Uhr beschränkt werden. Im Zeitraum von 1:00 Uhr bis 11:00 Uhr haben in etwa 30−60 % der Messungen geringere Reichweiten. Teilweise (beispielsweise zwischen 4:00 Uhr und 5:00 Uhr) sind außerdem zu wenige Daten für ein ausreichendes Training vorhanden. Es ergeben sich also 14 Dummy-Variablen für die Beginnzeiten und insgesamt 35 *input-units*. In Summe stehen nun 9 815 Daten zur Verfügung (Training: 6 871; Validierung: 1 963; Prognose: 982).

Die besten Netze sind in Tabelle 13 zu sehen.

Netz	Zyklen	MAPE (%)	R (%)	R^2 (%)
35-18-1	800	64,29	63,48	40,29
35-17-1	600	66,21	62,20	38,69
35-19-1	600	67,36	62,09	38,56

Tabelle 13: Gütekriterien für den Sender KAB1 bei verschiedenen Netzwerkkonfigurationen

15.2.10 GoTV

Es stehen beim Sender GoTV 16 178 Daten zur Verfügung. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 0,401 %. 15 275 Messungen ergaben eine Reichweite kleiner 0,1 % – das entspricht 94,42 % der vorhandenen Daten. Es wurden einige Versuche geeignete Netze zu finden unternommen – die Ergebnisse bezüglich MAPE und R^2 waren jedoch nicht ausreichend um vernünftig prognostizieren zu können. Da die Prognose von Reichweiten in einem Bereich kleiner 0,1 % (das entspricht in absoluten Zahlen ca. 4 250 Zusehern) ohnehin nicht von großem Interesse ist, wurde der Sender GoTV nicht weiter untersucht.

Abbildung 46, 47 und 48 zeigen die Reichweiten abhängig von der Beginnzeit, dem Monat und dem Wochentag. Man erkennt, dass die Reichweiten im Allgemeinen nur sehr geringen Schwankungen im zeitlichen Verlauf unterliegen, die Streuung der Daten jedoch groß ist. Das könnte ein möglicher Grund für das schlechte Abschneiden bei der Prognose sein. Bei den Sendern MTV und PulsTV sind ähnliche Verteilungen zu beobachten.

