Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Bianca Ott

Erstellung von Bedarfsprognosen durch Künstliche Neuronale Netze am Beispiel von Backmengenempfehlungen im Einzelhandel

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Bianca Ott

Erstellung von Bedarfsprognosen durch Künstliche Neuronale Netze am Beispiel von Backmengenempfehlungen im Einzelhandel

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Michael Neitzke
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 13.05.2013

Bianca Ott

Erstellung von Bedarfsprognosen durch Künstliche Neuronale Netze am Beispiel von Backmengenempfehlungen im Einzelhandel

Stichworte
Künstliches Neuronales Netz, Bedarfsprognose, Zeitreihenprognose, kausale Prognose

Kurzzusammenfassung
Zuverlässige Bedarfsprognosen entscheiden im Einzelhandel oft über Gewinn oder Verlust. Bei einer geringen Gewinnmarge ist es unerlässlich, dass einerseits eine ausreichende Anzahl eines Artikels vorrätig ist, damit alle Käufer den gewünschten Artikel vorfinden aber andererseits so wenig Abschriften wie möglich produziert werden. Die vorliegenden Bachelorarbeit untersucht in wie fern sich Künstliche Neuronale Netze dazu eignen den Bedarf an Backwaren in Filialen eines Einzelhandel-Discounters für den Folgetag zuverlässig zu prognostizieren. Voraussetzung dafür ist eine Modellierung des Anwendungsfalls, um die passenden Informationen und die geeignete Form der Daten zur ermitteln, die ein Künstliches Neuronales Netz benötigt, um optimale Bedarfsprognosen liefern zu können. Untersucht werden in diesem Zusammenhang Zeitreihen-, kausale und kombinierte Prognosemodelle, die durch Künstliche Neuronale Netze implementiert werden. Die daraus resultierenden Prognoseergebnisse werden abschließend miteinander verglichen, um die bestmögliche Modellierungsform zu ermitteln.

Bianca Ott

Creation of demand forecasts by artificial neural networks using the example of baking amount recommendations in retail

Keywords
Artificial Neural Network, Demand Forecast, Time-Series Forecast, Causal Forecast

Abstract
Reliable demand forecasts in retail often decide over profit or loss. At a low profit margin, it is essential, that a sufficient quantity of an item is in stock, so that all buyers find the item they are looking for and at the same time there is as little overproduction as possible. The aim of this thesis is to investigate to what extent artificial neural networks are suitable to reliably predict the demand of baked goods in a retail store discounter, for the following day. This requires a modeling of the use case to determine the necessary information and the appropriate form of the data an artificial neural network needs in order to provide optimal demand forecasts. Therefore time series, causal and composite forecasting models are examined and implemented by artificial neural networks. In conclusion the resulting predictions are compared to determine the best forecasting model.

Inhaltsverzeichnis

Teil I.

Einführung

1. Einleitung

Zuverlässige Bedarfsprognosen entscheiden im Einzelhandel oft über Gewinn oder Verlust. Bei einer geringen Gewinnmarge ist es unerlässlich, dass einerseits eine ausreichende Anzahl eines Artikels vorrätig ist, damit alle Käufer den gewünschten Artikel vorfinden aber andererseits so wenig Abschriften wie möglich produziert werden.

Bei durchschnittlichen Gewinnmargen im Lebensmittel Einzelhandel von unter einem Prozent (Jacobsen, 2013) schlägt jeder zu viel produzierte Artikel, jede Stunde unnötige Lagerhaltung und jeder ungenutzte Regalmeter negativ zu Buche. Gleichzeitig kann ein guter Umsatz nur dann verbucht werden, wenn jeder einzelne Artikel möglichst oft verkauft wird. Die Zielsetzung muss also stets lauten: soviel wie nötig und so wenig wie möglich eines Artikels vorrätig zu haben. Doch wie lässt sich diese 'goldene' Menge, die nicht zu viel und nicht zu wenig sein darf, ermitteln? Herkömmlicher Weise wird sich eines großen Spektrums mathematischer Verfahren bedient. Von naiven Verfahren, die eine Prognose anhand der letzten Verkaufszahl ermitteln, bis hin zu hochkomplexen statistischen Verfahren, die ohne die Betreuung durch Wirtschaftsmathematiker nicht eingesetzt werden können. Seit Ende des letzten Jahrtausends kommen aber vermehrt Künstliche Neuronale Netze (KNN) zum Einsatz (Vemuri und Rogers, 1994), die deutlich bessere Ergebnisse liefern, als die naiven Verfahren und deutlich weniger komplex in der Anwendung sind als die hochspezialisierten statistischen Prognoseverfahren.

Die vorliegenden Bachelorarbeit untersucht in wie fern sich Künstliche Neuronale Netze dazu eignen den Bedarf an Backwaren in Filialen eines Einzelhandel-Discounters für den Folgetag zuverlässig zu prognostizieren. Voraussetzung dafür ist eine Modellierung des Anwendungsfalls, um die passenden Informationen und die geeignete Form der Daten zur Ermitteln, die ein KNN benötigt, um optimale Bedarfsprognosen liefern zu können.

1.1. Problemstellung

Im Rahmen einer großen Softwarelösung für einen Einzelhandelsdiscounter hat die Firma *mgm technology partners* ein Modul entwickelt, das Backmengenempfehlungen errechnet und diese jeder Filiale täglich per E-Mail zur Verfügung stellt.

Die Anforderung besteht darin für alle Filialen mit Backstraßen - also solche die Backwaren selbst backen - eine Backmengenempfehlung für den Folgetag zu erstellen. Diese muss taggenau, filialgenau und artikelgenau sein. Das heißt, dass für jeden Öffnungstag, jede Filiale und jeden dort gebackenen Artikel eine Backmengenempfehlung errechnet wird.

In der bestehenden Softwarelösung wird an jedem Tag t (Ermittlungszeitpunkt) für jede Filiale ein nächtlicher Batch gestartet, der für jeden dort gebackenen Artikel eine Backmengenempfehlung für den Tag $t+1$ (Prognosezeitpunkt) errechnet. Der Berechnungsalgorithmus[1] basiert auf den historischen Abverkaufszahlen dieser Filiale, die über die Scannerkassen und ein angebundenes Datawarehouse bereitgestellt werden. Aus dieser Datenmenge wählt der Algorithmus n Vergleichstage zu $t+1$, streicht den Tag mit dem geringsten Abverkauf und bildet über die restlichen Werte den Mittelwert. Die Vergleichbarkeit der Tage ist über die Gleichheit des Wochentages definiert. Dieser Vorgang ist in der folgenden Tabelle 1.1 exemplarisch dargestellt.

Vergleichstag	**Abverkaufzahl**
Montag KW 20	17
Montag KW 21	15
Montag KW 22	13
Montag KW 23	11
~~Montag KW 24~~	~~8~~
Montag KW 25	23
Montag KW 26	19
Summe Abverkauf	106
Bereinigte Summe	98
Mittelwert über bereinigte Summe	98/6=16
Prognose: Montag KW 27	16

Tabelle 1.1.: Mittelwertbildung über Vergleichstage mit $n = 7$

1.1.1. Abgrenzung

Die hier erwähnte Backmengenempfehlung ist das Endprodukt der Implementierung beim Kunden. Dessen Grundlage eine zuvor ermittelte Bedarfsprognose ist. Dieser prognostizierte Bedarf wird dann aber u.a. noch auf verschiedenen Zeitfenster verteilt, es werden Backblechbelegungen errechnet und Überhänge auf Zusatzblechen zusammengefasst. Am Ende dieses Prozesses, steht ein versandfertiges PDF mit allen nötigen Informationen für die einzelnen Filialleiter. Die Backmengenempfehlung ist, im Gegensatz zur Bedarfsprognose, nicht Gegenstand dieser Arbeit.

[1] Der Berechnungsalgorithmus darf aus lizenzrechtlichen Gründen hier nur vereinfacht und in Ansätzen dargestellt werden.

2. Motivation und Zielsetzung

Wie in Abschnitt 1.1 beschrieben, handelt es sich bei der vorliegenden Problemstellung um einen Anwendungsfall aus einer bereits produktiven Softwarelösung. Der beschriebene Ansatz ist seit Jahren beim Kunden im Einsatz und soll durch diese Arbeit eine mögliche Alternative aufgezeigt bekommen.

Dazu werden folgende Fragestellungen untersucht:

1. Wie lässt sich die Problemstellung optimal modellieren, um mit Hilfe von KNN gelöst zu werden?
2. Wie gut eignen sich KNN zur Ermittlung von Bedarfsprognosen für Backwaren im Einzelhandel und wie schneiden diese Prognosen im Vergleich zum bisher eingesetzten Prognoseverfahren ab?

Außerdem sollen während der Bearbeitung der genannten Fragestellungen, Erfahrungen bzgl. der Handhabbarkeit einer KNN-Lösung, im Umfeld einer klassischen Java Softwareentwicklung, gesammelt und kompakt zusammengefasst werden.

3. Aufbau der Arbeit

Die vorliegende Arbeit wird in den ersten Kapiteln die theoretischen Grundlagen legen, um Bedarfsprognosen einschätzen und die Funktionsweise KNN verstehen zu können. Den Anfang macht Kapitel 4 mit einer Einordnung von Bedarfsprognosen. Kapitel 5 wird zeigen, dass die Form und der semantische Zusammenhang der Daten, die einem Prognoseverfahren zur Verfügung gestellt werden, eine entscheidende Rolle bei der Prognose spielen. Kapitel 6 wird im Anschluss daran auf die Vor- und Nachteile von KNN als Prognoseverfahren eingehen. Abgeschlossen werden die theoretischen Grundlagen durch eine Einführung in die Funktionsweise KNN in Kapitel 7.

Kapitel 8 liefert die praktischen Grundlagen und dient als Überleitung zum praktischen Teil dieser Arbeit. Es beschreibt wie KNN konkret eingesetzt und wie Bedarfsprognosen mit KNN modelliert werden.

Der praktische Teil dieser Arbeit umfasst die Kapitel 9, 10, 11 und 12. Kapitel 9 erläutert die Wahl des KNN-Frameworks. Kapitel 10 definiert die benutzte Datenbasis und beschreibt die Architektur des bestehenden und des zu Implementierenden Systems. In Kapitel 11 werden die zuvor erläuterten theoretisch Modellierungsaspekte, mit der konkreten Problemstellung und Datenbasis kombiniert und die konkrete Konfiguration der verwendeten KNN beschrieben. Die Darstellung der verschiedenen Versuchsaufbauten und die Auswertung der Ergebnisse wird in Kapitel 12 erfolgen.

Mit einem Fazit in Kapitel 13 und einem Auslick auf mögliche Erweiterungen, in Kapitel 14, wird diese Arbeit abgeschlossen.

Teil II.

Theoretische Grundlagen

4. Bedarfsprognosen und ihre Einordnung

Der Bedarf ist die am Markt tatsächlich auftretende Nachfrage nach Gütern (Kirchgeorg und Piekenbrock, 2013) und die Bedarfsprognose ist somit die Vorhersage der insgesamt und maximal benötigten Menge eines Produktes am Markt zu einem zukünftigen Zeitpunkt. Mit *maximal* ist gemeint, dass nicht mehr prognostiziert werden sollte, als tatsächlich benötigt wird und *insgesamt* deutet an, dass eine ideale Bedarfsprognose keine Überschüsse enthält - denn Abschriften vermindern den Gewinn und *"Fehlmengen besitzen eine sofortige Umsatzrelevanz durch verlorene Kaufakte"* (Crone, 2010, S. 18-19).