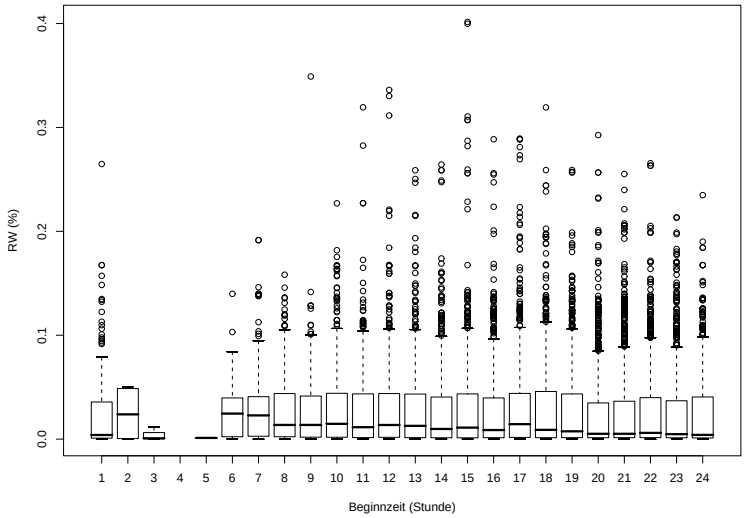


Abbildung 46: Reichweiten vom Sender GoTV abhängig von der Beginnzeit

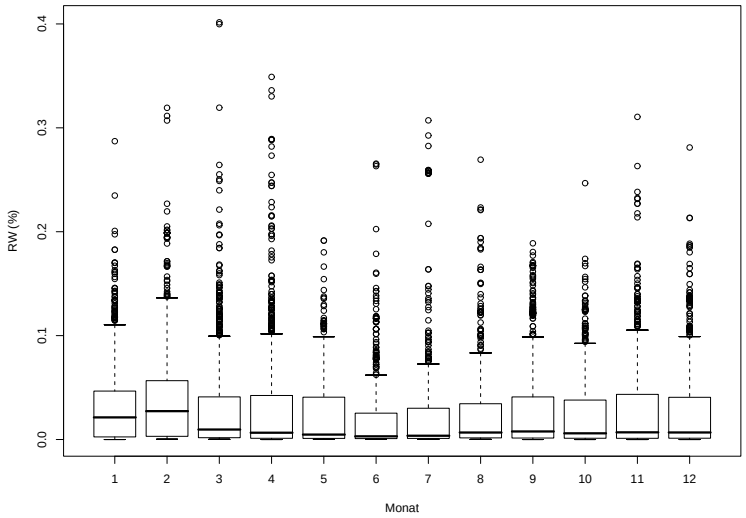


Abbildung 47: Reichweiten vom Sender GoTV abhängig vom Monat

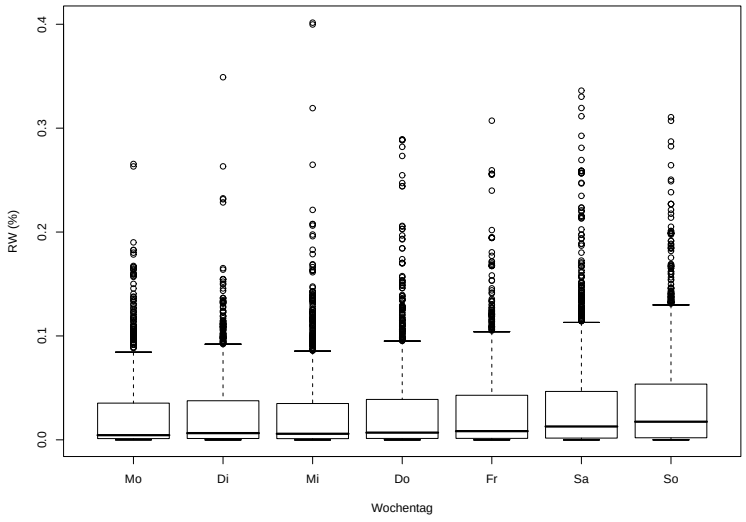


Abbildung 48: Reichweiten vom Sender GoTV abhängig vom Wochentag

15.2.11 MTV

Es stehen beim Sender MTV 11 549 Daten zur Verfügung. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 0,423 %. 10 295 Datensätze haben eine Reichweite kleiner 0,1 % – das entspricht 89,14 % der vorhandenen Daten. Aus diesem Grund wurde keine Prognose vorgenommen (siehe dazu auch Kapitel 15.2.10).

Abbildung 49 zeigt die Reichweiten abhängig von der Beginnzeit der Werbespots. Man sieht, dass im Zeitraum von 0:00 Uhr bis 9:00 Uhr keine Daten zur Verfügung standen.