Die Anforderung nicht über den Bedarf hinaus zu prognostizieren (*maximal* benötigte Menge) ist anhand des historischen Absatzes[1] erlern- bzw. errechenbar. Schwieriger ist hingegen die Prognose des Mindestbedarfs (*insgesamt* benötigte Menge) unter Verwendung der vergangenen Abverkäufe.

Zur Verdeutlichung folgendes, an die gegebene Problemstellung angelehntes, Beispiel:
Am Ende eines Verkaufstages t sind 50 Mandelhörnchen verkauft worden und es liegen:

A. noch 20 weitere im Korb oder

B. keine weiteren im Korb

Im **Fall A** kann davon ausgegangen werden, dass die abgesetzten 50 Mandelhörnchen den Bedarf für Tag t darstellen, da noch Einheiten vorhanden waren, die hätten verkauft werden können, die Nachfrage die 50 Stück aber nicht überstiegen hat. Eine Ideale Bedarfsprognose für Tag t hätte somit 50 statt der 70 gebackenen Mandelhörnchen vorausgesagt.

Im **Fall B** kann der Absatz der 50 Stück hingegen nicht mit dem Bedarf gleichgesetzt werden. Es ist in diesem Fall durchaus wahrscheinlich, dass weitere Nachfrage bestand, der Artikel jedoch (frühzeitig) ausverkauft war - der Bedarf wäre somit größer als der Absatz. Eine ideale Bedarfsprognose hätte also einen Bedarf von $50 + x$ voraussagen müssen.

Die Ermittlung der Größe von x ist jedoch im Allgemeinen nicht eindeutig möglich. Durch den fehlenden Kundenkontakt und die Selbstbedienung im Einzelhandel ist es meist nicht möglich zu ermitteln, wie viele Kunden einen gewünschten Artikel nicht vorfanden und welche Anzahl des Artikels sie gekauft hätten, wäre er noch vorhanden gewesen. Darüber hinaus wäre denkbar, dass das Fehlen des Artikels selbst mit einer stagnierten Nachfrage gleichzusetzen ist . Lecker duftende, goldbraune Mandelhörnchen unterliegen möglicherweise einer sehr hohen

[1] Im Einzelhandel auch als *Abverkauf* bezeichnet.

Nachfrage. Doch sobald sie ausverkauft sind, wissen die meisten Kunden nicht, dass es diesen Artikel überhaupt gab, so dass keine weitere Nachfrage besteht, im Grunde genommen aber durchaus ein weiterer Bedarf. Der Bedarf unterscheidet sich also durch Fehlmengen vom Absatz (Crone, 2010).

Wie Abschnitt 5.2.6 zeigen wird, gibt es viele Verfahren zur Ermittlung von Bedarfsprognosen und trotz ihrer Vielfältigkeit und deutlichen Unterschiede haben alle gemein, dass der Wert, der prognostiziert werden soll in seinen historischen Ausprägungen vorhanden sein muss. Um also den Bedarf eines Artikels für die Zukunft prognostizieren zu können, wird der Bedarf des Artikels in der Vergangenheit benötigt. *"If you want to produce a demand forecast, then you need to use the demand history as input. In practice, it means that you need to correct your sales history to reflect the demand"* (Singh, 2008, o.S.).

Sollte es also nicht möglich sein, den Absatz eines Artikels so zu korrigieren, dass er dem tatsächlichen Bedarf zumindest nahe kommt, dann kann keine Bedarfsprognose, sondern lediglich eine Absatzprognose erstellt werden. Je nach Anwendungsfall kann die Genauigkeit einer Absatzprognose ausreichend sein, meist dann, wenn die Abweichungen zwischen Absatz und Bedarf gering sind. Bei größeren Abweichungen werden sich die Planungsfehler oder Fehleinschätzungen der Vergangenheit jedoch negativ auf die Zukunft auswirken. Wenn der tatsächliche Bedarf an Mandelhörnchen in der Vergangenheit größer war als der Absatz, dem Prognoseverfahren aber nur die historischen Absatzzahlen zur Verfügung stehen, dann wird auch für die Zukunft einen zu geringer "Bedarf" an Mandelhörnchen vorhergesagt werden, was wiederum zu Gewinneinbußen führen wird.

Sollte also eine Bedarfsprognose B angestrebt werden, muss ein Algorithmus zur Annäherung des historischen Bedarfs entwickelt werden, um aus den Absatzzahlen A und meist weiteren Faktoren P den Bedarfs anzunähern. Meist handelt es sich dabei um eine automatisierte Rechenvorschrift $B = f(A, P)$, wobei dies weder die einzige noch die zwangsläufig vielversprechendste Methode darstellt. Es ist auch denkbar, dass die Korrekturen manuell durch Disponenten oder Domänenexperten durchgeführt werden. Dies ist ein sehr aufwendiges Verfahren, aber in der Regel sehr erfolgreich.

5. Modellierung von Prognoseverfahren im Handel

Ein Prognoseverfahren ist ein System, das Werte als Eingabe erhält und diese nach bestimmten Regeln miteinander verknüpft, um als Ergebnis der Verknüpfung den Prognosewert zu liefern (Hansmann, 1983). Zur Ermittlung dieser Prognosewerte wurde eine Vielzahl konkurrierender Prognoseverfahren entwickelt, die sich in zwei große Klassen einteilen lassen: In *urteilsbasierte* und *datenbasierte* Verfahren.

Die Einteilung stammt aus (Staub, 2010), wird aber in anderen Werken, mit anderer Terminologie, ebenso gewählt. (Crone, 2010) spricht von Subjektiven- und Objektiven-Verfahren und (Makridakis u. a., 1998) von qualitativen und quantitativen. (Staub, 2010) plädiert jedoch für die Formulierung urteilsbasiert und datenbasiert, da *"[...] die gewählte Dichotomie trennschärfer erscheint als die in der Prognoseliteratur dominierende Einteilung in "subjektiv und objektiv" bzw. "intuitiv und analytisch", da auch objektive (bzw. analytische) Verfahren auf subjektive (bzw. intuitive) Überlegungen angewiesen sind"* (Staub, 2010, S. 217).

5.1. Urteilsbasierte Prognoseverfahren

Urteilsbasierte Prognoseverfahren *"sind ein Sammelbegriff für zweckmäßige, methodisch erarbeitete Prognoseregeln [...] ohne schematisches Prognosemodell"* (Hansmann, 1983, S. 18)[1].

Sie zeichnen sich durch folgende Eigenschaften aus (Hansmann, 1979):

1. Die der Prognose zugrundeliegende Theorie ist nur schwach ausgebildet und/oder enthält viele subjektive, d.h. nicht unmittelbar nachprüfbare Elemente.
2. Die statistisch-mathematischen Instrumente treten in ihrer Bedeutung für die Prognose zurück.
3. Der Einsatz von *Experten*, deren Erfahrung auf Spezialgebieten für die Prognose nutzbar gemacht werden, ist stark verbreitet.

Urteilsbasierte Prognoseverfahren eignen sich für eine Prognose, wenn eine Formalisierung nicht möglich oder unwirtschaftlich ist und/oder datenbasierte Verfahren nicht eingesetzt wer-

[1] Hansmann bezeichnet die urteilsbasierten Prognoseverfahren allerdings abweichend als heuristische Prognoseverfahren, von heuriskein (gr.) = systematisch suchen.

den können. Dies ist insbesondere bei der Erfassung von schwer strukturierbaren oder quantifizierbaren Marktentwicklungen der Fall. Es ist beispielsweise schwierig, Parameter wie Werbung oder Produktionsengpässe, die hohe Schwankungen in den Abverkaufszahlen auslösen können, in die Zeitreihen objektiver Verfahren zu integrieren (Crone, 2010). Armstrong (2001) spricht in diesem Zusammenhang von *combined forecasts*, was u.a. dem Kombinieren von objektiven Prognoseverfahren mit vor- oder nachgestellten Bewertungen und Anpassung durch Experten entspricht.

Das Spektrum der urteilsbasierten Prognoseverfahren ist umfangreich. Häufige Erwähnung finden in der Literatur folgende Verfahren (u.a. Hansmann (1983), Makridakis u. a. (1998), Crone (2010)):

1. Expertenbefragungen
2. die Delphi-Methode
3. die Szenario-Technik

Da die urteilsbasierten Prognoseverfahren für diese Arbeit keine Rolle spielen, wird auf die Vorstellung der einzelnen Verfahren verzichtet. Eine anschauliche und dennoch knappe Beschreibung der Delphi-Methode und der Szenario-Technik findet sich bei (Hansmann, 1983, S. 18-26).

5.2. Datenbasierte Prognoseverfahren

Datenbasierte Prognoseverfahren können angewandt werden, wenn folgende Annahmen zutreffen (Makridakis u. a., 1998):

1. Informationen über die Vergangenheit stehen zur Verfügung.
2. Diese Informationen können in Form numerischer Werte quantifiziert werden.
3. Es kann angenommen werden, dass zumindest einige Aspekte der Vergangenheit Auswirkungen auf die Zukunft haben.

All diese Annahmen treffen auf den vorliegenden Anwendungsfall zu, so dass Datenbasierte Prognoseverfahren in dieser Arbeit Anwendung finden.

Die datenbasierten Prognoseverfahren unterteilen sich wiederum in Verfahren zur:

- Zeitreihenprognose
- kausalen Prognose
- kombinierten Prognose

Vor der Wahl des richtigen Prognoseverfahrens stehen also grundlegende Modellierungsfragen: Welche zugrundeliegenden Kenntnisse über die Domäne und den Gegenstand der Prognose stehen zur Verfügung? Wovon wird der Prognosegegenstand beeinflusst? Welches Verfahren braucht welche Informationen, um eine optimale Prognose ermitteln zu können? In welcher Form müssen diese Informationen präsentiert werden? Die folgenden Abschnitte werden auf diese Modellierungsaspekte eingehen und Abschnitt 5.2.6 liefert eine graphische Übersicht über die bekanntesten datenzentrierten Prognoseverfahren.

5.2.1. Zeitreihenprognosen

Eine Zeitreihe ist eine *"Folge von Werten einer Variablen, die sich auf aufeinander folgende Zeitpunkte oder Zeiträume bezieht"* (Auer, 2013, o.S.). Dies können beispielsweise sekündliche Zugriffszahlen auf einen viel frequentierten Webserver sein, stündlich schwankender Energiebedarf eines Haushalts, tägliche Abverkaufszahlen eines bestimmten Produkts, wöchentliche Bestellmengen eines Versandhandels oder das jährliche Bruttoinlandsprodukt eines Landes.

Die Modellierung von KNN zur Zeitreihenprognose beruht auf der Vorhersage einer *abhängigen* Variablen $*y_{t+h}$. Ermittelt wird diese Vorhersage unter Verwendung von n historischen Realisationen derselben Variablen $y_t, y_{t-1}, ..., y_{t-n-1}$ (Crone, 2010). Hierbei spezifiziert:

- h den Prognosehorizont, also den Abstand des Prognosezeitpunkts vom Ermittlungszeitpunkt. Im einfachsten Fall, wie auch in dieser Arbeit, ist $h = 1$, der Prognosezeitpunkt also der Folgetag des Ermittlungszeitpunkts.
- n das historische Zeitfenster (engl. *time-window*), das definiert wie viele historische Realisationen von y in die Prognose einfließen.

Eine Zeitreihenprognose charakterisiert sich also dadurch, dass sie zukünftige Realisationen eines Wertes anhand der historischen Realisationen desselben Wertes vorhersagt. Die zu prognostizierenden Werte hängen ausschließlich von ihrer zeitlichen Entwicklung ab.