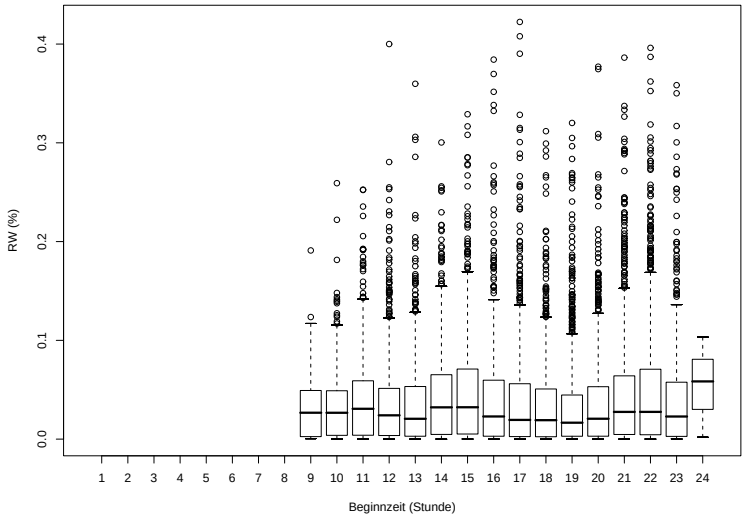


Abbildung 49: Reichweiten vom Sender MTV abhängig von der Beginnzeit

15.2.12 PulsTV

Es stehen beim Sender PulsTV 5 430 Daten zur Verfügung. Das Minimum der Reichweiten liegt unter 0,001 %, das Maximum bei 0,333 %. 4 884 Datensätze haben eine Reichweite kleiner 0,1 % – das entspricht 89,94 % der vorhandenen Daten. Aus diesem Grund wurde auch hier keine Prognose durchgeführt (siehe auch Kapitel 15.2.10).

In Abbildung 50 sind wieder die Reichweiten in Abhängigkeit von der Beginnzeit zu finden. Auch hier waren nicht für alle Stunden Werte im Datenbestand enthalten.

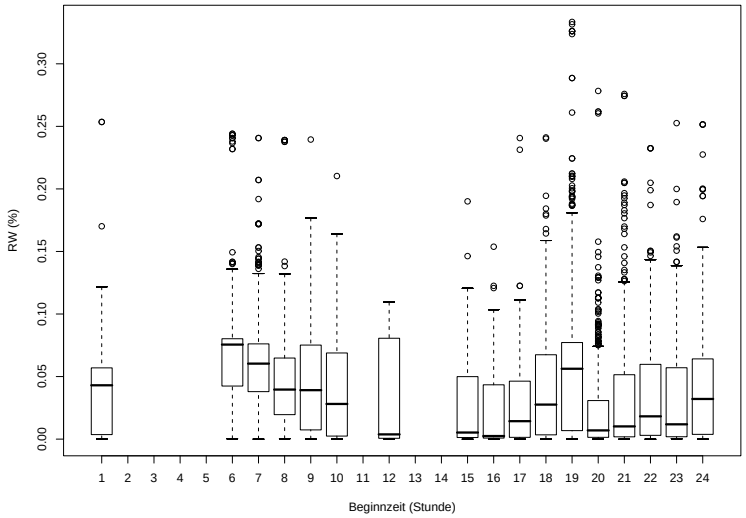


Abbildung 50: Reichweiten vom Sender PulsTV abhängig von der Beginnzeit

15.3 Zusammenfassung

Der Versuch die Reichweiten von Fernsehwerbespots zu prognostizieren hat nicht bei allen Sendern zu gleich guten Resultaten geführt. Die besten Ergebnisse hinsichtlich MAPE und R^2 konnten für den ORF erzielt werden. Betrachtet man die Prognosen jedoch genauer, sieht man, dass das Netz zwar den allgemeinen Trend (Rückgang der Reichweiten im Sommer, Anstieg der Reichweiten am Abend ...) in den Daten erkennt, im Detail aber des öfteren Ergebnisse liefert, die von den tatsächlichen Werten eine zu starke Abweichung aufweisen.

Die Parameter Monat, Wochentag, Temperatur, Feiertag und Beginnzeit des Werbespots liefern also für eine exakte Prognose nicht genügend Informationen. Eine Möglichkeit besteht darin, die Inputvariablen zu erweitern, so dass die Prognosen auch für den praktischen Einsatz relevant werden. Prädikatoren, die ergänzt werden können, sind beispielsweise der Marktanteil des Senders oder die Programmumgebung in die die Werbung eingebettet ist. Unter Programmumgebung ist die Art des Vorlauf- und Nachlaufprogramms einer Sendung gemeint – handelt es sich beispielsweise um Nachrichten, eine Serie, eine Reportage usw. Außerdem haben auch die Konkurrenzprogramme einen gewissen Einfluss, den man berücksichtigen könnte. Aufgrund der zahlreichen Sender und der häufig wechselnden Programme ist die Einbeziehung der Konkurrenzprogramme jedoch keine triviale Aufgabe. Ausreißer in den Reichweiten, die durch besondere Ereignisse, die nicht vorhersehbar waren, eintreten, werden jedoch immer zu Fehlprognosen führen – bei den betrachteten Daten wäre dies zum Beispiel die Papstwahl. Man könnte jedoch in den *input* im vorhinein bekannte Ereignisse, wie beispielsweise die Olympischen Spiele oder eine Fußball Weltmeisterschaft einbeziehen. Solche Ereignisse erhöhen in der Regel die Anzahl der Zuseher.