Ziel der Zeitreihenprognose ist die funktionale Abbildung zwischen der zu prognostizierenden Variable und ihren historischen Ausprägungen in der Annahme, dass die Vergangenheitswerte Strukturen, Zusammenhänge oder Gesetzmäßigkeiten enthalten, die sich auf zukünftige Daten extrapolieren lassen (Makridakis u. a., 1998):

$$*y_{t+1} = f(y_t, y_{t-1}, ..., y_{t-n-1})^{[2]} \tag{5.1}$$

Beispiel: Absatzprognose für Gitarren
Anhand monatlich vorliegender Absatzzahlen soll eine Absatzprognose[3] für Gitarren AP_{Git} für

[2] Angelehnt an (Zhang u. a., 1998)

[3] Das Beispiel sieht eine Absatzprognose statt einer Bedarfsprognose vor, da diese einfacher nachzuvollziehen ist, weil sie ohne einen Algorithmus zur Bedarfsannäherung auskommt.

den Folgemonat ($h = 1$) erstellt werden. Bezogen auf die folgenden (hypothetischen) Absatzzahlen seien der Ermittlungszeitpunkt t der Monat Mai (05) 2013 und der Prognosezeitpunkt $t + 1$ der Monat Juni (06) 2013.

Monat	Absatz 2010	Absatz 2011	Absatz 2012	Absatz 2013
01	94	52	60	52
02	58	38	50	33
03	69	48	69	92
04	68	49	42	48
05	71	37	36	t
06	66	39	38	$t+1$
07	63	35	45	
08	47	44	44	
09	48	36	65	
10	59	87	72	
11	45	46	64	
12	43	69	44	

Tabelle 5.1.: Hypothetischer Absatz von Gitarren eines bestimmten Modells in einem Gitarrenfachgeschäft - für die Jahre 2010, 2011, 2012 und 2013 (soweit vorhanden)

Um diese Problemstellung als Zeitreihenprognose zu modellieren, wären folgende Herangehensweisen denkbar:

- Wenn man davon ausgeht, dass die Verkaufsentwicklung der unmittelbaren Vergangenheit den größten Einfluss auf den zukünftigen Verkauf hat, wäre es sinnvoll, ein relativ kleines Zeitfenster zu wählen und n beispielsweise auf 6 Monate zu setzen. Somit ergäbe sich folgende Funktion:
  $AP_{Git}06/13 = f(48, 92, 33, 52, 44, 64)$
- Es wäre aber auch denkbar, dass sich der Absatz eines zukünftigen Monats ähnlich verhält wie der Absatz desselben Monats in den vergangenen Jahren. In diesem Fall wäre es möglich, n auf den Absatz des Monats 06 in den letzten 3 Jahren zu setzen. Somit ergäbe sich folgende Funktion:
  $AP_{Git}06/13 = f(38, 39, 66)$

5.2.2. Kausale Prognosen

Bei der Modellierung der Bedarfsprognose in Form einer kausalen Prognose wird die Beziehung der zu prognostizierenden Variable $*y_{t+h}$ nicht zu den abhängigen Variablen hergestellt, sondern zu sogenannten *unabhängigen* Variablen x_i zum Prognosezeitpunkt $t+h$. Diese Variablen werden im Allgemeinen Einflussvariablen (engl. *predictor variables*) [4] oder Erklärungsvariablen

[4] (Zhang u. a., 1998)

(engl. explanatory variables)[5] genannt. Dabei handelt es sich um zukünftige Realisierungen von Variablen, die entweder bekannt sind (z.B. der morgige Wochentag) oder ebenfalls prognostiziert werden (z.B. die morgige Regenwahrscheinlichkeit).

Eine kausale Prognose charakterisiert sich also dadurch, dass zukünftige Realisationen eines Wertes ($*y_{t+h}$) anhand von Variablen bestimmt werden, die ausreichende Informationen über den Prognosegegenstand $*y$ beinhalten. Die zu prognostizierenden Werte hängen also ausschließlich von äußeren Einflüssen zum Prognosezeitpunkt ab.

Ziel der kausalen Prognose ist die funktionale Abbildung zwischen der zu prognostizierenden Variable und der Einflussvariablen:

$$*y_{t+1} = f(x_{1,t}, x_{2,t}, ..., x_{i,t})^{6} \tag{5.2}$$

Es wurde bewusst darauf verzichtet die Abhängigkeit zwischen Einflussvariablen und dem Prognosegegenstand, als *kausal* zu bezeichnen. Die Terminologie - kausale Prognose - ist in dieser Hinsicht irreführend, da die meisten Prognosesettings nicht darauf ausgelegt sind die Kausalbeziehung zwischen x_i und $*y$ nachzuweisen, sondern lediglich Variablen x_i suchen, die Informationen über $*y$ enthalten und durch die der Prognosegegenstand hinreichend bestimmt wird. Eine kausale Abhängigkeit ist natürlich wünschenswert, aber keine zwingende Voraussetzung, um gute Prognosen liefern zu können. Ein gutes Prognoseergebnis ist somit kein Beweis für eine Kausalbeziehung zwischen x_i und $*y$ - dieser muss, wenn erwünscht, auf anderem Wege erbracht werden[7].

Beispiel: Absatzprognose für Regenschirmen
Anhand von täglich vorliegenden Absatzzahlen, soll eine Absatzprognose für Regenschirme AP_{Reg} für den Folgetag ($h = 1$) erstellt werden.

Um die Problemstellung als kausale Prognose modellieren zu können, müssen die Einflussfaktoren auf den Gegenstand der Prognose bekannt sein. Im Fall einer Absatzprognose für Regenschirme ist dies sicherlich einfacher als bei der oben skizzierten Prognose für Gitarren.

Eine einfache Herangehensweise wäre, den Abverkauf für $t + 1$ anhand der vorhergesagten prozentualen Regenwahrscheinlichkeit (x_R) für $t + 1$ zu ermitteln.

Denkbar wären aber sicherlich noch viele weitere Einflussvariablen x_i, wie beispielsweise:

[5] (Makridakis u. a., 1998)

[6] Angelehnt an (Zhang u. a., 1998)

[7] Hansmann postulierte 1983, dass *"ausdrücklich betont werden [muss], dass eine Kausalbeziehung zwischen zwei Zeitreihen [eigene Anmerkung: zwischen unabhängigen Variablen und Prognosegegenstand] niemals mit statistischen Mitteln aufgedeckt oder gar bewiesen werden kann, sondern nur mit Hilfe einer fachwissenschaftlichen Theorie"* (Hansmann, 1983, S. 123). Neuere Erkenntnisse deuten jedoch darauf hin, dass die kausale Abhängigkeit zwischen zwei Werten durchaus nachweisbar ist. Der über kaggle.com ausgegebene *causality challenge* liefert einen guten Überblick und eine Reihe an Algorithmen, die bereits entwickelt wurden, um Kausalbeziehungen nachzuweisen. Eine Einführung mit weiterführenden Links befindet sich auf: `https://www.kaggle.com/c/cause-effect-pairs`

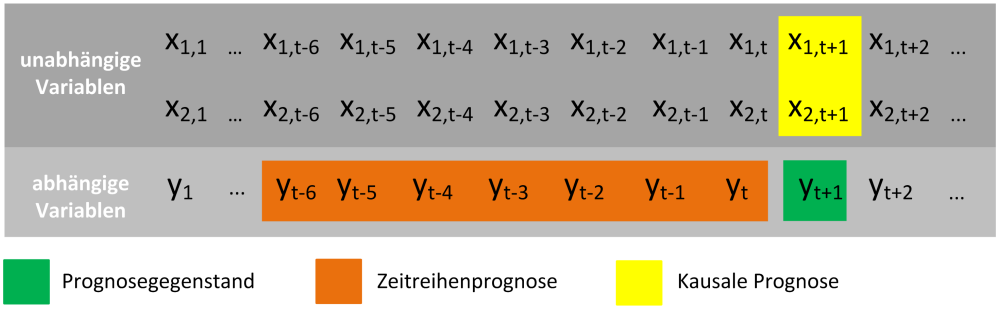


Abbildung 5.1.: Verwendung verschiedener Variablen zur Prognose eines Wertes y_{t+h} (dargestellt mit $h = 1$) bei Zeitreihen- und kausalen Prognosen

- x_N: durchschnittliche Niederschlagsmenge in den letzten n Tagen
- x_R+: durchschnittliche prozentuale Regenwahrscheinlichkeit der nächsten m Tage
- x_M: Monat
- x_A: Absatz $t-1$

Eine Funktion zur Ermittlung einer Absatzprognose für Regenschirme könnte somit folgendermaßen aussehen:

$$AP_{Reg(t+1)} = f(x_R, x_N, x_R+, x_M, x_A) \tag{5.3}$$

Eine mögliche Ausprägung einer Einflussvariable x_i kann durchaus auch eine Zeitreihe sein, sollte ein Zusammenhang zwischen $*y$ und der Zeitreihe bestehen. Im Rahmen einer kausalen Prognose besteht diese Zeitreihe x_i aber nicht aus historischen Realisationen von y, sondern aus historischen Realisationen unabhängiger Variablen. Im Rahmen des Beispiels einer Absatzprognose für Regenschirme könnte beispielsweise die Variable x_N, statt als Mittelwert, auch als Zeitreihe modelliert werden.

5.2.3. Vergleich von Zeitreihen- und kausaler Modellierung

In Abbildung 5.1 (Seite 14) ist der Unterschied zwischen Zeitreihen- und kausalen Prognosen graphisch aufbereitet. Kausale Prognosen werden anhand von Werten erstellt, die unabhängig vom Prognosegegenstand sind und für den Prognosezeitpunkt existieren: x_i zum Zeitpunkt $t+1$. Die Zeitreihenprognose hingegen bedient sich vergangener Werte des Prognosegegenstands: $y_t ... y_{t-n-1}$. Die Funktionseingaben beider Modellierungen bilden disjunkte Mengen.

In welchen Fällen sollte nun ein Verfahren zur Zeitreihenprognose und wann eines zur kausalen Prognose gewählt werden? Die beiden Beispiele zur Erstellung von Absatzprognosen für

Gitarren und Regenschirme, einmal als Zeitreihenmodellierung und einmal als kausale Modellierung, haben die Vor- und Nachteile der beiden Verfahrensklassen bereits in Ansätzen aufgezeigt.

Kausale Prognosen erscheinen auf den ersten Blick nachvollziehbarer als Zeitreihenprognosen. Durch die Modellierung von Einflussvariablen, die den Gegenstand der Prognose beeinflussen, stehen dem Prognoseverfahren wichtige Informationen zur Verfügung, die einem Prognoseverfahren zur Zeitreihenmodellierung fehlen. Doch was passiert, wenn die zeitliche Entwicklung des Prognosegegenstandes Muster enthält, die zur Extrapolation zukünftiger Werte unerlässlich sind? Oder wenn die Einflussfaktoren nicht hinreichend bekannt sind? Versucht man beispielsweise die Absatzprognose für Gitarren als kausale Prognose zu modellieren, wird es schwierig genügend Einflussvariablen zu finden, die den zukünftigen Absatz ausreichend gut beschreiben.

Zeitreihenprognosen hingegen können modelliert werden, sobald genügend historische Daten der abhängigen Variable zur Verfügung stehen, ohne dass es äußere Einflüsse auf den zu prognostizierenden Wert geben muss bzw. ohne dass diese bekannt sein müssen. Dies kann aber gleichzeitig der Nachteil von Zeitreihenprognosen sein. Stellt man sich beispielsweise vor, die Absatzprognose für Regenschirme würde als Zeitreihenprognose modelliert, dann würde man schnell feststellen, dass die Hauptfaktoren für eine Kaufentscheidung nicht berücksichtigt wurden und die Muster in den historischen Absatzzahlen nicht ausreichen werden, um eine zuverlässige Absatzprognose zu ermitteln. Hyndman und Athanasopoulos liefern jedoch gute Gründe, weshalb Zeitreihenprognosen den kausalen Prognosen oft vorzuziehen sind: *"First, the system may not be understood, and even if it was understood it may be extremely difficult to measure the relationships assumed to govern its behavior. Second, it is necessary to know or forecast the various predictors in order to be able to forecast the variable of interest, and this may be too difficult. Third, the main concern may be only to predict what will happen and not to know why it happens. Finally, the time series model may give more accurate forecasts than an explanatory or mixed model"* (Hyndman und Athanasopoulos, 2012, Ch. 1/4).