Aufgrund der schlechteren Ergebnisse der deutschen Sender, liegt die Vermutung nahe, dass bei diesen Sendern die Vor- und Nachlaufprogramme, sowie die Konkurrenzprogramme einen stärkeren Einfluss auf das Seherverhalten ausüben als beim ORF. Es ist auch zu beobachten, dass bei den deutschen Sendern der saisonale Trend weniger stark ausgebildet ist. In [Web00] wird außerdem darauf hingewiesen, dass die Privatsender weniger ritualisiert sind – vor allem am Abend nimmt die Variabilität aufgrund der vermehrten Werbeunterbrechungen zu. Weiters wird erwähnt, dass die Prognose für die Abendstunden am schwierigsten ist. Grund dafür ist die größere Streuung der Reichweiten am Abend (siehe dazu sämtliche Abbildungen der Reichweiten in Abhängigkeit der Beginnzeit). Das Frühjahr ist außerdem schwieriger zu prognostizieren als der Winter – der Grund dafür ist das Wetter, das eine gewisse Auswirkung auf das Fernsehverhalten der Zuseher hat und im Frühjahr am schwierigsten vorhersehbar ist. Dies kam in der Arbeit vor allem bei der Prognose des Monats Juli zu tragen, wo aufgrund der im Durchschnitt höheren Temperaturen ein Überschätzung der Reichweiten stattfand.

Die Einbeziehung der Temperatur hat sich für *ex-ante* Prognosen also als nicht immer ideal herausgestellt, da die Gefahr besteht, dass die Temperaturen stark von den Mitteltemperaturen abweichen. Für langfristige Prognosen sollte man auf diese Variable eher verzichten und stattdessen konstante, im vorhinein bekannte Prädikatoren verwenden. Das die Temperatur aber trotzdem einen

Einfluss auf den Fernsehkonsum hat ist unbestritten – für kurzfristige Prognosen, für die die voraussichtlichen Temperaturen mit einer gewissen Wahrscheinlichkeit vorhergesagt werden können, ist die Einbeziehung der Temperatur also durchaus zu überlegen.

Hinsichtlich der abhängigen Variablen, also der Reichweiten, könnte man ebenfalls einen Versuch zur Verbesserung unternehmen, indem man nicht die absoluten Reichweiten prognostiziert, sondern sie in Beziehung zu einer durchschnittlichen Reichweite setzt und versucht die Abweichungen vorherzusagen.

Laut [Web00] ist das Fernsehverhalten von Erwachsenen schwieriger zu prognostizieren als jenes von Kindern. Man sieht also, dass für die Prognosen in dieser Arbeit nicht die einfachsten Voraussetzungen angenommen wurden.

Abgesehen von der Ergänzung von Variablen hat sich gezeigt, dass die Intervalle der Beginnzeiten nicht zu groß angenommen werden dürfen. Vor allem beim Sender ORF sind innerhalb einer Stunde starke Schwankungen zu beobachten, die das Netz bei stündlichen Intervallen nicht differenzieren kann. Eine Verkürzung der Intervalle bei den deutschen Sendern bietet daher ebenfalls eine Möglichkeit der Verbesserung.

Im Kapitel 13 wurden bereits einige alternative Netzwerktypen und Lernverfahren angesprochen – man sieht also, dass es auch hinsichtlich der Struktur der Netze noch eine Vielzahl an Anknüpfungspunkten zur Optimierung gibt.

Abbildungsverzeichnis

1 Mehrschichtiges *feed-forward*-Netz 5
2 Tangens-Hyperbolicus tanh(x) 7
3 Logistische Aktivierungsfunktion 7
4 Konvergieren in ein lokales Minimum 14
5 Auftreten des *flat-spot* Problems 14
6 Auftreten von Oszillation 15
7 Überspringen des globalen Minimums 15
8 Rprop – Fall 1 19
9 Rprop – Fall 2 20
10 Rprop – Fall 3 20
11 Rprop – Fall 4 20
12 *Cascade-correlation* Architektur 23
13 Beispiel für *crossover* 25
14 Stopped-Training-Punkt 28
15 Reichweiten vom Sender ORF abhängig vom Monat 65
16 Reichweiten vom Sender ORF abhängig von der Temperatur . . 66
17 Reichweiten vom Sender ORF abhängig vom Wochentag 66
18 Reichweiten von ORF1 abhängig vom Monat 68
19 Reichweiten von ORF1 abhängig von der Temperatur 68
20 Reichweiten von ORF1 abhängig von der Beginnzeit 69
21 Reichweiten von ORF1 abhängig vom Wochentag 69
22 Reichweiten vom Sender ORF2 abhängig vom Monat 70
23 Reichweiten vom Sender ORF2 abhängig von der Temperatur . . 70
24 Reichweiten vom Sender ORF2 abhängig von der Beginnzeit . . . 71
25 Reichweiten vom Sender ORF2 abhängig vom Wochentag 71
26 Prognose für den Sender ORF für den Februar 2006 73
27 Prognose für den Sender ORF1 für den Februar 2006 73
28 Prognose für den Sender ORF2 für den Februar 2006 75
29 Prognose für den Sender ORF für den Mai 2006 75
30 Prognose für den Sender ORF1 für den Mai 2006 76
31 Prognose für den Sender ORF2 für den Mai 2006 76
32 Prognose für den Sender ORF für den Juli 2006 77
33 Prognose für den Sender ORF1 für den Juli 2006 77
34 Prognose für den Sender ORF2 für den Juli 2006 78
35 Prognose für den Sender ORF2 für den Juli 2006 bei Erhöhung der Mitteltemperatur um 5 Grad Celsius 78
36 Prognose für den Sender ORF2 für den Juli 2006 ohne Einbeziehung der Temperatur 80
37 Prognose für das Jahr 2006 80
38 Reichweiten vom Sender ATV abhängig von der Beginnzeit . . . 81
39 Reichweiten vom Sender RTL abhängig von der Beginnzeit . . . 83
40 Reichweiten vom Sender SAT1 abhängig von der Beginnzeit . . . 85
41 Reichweiten vom Sender PRO7 abhängig von der Beginnzeit . . . 87
42 Reichweiten vom Sender RTL2 abhängig von der Beginnzeit . . . 89
43 Reichweiten vom Sender VOX abhängig von der Beginnzeit . . . 90

44 Reichweiten vom Sender SRTL abhängig von der Beginnzeit . . . 92
45 Reichweiten vom Sender KAB1 abhängig von der Beginnzeit . . 94
46 Reichweiten vom Sender GoTV abhängig von der Beginnzeit . . 95
47 Reichweiten vom Sender GoTV abhängig vom Monat 96
48 Reichweiten vom Sender GoTV abhängig vom Wochentag 96
49 Reichweiten vom Sender MTV abhängig von der Beginnzeit . . . 97
50 Reichweiten vom Sender PulsTV abhängig von der Beginnzeit . . 98