5.2.4. Kombinierte Prognose

Das im vorangegangenen Zitat von Hyndman und Athanasopoulos erwähnte *mixed model* ist ein Ansatz, der es ermöglicht sowohl zeitliche als auch kausale Einflussfaktoren in die Prognose einfließen zu lassen. Zusätzlich zur Zeitreihe der abhängigen Variablen werden auch unabhängige Variablen in den Prognoseprozess einbezogen (Hyndman und Athanasopoulos, 2012). Im Idealfall werden dadurch die Vorteile zeitreihenanalytischer und kausalanalytischer Modellierungsformen vereint. Die Absatzprognose von Regenschirmen könnte beispielsweise davon profitieren, dass sie zusätzlich zu den Einflussfaktoren die Absatzzahlen der letzten n Verkaufstage kennt, in der Annahme, dass das Kaufverhalten der vorangegangen Tage Einfluss auf das Kaufverhalten der Zukunft hat. Bei sehr vielen Verkäufen könnte der Bedarf beispielsweise gesättigt sein, sehr wenige Verkäufe könnten hingegen Einfluss auf die Gewichtung der Einflussfaktoren haben. AP_{Reg} könnte also in Abhängigkeit zu ihrem historischen Absatz *und*

den kausalen Einflussfaktoren modelliert werden. In Anlehnung an die Notation in den Funktionsbeschreibungen 5.1 und 5.3, wären bei kausalen Prognosen u.a. folgende funktionale Abbildungen möglich:

Die Vereinigungsmenge der bisher gezeigten Variablen

$$*y_{t+1} = f(\, x_{1,t+1}, x_{2,t+1}, y_t, y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, y_{t-6})$$

Die Vereinigungsmenge der bisher gezeigten Variablen,
plus der historischen Realisierungen der Einflussvariablen

$$*y_{t+1} = f(\, x_{1,t+1}, x_{2,t+1}, y_t, y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, y_{t-6}, x_{1,t}, x_{1,t-1}, x_{1+,t-2}, x_{1,t-3}, x_{1,t-4}, x_{1,t-5}, x_{1,t-6}, x_{2,t}, x_{2,t-1}, x_{2+,t-2}, x_{2,t-3}, x_{2,t-4}, x_{2,t-5}, x_{2,t-6})$$

Abbildung 5.2 (Seite 16) veranschaulicht die involvierten Variablen anhand der bereits bekannten graphischen Darstellungsweise.

Abbildung 5.2.: Verwendung verschiedener Variablen zur Prognose eines Wertes y_{t+h} (dargestellt mit $h = 1$) bei kombinierten Prognosen

Sowohl der Umgang mit gemischten Modellen, als auch die Erwähnung dieser Modelle ist in der Literatur sehr unterschiedlich und lässt noch nicht einmal eine konsistente Namensgebung erkennen. In Teilen der Prognoseliteratur finden sie keine Erwähnung (Hansmann, 1983), in anderen Werken werden sie ganz selbstverständlich verwendet (Thiesing, 1996), ohne jedoch

einen theoretischen Hintergrund zu liefern und wieder andere widmen ihnen ganze Bücher (Pankratz, 1991)[8]. Die folgende Tabelle fasst die von mir in der Literatur gefundenen Bezeichnungen zusammen (die jeweiligen Referenzen befinden sich in der Fußnote[9]):

	Modellierungsart		
Zeitreihe	**Kausal**	**Kombiniert**	**in**
Zeitreihe	Regression	Dynamische Regression	A
Horizontal	Vertikal	Horizontal und Vertikal	B
Endogene Datenreihen	Exogene Datenreihen	Endogene und exogene Datenreihen	C
Univariate Zeitreihe	-	Multivariate Zeitreihe	D
Univariat	Multivariat	-	E

Dynamische Regression ist die am häufigsten verwendete Bezeichnung für Modellierungen, die sowohl auf abhängige als auch auf unabhängige Variablen zugreifen. Sie wird in den Standardwerken der Prognoseliteratur benutzt. Jedoch impliziert die Namensgebung, dass ein Regressionsverfahren zur Berechnung der Prognose verwendet wird, was in den referenzierten Werken auch der Fall ist. KNN hingegen arbeiten immer gleich, unabhängig davon welche Modellierung der Eingabewerte ihnen zugrunde liegt, weshalb ich diese Bezeichnung im Zusammenhang mit KNN für irreführend halte.

Ich werde deshalb im weiteren Verlauf dieser Arbeit die Bezeichnung *kombinierte Prognose* benutzen, wenn die Modellierung auf unabhängige und abhängige Variablen zurückgreift, da die anderen genannten Bezeichnungen entweder eine bestimmte Darstellungsform der Modellierung voraussetzen (horizontal/vertikal) oder die kombinierte Prognose als eine Form der Zeitreihenprognose betrachten, was m.E. der Definition von Zeitreihen widerspricht.

Die Uneinigkeit in der Literatur lässt erahnen, dass kombinierte Prognosemodelle in der Vorhersage nicht zum elementaren Handwerkszeug gehören. *"Aufgrund der Komplexität dieser Modelle werden selbst einfachste Ansätze nur von wenigen Innovationsführern des Handels zur Prognose eingesetzt. [...] Sie stellen hohe Anforderungen an Experten in der manuellen Modellierung [...] und sind auf Grund ihres hohen Ressourcenbedarfs nur bedingt auf Massendaten wie etwa im Handel anwendbar"* (Crone, 2010, S. 126). Wie jedoch in Abschnitt 6 und vor allem im praktische Teil dieser Arbeit noch gezeigt werden wird, stellt diese Modellierungsform für KNN kein Hürde dar. Lediglich der Ressourcenbedarf wächst mit der Anzahl der involvierten Variablen und diese ist im Allgemeinen bei einer kombinierten Prognose größer als bei kausalen oder Zeitreihenmodellierungen.

[8] Die 3 hier referenzierten Werke stehen nicht alleine, sondern lediglich stellvertretend für den unterschiedlichen Umgang mit kombinierten Prognosemodellierungen.

[9] A: (Hyndman und Athanasopoulos, 2012), (Makridakis u. a., 1998), (Pankratz, 1991); B: (Haar, 2011); C: (Eisenbach, 2005); D: (Horváth, 2003); E: (Hansmann, 1983)

5.2.5. Annahmen und Notation zur Modellierung von Prognosen

Die drei dargestellten Modellierungsformen werden im Verlauf dieser Arbeit erweitert und konkretisiert. Folgende grundlegenden Annahmen und Notationen werden im weiteren Verlauf dieser Arbeit vorausgesetzt.

Annahmen:

- Alle behandelten Prognosen sind Tagesprognosen und alle abhängigen und unabhängigen Variablen liegen als Tageswerten vor.
- Der Prognosehorizont **h** ist stets 1, es werden also ausschließlich Prognosen für den Folgetag erstellt.
- Die Prognosen werden in einem nächtlichen Batch für den Folgetag erstellt. Dies bedeutet, dass abhängige und unabhängige Variablen für Tag t bereits vorliegen, so dass alle Werte $\leqq t$ in die Prognose einfließen können.

Notation:

- **$y_{i,t}$** bezeichnet eine abhängige Variable y_i[10] zum Zeitpunkt t
- **$x_{i,t}$** bezeichnet eine unabhängige Variable x_i zum Zeitpunkt t
- **n** steht für die Größe des historischen Zeitfensters - also die Anzahl an historischen Tagen, für die *abhängige* Variablen y in den Input-Vektor einfließen. Wenn beispielsweise die abhängigen Werte der letzten 7 Tage prognoserelevant sind, dann ist $n = 7$, und die historischen Realisationen von y die in die Prognose einfließen sind:
  $y_t, y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, y_{t-6}$.
- **m** steht für die Anzahl an Tagen, für die *unabhängige* Variablen x_i in den Input-Vektor einfließen.
  - Im Falle einer kausalen Prognose ist $m = 1$ und bezieht sich stets auf die Ausprägung von x_i zum Prognosezeitpunkt, also $x_{i,t+1}$.
  - Im Falle einer kombinierten Prognose können auch historische Realisationen von unabhängigen Variablen in die Prognose einfließen. $m = n+1$ definiert somit den selben historischen Zeitraum wie n und fügt diesem noch die Ausprägung von x_i für den Prognosezeitpunkt hinzu.

Tabelle 5.2 bietet eine Zusammenfassung der Notation, die in den folgenden Kapiteln als Grundlage zur Modellierung von Zeitreihen-, kausalen und kombinierten Prognosen verwendet wird:

[10] Für den Fall, dass nur eine abhängige Variable existiert, wird auf den Index i verzichtet.

Modellierung	Variablen	n	m	Anmerkung
Zeitreihe	abhängig, Bezeichner: y_i	$n \geq 1$	$m = 0$	Die jüngste Realisierung von y ist y_t.
Kausal	unabhängig, Bezeichner: x_i	$n = 0$	$m = 1$	Die jüngste Realisierung von x_i ist $x_{i,t+h}$ was unter den getroffenen Annahmen immer $x_{i,t+1}$ ist.
Kombiniert	abhängig und unabhängig	$n \geq 1$	$m \geq 1$	Es gilt immer $n \geq 1 \wedge m \geq 1$, da es sich sonst um eine Zeitreihen- oder kausale Prognose handeln würde.

Tabelle 5.2.: Notation zur Modellierung von Zeitreihen-, kausalen und kombinierten Prognosen

5.2.6. Überblick über Klassen von datenzentrierten Prognoseverfahren

Zur Lösung von Zeitreihen-, kausalen und kombinierten Prognosemodellen wurden eine Vielzahl konkurrierender Prognoseverfahren entwickelt *"Dabei kann beispielsweise ein einfaches, zeitreihenanalytisches Erklärungsmodell [...] sowohl durch ein Naives Verfahren, einen gleitenden Mittelwert unterschiedlicher Länge und Gewichtung, Verfahren der Exponentiellen Glättung, ARIMA-Verfahren, Schätzung mit unterschiedlichen Verteilungsfunktionen oder Verfahren der künstlichen Intelligenz einschließlich der Neuronalen Netze berechnet werden"* (Crone, 2010, S. 88).

Abbildung 5.3 (Seite 20) bietet eine Übersicht über datenbasierte Verfahrensklassen, die von Sven F. Crone in (Crone, 2010) im Rahmen einer sehr umfangreichen Literaturrecherche zusammengetragen wurden. Es handelt sich hierbei um Verfahrensklassen, die fast beliebig tief strukturiert sind und in entsprechend vielen Ausprägungen existieren. Es würde den Rahmen dieser Arbeit sprengen, all diese Modelle vorzustellen. Als weitere Lektüre seien die Seiten 111-127 in (Crone, 2010) oder das (nicht so kurze) *Kurzlehrbuch Prognoseverfahren* von K-W. Hansmann (Hansmann, 1983) empfohlen.

Das gemeinsame Ziel all dieser Verfahren ist die - mit f bezeichnete - Funktion zwischen einer Menge an Eingabewerten I und der Ausgabe O zu ermitteln. Prognoseverfahren liefern Regeln, Algorithmen und Rechenvorschriften zum Kombinieren von I zur Abbildung auf O. Die bisherige Implementierung der produktiven Softwarelösung zur Errechnung von Backmengenempfehlungen gehört zu den Mittelwertbildenden Verfahren der Zeitreihenprognose (siehe zur Erinnerung Abbildung 1.1, Seite 3).

Es ist schwierig eine eindeutige namentliche Trennung zwischen KNN und den anderen dargestellten Prognoseverfahren zu finden. Deshalb wird im Folgenden die informelle Bezeichnung *herkömmlich* oder *klassisch* verwendet, statt über die Namensgebung eine eindeutige Trennung zu suggerieren, die es nicht gibt. Mit *herkömmlich* oder *klassisch* werden also im Weiteren alle Prognoseverfahren bezeichnet, die als etablierte Konkurrenz zu KNN zu sehen sind.

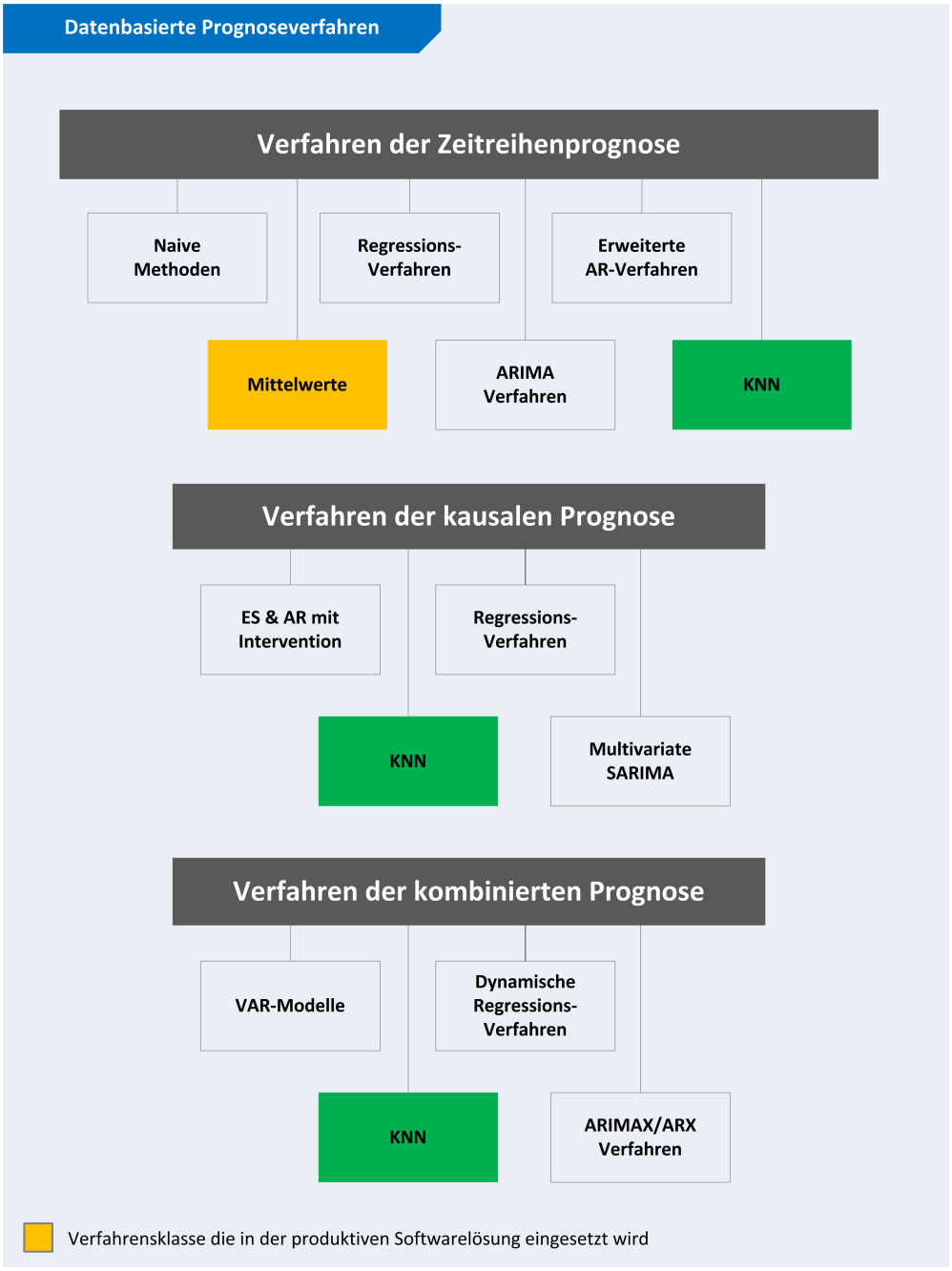


Abbildung 5.3.: Überblick über die verschiedenen Prognosemodelle

6. Beurteilung von Künstlichen Neuronalen Netzen als Prognoseverfahren im Handel

Wie in Kapitel 2 erwähnt, ist diese Arbeit dadurch motiviert, überprüfen zu wollen, ob KNN in der Lage sind gute Bedarfsprognosen für Backwaren im Einzelhandel liefern zu können. Die Motivation dem bisherigen Verfahren eine Prognose durch KNN gegenüber zu stellen wäre aber sicherlich fehlgeleitet, wenn sich der Einsatz von KNN nicht durch wissenschaftliche Erkenntnisse stützen ließe. Eine Entscheidungsgrundlage für die Wahl von KNN soll in diesem Kapitel gelegt werden.

Die Akzeptanz der klassischen Prognoseverfahren ist sehr groß. *"[M]ost of them have solid theoretical foundations and proof of competence"* (Masters, 1993, S. 48). Die Abbildungen 6.1 (Seite 22) und 6.2 (Seite 22) zeigen die Häufigkeit der Anwendung verschiedener Prognoseverfahren in der Praxis. Die Zahlen stammen aus einer der wenigen Studien die es in diesem Bereich gibt. Sie wurde von C.L. Jain in den Jahren 2001 und 2002 durch Befragungen bei Konferenzen des IBF (Institute of Business Forecasting & Planing) erstellt[1] und in (Jain, 2005) veröffentlicht. Die graphische Aufbereitung ist (Crone, 2010) entnommen.

Abbildung 6.1 (Seite 22) spiegelt den Einsatz von Prognoseverfahren in unterschiedlichen Industriezweigen wider, wohingegen Abbildung 6.2 (Seite 22) konkret den Einsatz von Prognoseverfahren in Unternehmen des Handels wiedergibt. Unter Betrachtung aller Industriezweige dominiert die Anwendung datenbasierter Verfahren in 91% aller Betriebe, während urteilsbasierte Verfahren nur in 6% der Betriebe angewendet werden. Aus der Befragung der Unternehmen des Handels ergibt sich ein gegensätzliches Bild, in dem urteilsbasierte Verfahren mit 73% die datenzentrierten Verfahren klar verdrängen. Aus den Antworten beider Befragungsgruppen geht jedoch hervor, dass KNN in der Praxis der befragten Unternehmen kaum eine Rolle spielen.

Auf den ersten Blick mögen diese Ergebnisse ernüchternd erscheinen und suggerieren, dass KNN im Bereich der Prognose nicht sinnvoll einsetzbar sind. Viele Faktoren deuten jedoch darauf hin, dass sich KNN, vor allem im Handel, sehr gut für Prognosen eignen. Es gibt deutliche Indikatoren dafür, dass die präsentierten Zahlen eine Momentaufnahme des Einsatzes von KNN in bestimmten US Amerikanischen Unternehmen zur Jahrtausendwende widerspiegeln und keine Aussage über die grundlegende Tauglichkeit von KNN für Prognosen des Handels enthal-

[1] Die Befragung umfasst 1072 Antworten von Absatzplanern aus 13 Industriezweigen, durch Kombination der Befragung von 379 Absatzplanern bei fünf Konferenzen der IBF im Jahr 2001 und von 693 Absatzplanern in 19 Industriezweigen bei 5 IBF-Konferenzen in 2002 (Crone, 2010).

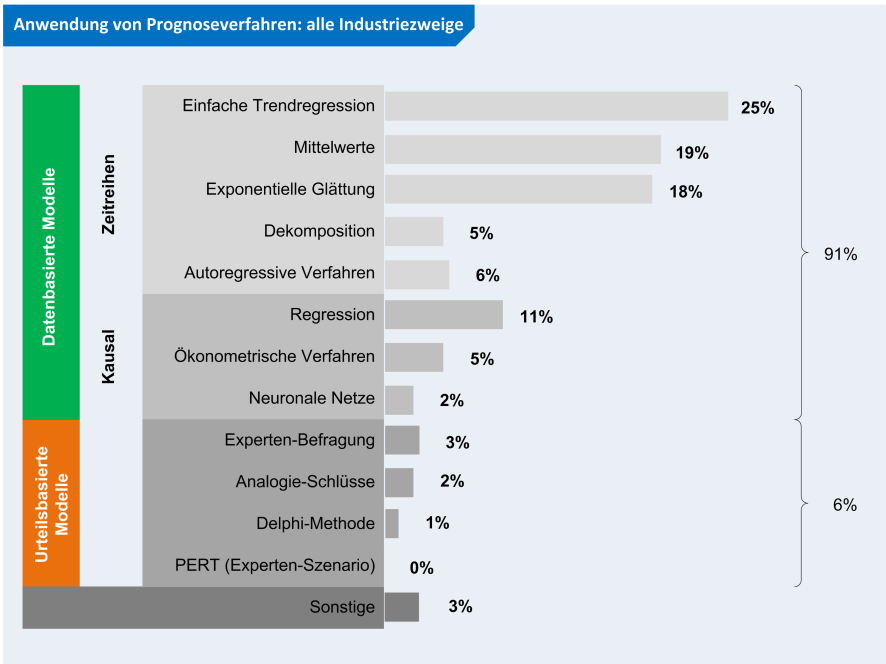


Abbildung 6.1.: Anwendung von Prognoseverfahren in der betrieblichen Praxis

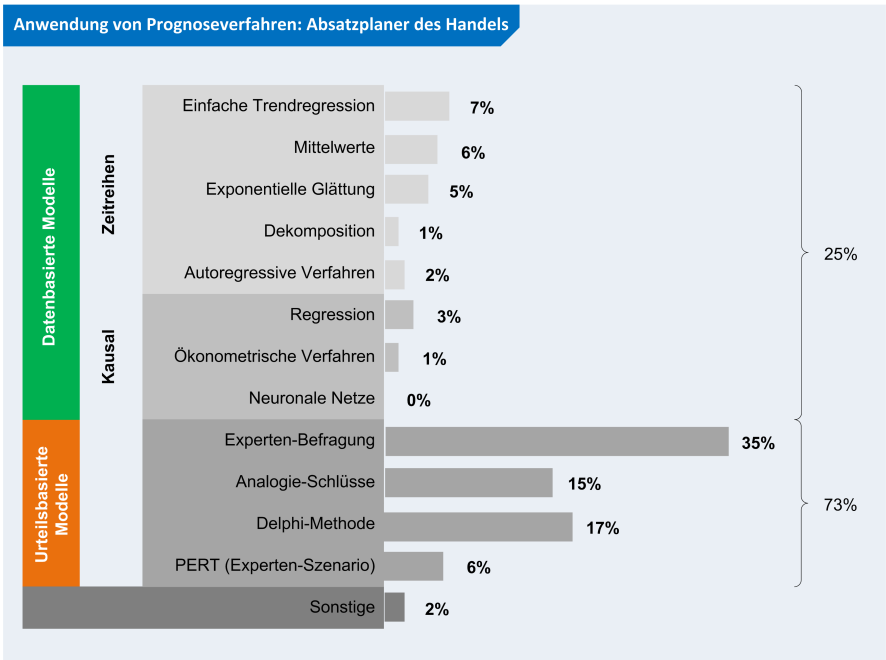


Abbildung 6.2.: Anwendung von Prognoseverfahren im Handel

ten. Es folgt eine kurze Zusammenfassung der Fakten, die andeuten, dass die Ergebnisse der präsentierten Studie einerseits eine nicht so deutliche Sprache sprechen wie es den Anschein haben mag und dass sie andererseits Hinweise auf die Entfaltungsmöglichkeiten von KNN in Prognosen des Handels liefern.