Tabellenverzeichnis

1 Gütekriterien für die Sender ORF1, ORF2 und ORF bei verschiedenen Netzwerkkonfigurationen 72
2 Gütkriterien für den Sender ATV bei verschiedenen Netzwerkkonfigurationen 82
3 Gütekriterien für den Sender RTL bei verschiedenen Netzwerkkonfigurationen 84
4 Gütekritierien für den Sender RTL bei verschiedenen Netzwerkkonfigurationen 84
5 Gütekritierien für den Sender SAT1 bei verschiedenen Netzwerkkonfigurationen 86
6 Gütekriterien für den Sender PRO7 bei verschiedenen Netzwerkkonfigurationen 88
7 Gütekriterien für den Sender PRO7 bei verschiedenen Netzwerkkonfigurationen 88
8 Gütekriterien für den Sender RTL2 bei verschiedenen Netzwerkkonfigurationen 89
9 Gütekriterien für den Sender VOX bei verschiedenen Netzwerkkonfigurationen 91
10 Gütekriterien für den Sender VOX bei verschiedenen Netzwerkkonfigurationen 91
11 Gütekriterien für den Sender SRTL bei verschiedenen Netzwerkkonfigurationen 92
12 Gütekriterien für den Sender SRTL bei verschiedenen Netzwerkkonfigurationen 93
13 Gütekriterien für den Sender KAB1 bei verschiedenen Netzwerkkonfigurationen 94

Literatur

[And97] Anders, U.: *Statistische neuronale Netze.* München : Verlag Vahlen, 1997

[BD02] Bortz, J. ; Döring, N.: *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler.* Heidelberg : Springer Verlag, 2002

[Bis95a] Bishop, C. M.: *Neural Networks for Pattern Recognition.* Oxford : Oxford University Press, 1995

[Bis95b] Bishop, C. M.: Training with Noise is Equivalent to Tikhonov Regularization. In: *Neural Computation 7* (1995)

[CDS90] Cun, Y. L. ; Denker, J. S. ; Solla, S. A.: Optimal Brain Damage. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 2.* Morgan Kaufmann, 1990

[Cha89] Chauvin, Y.: A back-propagation algorithm with optimal use of hidden units. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 1.* Morgan Kaufmann, 1989

[Fah88] Fahlman, S. E.: An empirical study of learning speed in back-propagation networks. Pittsburgh, PA, 1988. – Forschungsbericht

[FL90] Fahlman, S. E. ; Lebiere, Ch.: The Cascade-Correlation Learning Architecture. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 2.* Morgan Kaufmann, 1990

[Heb49] Hebb, D. O.: *The Organization of Behavior.* New York : Wiley, 1949

[HP89] Hanson, S. J. ; Pratt, L. Y.: Comparing biases for minimal network construction with back-propagation. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 1.* Morgan Kaufmann, 1989

[HS93] Hassibi, B. ; Stork, D. G.: Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In: *Advances in Neural Information Processing Systems 5.* Morgan Kaufmann, 1993

[HS02] Hüttner, M. ; Schwarting, U.: *Grundzüge der Marktforschung.* München : Oldenburg, 2002

[HSW93] Hassibi, B. ; Stork, D. G. ; Wolff, G. J.: Optimal Brain Surgeon and General Network Pruning. In: *Proc. of the IEEE Int. Conf. on Neural Networks* (1993)

[Koh00] Kohonen, T.: *Self-Organizing Maps.* Heidelberg, New York : Springer Verlag, 2000

[Luk05] Lukassek, E.: *Künstliche Neuronale Netze zur Prognose ökonomischer Zeitreihen.* Aachen : Shaker Verlag, 2005

[Mec94] Mechler, B.: *Lernfähige, rechnergestützte Entscheidungsunterstützungssysteme.* Oftersheim, Universität Mannheim, Diss., 1994

[MS89] Mozer, M. C. ; Smolensky, P.: Skeletonization: A technique for trimming the fat from a network via relevance assessment. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 1.* Morgan Kaufmann, 1989

[NKK94] Nauck, D. ; Klawonn, F. ; Kruse, R.: *Neuronale Netze und Fuzzy-Systeme.* Braunschweig/Wiesbaden : Verlag Vieweg, 1994

[Oss90] Ossen, A.: *Zur Modularisierung und Interpretierbarkeit Neuronaler Netze*, Technische Universität Berlin, Diss., 1990