- Von den 1072 Befragten stammten lediglich 10 im Jahr 2001 und 35 im Jahr 2002 aus Betrieben des Handels (Crone, 2010). Die Ergebnisse können also nicht überbewertet werden, sind jedoch, in Ermangelung anderer, umfangreicherer, Studien, ein Indiz für die Akzeptanz des Einsatzes von KNN im Bereich der Handelsprognosen.
- Die Tatsache, dass die meisten befragten Unternehmen des Handels urteilsbasierte Verfahren einsetzen, deutet darauf hin, dass das Feld der datenbasierten Verfahren vielen Unternehmen nicht zugänglich ist. Bekannt ist jedoch, dass urteilsbasierte Verfahren *"[...] z. B. durch Expertenurteile oder strukturierte Ansätze von Delphi aufgrund der Einbindung von Experten zeitaufwendig, kostenintensiv und nicht zu automatisieren sind"* (Crone, 2010, S. 92). Ohne hier über die Ursachen des geringen Einsatzes datenbasierter Verfahren spekulieren zu können, bleibt doch festzuhalten, dass das Potenzial der datenbasierten Prognose im Handel mit 25% sicherlich noch nicht ausgeschöpft ist. Oder wie es D. Vonko in (Vonko, 2009, o.S.) beschreibt: *"What's really surprising, however, is the fact that a huge number of those who could benefit richly from neural network technology have never even heard of it, take it for a lofty scientific idea or think of it as of a slick marketing gimmick. "*
- Mögliche Gründe für die zögerliche Akzeptanz von KNN in der Praxis, wo zeitliche und finanzielle Einschränkungen eine feste Größe sind und der Druck verwertbare Ergebnisse liefern zu müssen groß ist, könnten folgende Faktoren sein:
  - Wie Kapitel 7 noch ausführlich zeigen wird, sind die Konfigurationsmöglichkeiten von KNN sehr vielfältig und über viele der entscheidenden Stellschrauben gibt es keine Einigkeit und kaum solide Vorgaben für die Praxis. Viele Fragen können erst durch Anwendung auf die jeweilige Problemstellung und umfangreiche Experimente beantwortet werden. (Crone und Kourentzes, 2009)
  - Gleichzeitig ist eine 'gute' oder 'richtigen' Konfiguration des KNN entscheidend für die Prognosequalität. (Masters, 1993)
  - KNN haben in wissenschaftlichen Studien und in den, im Folgenden noch erwähnten, Wettbewerben mit sehr zweideutigen Resultaten abschnitten. *"[ANN] demonstrated substantial variability in their predictive accuracy, ranging from some of the top contenders of the competition to some of the most inaccurate predictions which failed to outperform even naive statistical benchmark algorithms"* (Crone und Kourentzes, 2010).

- Wissenschaftliche Studien und Wettbewerbe[2] in denen verschiedene Prognoseverfahren gegeneinander antreten, zeigen allerdings, dass das Potenzial KNN enorm ist und bei richtiger Konfiguration mit den spezialisiertesten Verfahren mithalten können oder sie gar übertreffen (Palit und Popović, 2005) (Stepnicka u. a., 2013) (Crone u. a., 2011).
- Das Potenzial oder zumindest das Interesse an KNN scheinen durch Crone u. a. (2011) bestätigt zu werden, die ermittelt haben, dass in den letzten 20 Jahren über 5000 Publikationen in Akademischen Zeitschriften und Tagungsschriften zu Prognosen mit KNN, veröffentlicht wurden. Fildes et al. haben ähnliche Erkenntnisse veröffentlicht: *"while the last 25 years have seen rapid developments in forecasting across a broad range of topics, computer intensive methods such as NNs contribute the single largest area of publications in Operational Research (2008) and one of the top 4 in forecasting journals (2006)"* (Fildes u. a., 2008, S. 1151).
- Weder über praxisbezogene Studien noch wissenschaftliche Erkenntnisse oder Prognosewettbewerbe konnte bisher Einblick in die Erkenntnisse großer Softwarehäuser im Bereich KNN und KI im Allgemeinen (Siemens, Alyuda, NeuroDimensions, SAS etc.) oder Entwickler von Prognosesoftware (SAP, Oracle, John Galt, Smart etc.) erlangt werden. Zu vermuten ist allerdings, dass diese Erkenntnisse dazu beitragen könnten, den Einsatz von KNN im Bereich der Prognose zu professionalisieren. (Crone u. a., 2011)

KNN haben sich zu einem Standard in Datamining- und Klassifizierungs-Anwendungen entwickelt (Stepnicka u. a., 2013) wovon sie im Bereich der Handelsprognosen noch weit entfernt sind. Die folgenden 2 Abschnitte sollen jedoch einen Einblick in ihr Potenzial, auf diesem Gebiet, liefern und gleichzeitig auf die Hürden eingehen, die bekannt sein müssen, wenn man mit KNN arbeiten möchte.

6.1. Nachteile Künstlicher Neuronaler Netze

Einer der größten Nachteile KNN ist bereits angesprochen worden: die fehlende Eindeutigkeit bei der Wahl der Konfigurationsmöglichkeiten. Gleichzeitig ist der Erfolg oder Misserfolg beim Einsatz KNN, aber direkt von der richtigen bzw. falschen Konfiguration anhängig, was das Fehlen zuverlässiger und verwertbarer Richtwerte umso schwerwiegender macht. Die Herausfor-

[2] Einer der ersten Wettbewerbe zur Vorhersage (univariater) Zeitreihen wurde von Makridakis & Wheelwright ins Leben gerufen. In den Jahren 1982, 1993 und 1998 wurden die sogenannten M-Competitions durchgeführt, die das Abschneiden diverser Prognoseverfahren zur Lösung identischer Problemstellungen verglichen. Makridakis beschreibt die Wettbewerbe wie folgt: *"The M-Competitions are such empirical studies that have compared the performance of a large number of major time series methods using recognized experts who provide the forecasts for their method of expertise. Once the forecasts from each expert have been obtained they are evaluated and compared with those of the other experts as well as with some simple methods used as benchmarks"* (Makridakis und Hibon, 2000). Die M3-Competition war die erste an der ein Experte ein KNN gegen die etablierten Verfahren antreten ließ, mit mäßigem Erfolg. In den Jahren seitdem sind viele weitere Wettbewerbe dieser Art ausgerufen worden, die Teilweise auf den selben Daten arbeiten, die Makaridis et al. zur Verfügung gestellt haben. dazu gehören: NNGC (Neural Network Grand Challenge), NN3, NN5 und ESTSP (European Symposium on Time Series Prediction).

derung liegt also nicht nur bei der Modellierung der Problemstellung wie bei den klassischen Prognoseverfahren, sondern mindestens in gleichem Maße bei der Ermittlung der richtigen Konfiguration. In Kapitel 7 wird gezielt auf die einzelnen Konfigurationsschrauben und ihre Auswirkungen eingegangen.

Ein weiterer Nachteil kann die fehlende 'Erklärungsfähigkeit' KNN sein. Es ist meist schwierig zu Erklären, warum eine KNN eine gegeben Ausgabe produziert. Ergebnisse können bewertet werden, aber es gibt kaum eine Möglichkeit zu erkennen, wie das KNN zu dem Ergebnis gelangt ist (Trippi und Turban, 1996). Dies wird sehr anschaulich von Heaton bestätigt: *"A neural network can be very useful for solving the problem for which it was trained, but cannot explain its reasoning. The neural network knows something because it was trained to know it. However, a neural network cannot explain the series of steps followed to derive the answer"* (Heaton, 2011). Sollte die Lösung eines Problems also einen klar verfolgbaren Weg zum Ziel benötigen, dann sind KNN sicherlich nicht das Werkzeug der Wahl.

Weitere Nachteile KNN, werden im Folgenden zusammengefasst:

- Hohe Entwicklungszeit: Entwicklungszeit bezieht sich einerseits auf die Zeit die es braucht ein KNN zu trainieren, also sowohl auf die Länge des Zeitraums in dem man sich mit dem Training eines KNN beschäftigt als auch auf die Laufzeit eines einzelnen Trainingszyklus bezogen (Trippi und Turban, 1996). Andererseits wird viel Zeit in die Konfiguration des KNN einfließen müssen. *"A [neural network developer] will focus and spend quite a bit of time selecting and governing input items for his or her neural network and adjusting their parameters. He or she will spend from (at least) several weeks - and sometimes up to several months - developing the network"* (Vonko, 2009, o.S.).
- Die große Menge benötigter Trainingsdaten: Wie bereits erwähnt, handelt es sich bei KNN um Daten-getriebene Verfahren und salopp ausgedrückt bedeutet dies, dass man Daten braucht - je mehr desto besser. Bei nicht vorhandenen, schwer zu beschaffenden, lückenhaften oder nur in geringem Maße existierenden Daten können KNN daher nicht sinnvoll eingesetzt werden (Trippi und Turban, 1996).

6.2. Vorteile Künstlicher Neuronaler Netze

Die genannten Nachteile werden m.E. durch eine Vielzahl an Vorteilen aufgewogen die eine Verfolgung dieses Ansatzes im Rahmen der Handelsprognosen eindeutig rechtfertigen. Vor allem im Vergleich zu herkömmlichen Prognoseverfahren weisen KNN Vorteile auf die im Folgenden aufgezeigt werden sollen.

Die größte Schwächer der meisten herkömmlichen Prognoseverfahren ist ihre restriktive Annahme hinsichtlich der zulässigen Modellform und Verteilungsform der Variablen. Alle in Abbildung 5.3 (Seite 20) vorgestellten Verfahren sind a priori an eine der folgenden Annahmen gebunden:

- die Prognosegegenstand ist funktional anhängig von:
  - unabhängigen Variablen (kausale Prognose)
  - abhängigen Variablen (Zeitreihenprognose)
  - unabhängigen und abhängigen Variablen (kombinierte Prognose)
- der zu approximierende funktionale Zusammenhang ist:
  - linear
  - nicht linear (also die Fülle aller Zusammenhänge die nicht linear sind)

  (Zhang u. a., 1998)

Abhängig von diesen Entscheidungen ist bei vielen dieser Verfahren eine aufwendige Vorverarbeitung der Daten notwendig. KNN zeigen hingegen die Fähigkeit jegliche funktionale Formen anzunähern ohne a priori Spezifikation der funktionalen Abhängigkeiten zwischen unabhängigen und abhängigen Variablen (Zhang u. a., 1998). Masters drückt diesen Vorteil in seiner unterhaltsamen Schreibweise folgendermaßen aus: *"They [ANN] are so versatile that they do not require us to choose a model. Rather than having to choose between AR versus MA versus ARIMA versus exponential smoothing versus spectral decomposition versus ... ad nauseam, we can just throw a mountain of data at the network and let it ruminate. It will choose its own model and generally do a good job of it"* (Masters, 1993, S. 48).

Hinzu kommt, dass herkömmliche Verfahren schlecht mit chaotischen oder sogenannten verrauschten Einflüssen in den Daten umgehen können. Mit chaotisch sind zum einen schwer vorhersehbare 'Ausreißer' nach unten oder oben gemeint, die beispielsweise durch Werbemaßnahmen, plötzliche Wetterereignisse oder auch nur durch eine Baustelle vor dem Supermarkt ausgelöst werden können. Zum anderen ist nicht auszuschließen, dass in den historischen Daten Meß- oder Tippfehler enthalten sind. Unter verrauschten Einflüssen oder Daten (engl. *noisy data*)[3] versteht man Informationen, die dem KNN präsentiert werden, die aber in keinem direkten Zusammenhang zum Prognosegegenstand stehen. KNN sind dafür bekannt, verrauschte Daten deutlich besser tolerieren zu können als die meisten andere Verfahren. Und da Daten, die Handelsprognosen zugrunde liegen, meist über nicht unerhebliche chaotische und verrauschte Komponenten verfügen, ist dies als deutlicher Vorteil gegenüber herkömmlichen Verfahren zu werten (Masters, 1993).

Zusammenfassend können vier positive Eigenschaften von KNN hervorgehoben werden (nach Zhang u. a. (1998)):

1. Im Gegensatz zu herkömmlichen Model-basierten Verfahren brauchen KNN, als Datengetriebene selbst-anpassende Verfahren, wenige a priori Annahmen über Modell und Struktur der Abhängigkeiten. Ein Vorteil der besonders dann zum Tragen kommt, wenn die zugrundeliegende Beziehung zwischen Prognosegegenstand und beeinflussenden

[3] (Zhang u. a., 1998), (Masters, 1993) und viele andere

Faktoren schwer zu spezifizieren ist oder schlichtweg nicht bekannt ist. Dies ist ein bedeutender Vorteil, da KNN auch dann valide Prognosen erstellen können, wenn den Experten keine formalen Erkenntnisse über die zugrundeliegende Modellform vorliegen (Crone, 2010). Im Gegensatz dazu, wenn das Problem grundlegend verstanden wurde, wird die Prognosequalität von KNN die herkömmlicher Verfahren kaum übertreffen aber durchaus annähern können (Lippe, 2006).

2. KNN sind in der Lage zu Generalisieren. Sie lernen anhand von Beispielen und können daraus auf ungesehene Daten schließen, selbst wenn die Beispieldaten verrausche Daten enthalten. Prognosen funktionieren genau so, es muss von bekannten (historischen) Daten auf unbekannte (zukünftige) Daten geschlossen werden, so dass Prognosen, zumindest theoretische, ein ideales Anwendungsgebiet für KNN sein sollten.

3. KNN sind universelle funktionale Approximatoren (engl. *universal functional approximators*) und können im Gegensatz zu den herkömmlichen statistischen Verfahren jede beliebige Funktion annähern. Jede Form der Vorhersage geht davon aus, dass es eine (bekannte oder unbekannte) Beziehung zwischen den abhängigen oder unabhängigen Variablen und dem Prognosegegenstand gibt, herkömmliche Verfahren sind jedoch in der Annäherung sehr komplexer Funktionen eingeschränkt. In der Annahme, dass Handelsprognosen oft eine komplexe Funktion zugrunde liegt, könnten KNN eine gute Alternative sein.

4. KNN können nicht lineare Zusammenhänge abbilden. Das Prognostizieren zukünftiger Werte war lange Zeit die Domäne linearer statistischer Verfahren[4]. Diese Verfahren gehen grundsätzlich davon aus, dass die analysierten Daten von linearen Prozessen erzeugt werden und somit durch lineare Prozessmodelle prognostiziert werden können (Crone, 2010). Während lineare Modelle diverse verfahrenstechnische Vorzüge aufweisen (höhere Transparenz, vereinfachte Analyse, bessere Erklärbarkeit und leichtere Anwendung), sind sie zur Anwendung in nicht linearen Problemstellungen ungeeignet. Da viel Systeme und Prozesse der betrieblichen Realität häufig nicht linear sind, wurde eine Reihe von Prognoseverfahren[5] entwickelt, die nicht lineare Interaktionen der Variablen abbilden können. *"Allerdings erfordern diese parametrischen Ansätze der nichtlinearen Verfahren die a priori Spezifikation der expliziten Modellform des nichtlinearen Erklärungsmodells für eine Zeitreihe von Beobachtungen, vielfach ohne Kenntnis der zugrunde liegenden Gesetzmäßigkeiten und bei einer Vielzahl potenziell anwendbarer Modelle"* (Crone, 2010, S. 209). Wie bereits erwähnt, gehören KNN zu den *Daten*-getriebene nicht linearen Verfahrensklassen, im Gegensatz zu den gerade erwähnten *Modell*-basierten nicht linearen Verfahrensklassen. KNN sind also in der Lage nicht lineare Problemstellungen zu modellieren, ohne a priori Wissen über die Beziehung zwischen Eingabe und Ausgabe haben zu müssen, solange genügend historische Daten der abhängigen und un-

[4] Dazu zählen insbesondere die (univariate) Exponentielle Glättung, die ARIMA-Modelle und die lineare dynamische Regression (Crone, 2010)

[5] Hervorzugeben sind hier nicht-lineare Autoregressive (nAR) oder nicht-lineare Moving-average-(nMA-) Modelle, Bilineare Modelle und Threshold Autoregression (Crone, 2010)

abhängigen Variablen vorliegen. Sie sind somit ein viel flexibleres Prognoseverfahren als viel herkömmliche Verfahren.

Ein weiterer Vorteil gegenüber herkömmlichen Verfahren, den Zang et al. in ihrer Zusammenfassung nicht erwähnen, ist die geringere mathematische Komplexität der KNN. Mertens und Rässler erklären, dass *"[...] die Auswahl und Parametrisierung [herkömmlicher statistischer Verfahren] meist mathematische Experten erfordern, was in der betrieblichen Praxis zu einem Akzeptanzproblem geführt hat. [...] Eine höhere Genauigkeit bei betrieblichen Vorhersagen kann also dadurch erreicht werden, dass Praktiker aufgrund der KI-Unterstützung eher geneigt sind, komplexe Methoden einzusetzen"* (Mertens und Rässler, 2012, S. 4).

Ein letzter Vorteil, der auf die Anpassungsfähigkeit KNN während der Produktivphase eingeht, sollte nicht unerwähnt bleiben. Medesker, Turban und Trippi nennen diese Fähigkeit *Adaptability* und konstatieren: *"Since the network learns in new environments, training can occur continuously over its useful life and occur concurrently with the deployment of the network"* (Medesker u. a., 1996, S. 10). KNN können sich den Veränderungen in der Realität anpassen, indem das Netz immer wieder, automatisiert, mit den neusten Daten trainiert wird und somit die zugrundeliegende Abbildungsfunktion zwischen Ein- und Ausgabe verändert wird, sobald sich die Beziehung zwischen ihnen ändern sollte.

7. Grundlagen Künstlicher Neuronaler Netze

Ziel dieses Kapitels ist es, eine kurze Einführung in die wichtigsten Konzepte KNN zu geben und in Teilen bereits auf die speziellen Anforderungen der hier vorliegenden Problemstellung einzugehen. Dieses Kapitel liefert einen 'Durchstich' durch die Beschreibungsebenen von KNN, jedoch stets geleitet durch die Anforderungen des gegebenen Anwendungsfalls.

Ziel bei der Verwendung eines KNN ist die Modellierung eines üblicherweise nicht linearen Zusammenhangs von Eingabe- und Ausgabewerten. Dieser Zusammenhang wird oft als Muster (engl. *Pattern*) bezeichnet und wird durch die Approximation einer unbekannten und analytisch meist schwer fassbaren Abbildungsvorschrift $f(x) \rightarrow y$ erreicht. Diese Funktion wird durch das KNN in einer Trainingsphase anhand exemplarischer Werte erlernt, so dass es in der Lage ist den Funktionsverlauf so zu generalisieren, dass auch ein nicht trainiertes x auf ein sinnvolles y abgebildet werden kann (Thiesing, 1996).

"Neural networks do not loop, call subroutines, or perform any of the other tasks associated with traditional programming. Neural networks recognize patterns." (Heaton, 2011, S. xxiv) Der Unterschied zur traditionellen Programmierung besteht darin, dass neuronale Lösungen erlernt und nicht programmiert werden. (Callan, 2003)

Konkret ist ein KNN ein Verbund von Einheiten - auch als Neuronen oder Knoten bezeichnet. Diese Einheiten sind einfache Prozessoren, deren Informationsverarbeitungsfähigkeiten sich beschränken auf:

- Regeln zum Kombinieren von Input-Werten
- Aktivierungsregeln zum Berechnen von Output-Werten

Die Output-Werte können über gewichtete Verbindungen an andere Neuronen übertragen werden. Je größer der Absolutbetrag des Gewichtes, desto größer ist der Einfluss eines Neurons auf ein anderes (Rey, 2013). Das Wissen eines KNN ist in seinen Gewichte kodiert (Callan, 2003). Lernen ist bei KNN als Gewichtsveränderungen zwischen den Neuronen definiert.

Jedes KNN besteht aus einer Menge von Neuronen, die zur Aufnahme und Weitergabe von Input-Werten dient und einer Menge von Neuronen, die die Output-Werte zur Verfügung stellen. Im Zusammenhang von KNN spricht man von Schichten, also einer Input-Schicht und einer Output-Schicht[1]). Für die meisten Typen und Architekturen von KNN trifft außerdem zu, dass sie über mindestens eine weitere Verarbeitungsschicht verfügen. Im Falle des in dieser Arbeit

[1]Gebräuchlicher sind zwar die englischen Begriffe, input-layer und output-layer, ich habe mich jedoch für die denglischen Begriffe entschieden, da sich rein englische Begriffe schlecht in einen deutschen Text einfügen.

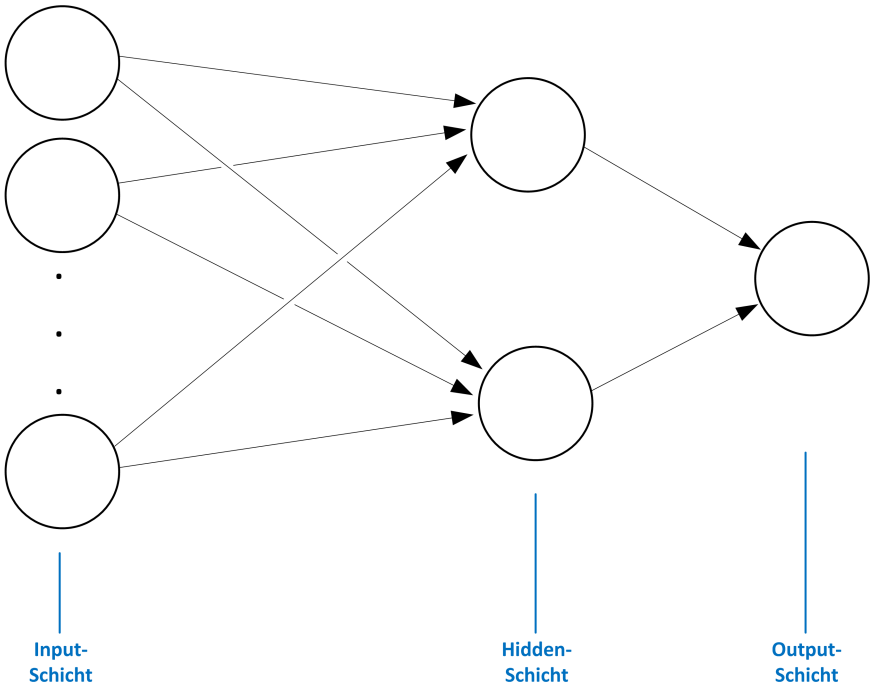


Abbildung 7.1.: Schematische Darstellung eines Künstlichen Neuronalen Netzes (KNN)

benutzen und im Folgenden beschriebenen, Multilayer-Perceptrons (MLP) handelt es sich um beliebig viele Hidden-Schichten, durch die das MLP in der Lage ist, jede beliebige lineare und nicht lineare Funktion anzunähern (Hornik u. a., 1989).