[Pod94] Podding, T.: Mittelfristige Zinsprognosen mittels KNN und ökonometrischer Verfahren. In: Rehkugler, H. (Hrsg.) ; Zimmermann, H. G. (Hrsg.): *Neuronale Netze in der Ökonomie.* München : Verlag Vahlen, 1994

[PR94] Pfister, M. ; Rojas, R.: Hybrid Learning Algorithms for Neural Networks – The adaptive Inclusion of Second Order Information. In: *Proc. 4th Dortmund Fuzzy Days* (1994)

[RB92] Riedmiller, M. ; Braun, H.: RPROP – A Fast Adaptive Learning Algorithm. In: *Proc. of ISCIS VII* (1992)

[RH93] Riedmiller, M. ; H.Braun: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: *Proc. of the IEEE Int. Conf. on Neural Networks* (1993)

[RHW86] Rumelhart, D. E. ; Hinton, G. E. ; Williams, R. J.: Learning Internal Representations by Error Propagation. In: Rumelhart, D. E. (Hrsg.) ; McCleeland, J. L. (Hrsg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1.* Cambridge, MA : MIT Press, 1986

[Roj93] Rojas, R.: *Theorie der neuronalen Netze.* Berlin : Springer Verlag, 1993

[SHG90] Schöneburg, E. ; Hansen, N. ; Gawelczyk, A.: *Neuronale Netzwerke.* Haar bei München : Markt- und Technik Verlag, 1990

[Web95] Weber, R.: *Vergleich der Prognosen von Künstlichen Neuronalen Netzen, von Arima-Modellen und der Spektralanalyse mit unterschiedlichen Gütemaßen.* Berlin, 1995

[Web00] Weber, R.: *Prognosemodelle zur Vorhersage der Fernsehnutzung.* München : Verlag Reinhard Fischer, 2000

[Web01] Weber, R.: Datenanalyse mittels Neuronaler Netze am Beispiel des Publikumserfolgs von Spielfilmen. In: *Zeitschrift für Medienpsychologie* Heft 4 (2001), S. 164–176

[Wer88] Werbos, P. J.: Backpropagation: Past and Future. In: *Proc. of the IEEE Int. Conf. on Neural Networks* (1988)

[Zel94a] Zell, A.: *Simulation neuronaler Netze.* Bonn : Addison Wesley Longmann Verlag, 1994

[Zel94b] Zell, A.: *SNNS, Stuttgarter Neural Networksimulator – User Manual, Version 4.2.* Stuttgart, 1994

[Zim94] Zimmermann, H. G.: Neuronale Netze als Entscheidungskalkül. In: Rehkugler, H. (Hrsg.) ; Zimmermann, H.G. (Hrsg.): *Neuronale Netze in der Ökonomie.* München : Verlag Vahlen, 1994

Internetquellen

[agf] *AGF – Arbeitsgemeinschaft Fernsehforschung*. http://www.agf.de/. – Stand: 10.8.2006

[bra] *California Scientific/ Brain Maker Neural Network Software*. http://www.calsci.com/. – Stand: 10.8.2006

[dim] *NeuroDimension*. http://www.nd.com/. – Stand: 10.8.2006

[eas] *EasyNN*. http://www.easynn.com/. – Stand: 10.8.2006

[int] *Neural networks software for data mining*. http://www.alyuda.com/. – Stand: 10.8.2006

[joo] *Joone - Java Objected Oriented Neural Engine*. http://www.jooneworld.com/. – Stand: 10.8.2006

[med] *Media-Analyse*. http://media-analyse.at/. – Stand: 10.8.2006

[neu] *Advanced Neural Network and Genetic Algorithm Software*. http://www.wardsystems.com/. – Stand: 10.8.2006

[orf] *ORF Daten Fakten Trend*. http://mediaresearch.orf.at/. – Stand: 10.8.2006

[snn] *SNNS - Stuttgart Neural Network Simulator*. http://www-ra.informatik.uni-thuebingen.de/SNNS/. – Stand: 10.8.2006

[sta] *Data Mining, Statistical Analysis, Quality Control - STATISTICA Software*. http://www.statsoft.com/. – Stand: 8.8.2006