Abbildung 7.1 (Seite 30) zeigt ein allgemeines KNN mit einer Input-, einer Hidden- und einer Output-Schicht mit respektive n, 2 und 1 Neuronen.

Formal lässt sich ein KNN durch die Ausprägungen des Tupels $A = [V, I, L]$ beschreiben, mit A = Netzwerkarchitektur (Alex, 1998)[2] - siehe dazu Tabelle 7.1 (Seite 31).

7.1. Das Feedforward Multilayer-Perceptron (MLP)

Bevor ich damit beginne die verschiedenen Ausprägungen des Tupels $A = [V, I, L]$ vorzustellen, möchte ich die Wahl der grundlegenden Netzarchitektur vorweg nehmen und so bereits die Ausprägungsvielfalt von V, I, L einschränken. Das Feedforward Multilayer-Perceptron (MLP) wird fast einstimmig in der Literatur als Architektur der Wahl für Zeitreihen-, kausalanalytische und/oder kombinierte Prognosen verwendet und empfohlen. Sven Crone sagt dazu in

[2] Übernommen aus der Ausarbeitung in (Crone, 2010) und für die Zwecke dieser Arbeit leicht modifiziert und vereinfacht.

$A = [V, I, L]$	V = Verarbeitungsschritte im Neuron, I = Informationsverarbeitung und Propagierung, L = Parametrisierung des Lernprozesses
mit $V = [F_I, F_A, F_O]$	F_I = Eingabefunktion, F_A = Aktivierungsfunktion, F_O = Ausgabefunktion
mit $I = [T, W, K]$	T = Netzwerktopologie, W = Verbindungs- oder Gewichtsmatrix, K = Aktivierungsstrategie
mit $L = [Z, G, S, O]$	Z = Lernziel, G = Lernalgorithmus, S = Stopkriterium, O = Fehlerfunktion

Tabelle 7.1.: Detailtiere Darstellung des Tupels $A = [V, I, L]$

(Crone, 2010, S. 187): *"Seit der Einführung des MLP konzentriert sich ein Großteil der wissenschaftlichen Forschung und der praktischen Anwendung auf diese Netzwerkarchitektur, auch im Bereich der Prognose."* Timothy Masters argumentiert etwas allgemeiner in (Masters, 1993, S. 116): *"This model [MLP] has a well-deserved reputation for being **the** neural network. It is a universal function approximator. We can, theoreticaly at least, teach anything learnable to this network. Thus, it is reasonable to consider this network for any problem."* Masters bezeichnet das MLP gar als *"standard workhorse"*, was deutlich macht, dass es sich bei einem MLP um eine sehr universale Architektur handelt, die sicherlich nicht alle Problemstellungen gleich gut lösen kann. Doch Vergleichsstudien zwischen MLP und konkurrierenden Architekturen wie FIR und Elman-Netzen in (Koskela u. a., 1996) haben beispielsweise ergeben, dass die Wahl eines geeigneten Lernalgorithmus (G) ein viel wichtigerer Faktor für die Ergebnisgüte ist, als die Wahl der zugrundeliegenden Netzarchitektur. Die Architektur hatte jedoch Auswirkungen auf die Trainingsgeschwindigkeit - und auch hier schnitt das MLP deutlich besser ab[3] als die in der Studie benutzten Vergleichsarchitekturen.

Auch Janetzke und Lewandowski (2012) und Zhang u. a. (1998) konstatieren, dass sich MLPs hervorragen für betriebswirtschaftliche Prognosen eignen.

Das MLP ist eine Weiterentwicklung des Perceptrons, das bereits 1958 von Frank Rosenblatt vorgestellt wurde (Rosenblatt, 1958). Im Gegensatz zu diesem ist es aber ein weitaus mächtigeres Konstrukt. Hornik, Stinchcombe und White postulierten 1989 in ihrem Beitrag in *Neural Networks*, dass *"[...] standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators"* (Hornik u. a., 1989, S. 359).

Bedeutet die Tatsache, dass ein MLP jede beliebige Funktion annähern kann, dass es zweifels-

[3] Elman-Netze waren beispielsweise 3-10 Mal langsamer als MLP.

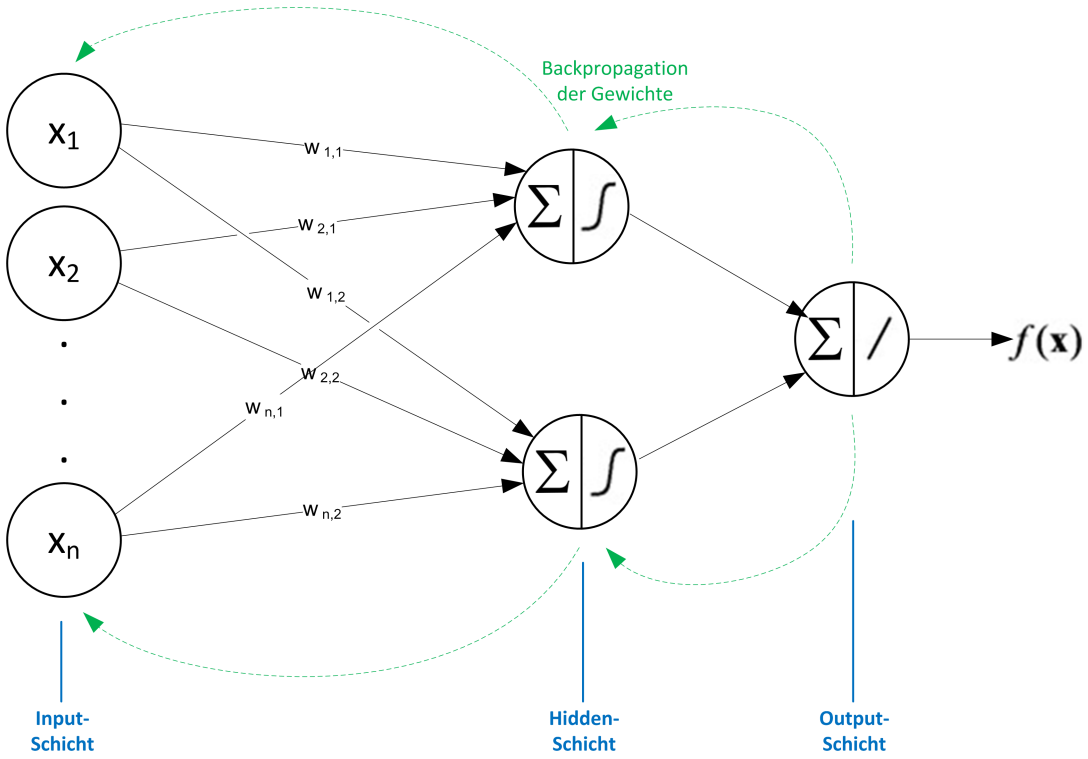


Abbildung 7.2.: Schematische Darstellung eines Multilayer Perceptrons (MLP)

frei in der Lage ist, eine Bedarfsprognose für Backwaren im Einzelhandel zu ermitteln? Nein, denn die grundlegende Voraussetzung dafür ist, dass es überhaupt eine Abbildungsfunktion zwischen Input-Werten, die dem MLP präsentiert werden, und dem Output-Wert, der Prognose, gibt. Wenn dies der Fall ist, dann ist ein MLP jedoch in der Lage diese Funktion zu finden bzw. sie beliebig nah anzunähern.

In Abbildung 7.2 (Seite 32) ist ein MLP mit einer Hidden-Schicht dargestellt. Alle Neuronen einer Schicht, sind mit allen Neuronen der folgenden Schicht verbunden. Die Pfeilrichtungen repräsentieren das Verhalten in Feedforward-Netzen, in denen die Ausgangssignale eines Neurons immer in eine Richtung durch das Netz propagiert werden, vorwärts. In Feedforward-Netzen gibt es keine Rückverbindungen und auch keine Verbindungen zwischen Neuronen derselben Schicht.

Die frühe Entscheidung für das MLP als Netzwerkarchitektur A schränkt die Ausprägungen der im Folgenden vorgestellten Parameter des Tupels $A = [V, I, L]$ bereits ein.

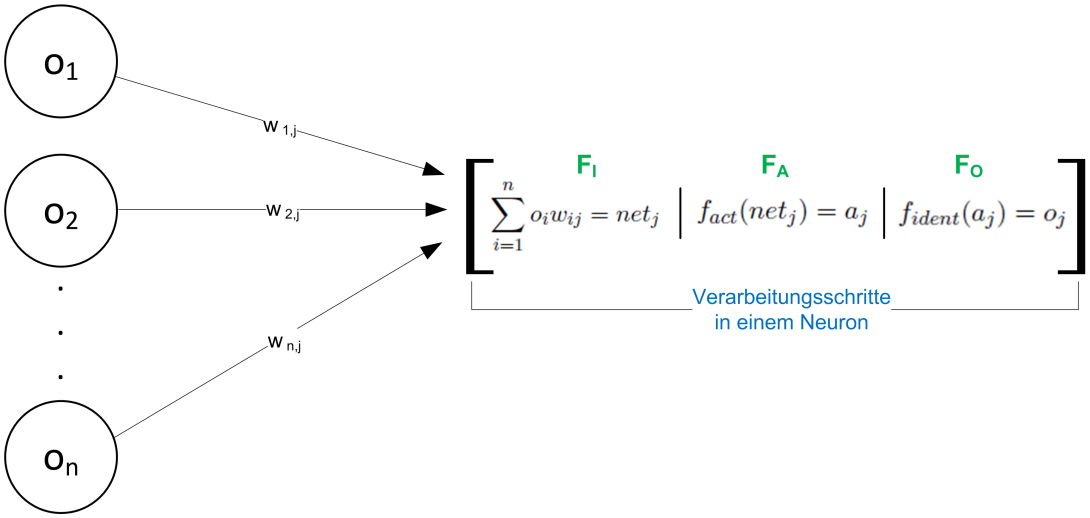


Abbildung 7.3.: Verarbeitungsschritte in einem einzelnen Neuron[4]

7.1.1. Verarbeitungsschritte im Neuron (V)

In einem Neuron finden drei Verarbeitungsschritte statt, die durch die Wahl verschiedener Funktionen und Algorithmen beeinflusst werden können:

1. Die eingehenden Werte der Neuronen aus der vorangegangenen Schicht müssen nach bestimmten Regeln zu einem Wert kombiniert werden. Diese Regel wird durch die Eingabefunktion F_I definiert und liefert die Netzeingabe net_j
2. Die Netzeingabe net_j wird dann in die Aktivierungsfunktion F_A eingesetzt, die bestimmt, wie der Aktivierungszustand a_j des Neurons von der Eingabe aller anderen Neuronen, die mit diesem Neuron verbunden sind, abhängt.
3. Die Ausgabefunktion F_O berechnet die Ausgabeinformation o_j eines Neurons aus seiner Aktivierung a_j

Abbildung 7.3 (Seite 33) stellt diesen Prozess graphisch dar.

Im Folgenden werden verschiedene Ausprägungen von F_I, F_A und F_O vorgestellt und ggf. bereits auf die MLP benutze Variante eingegangen. Es wird jedoch noch nicht darauf hingewiesen in welchen

Eingabefunktion F_I
In MLP fungiert meist folgende Berechnungsvorschrift als Eingabefunktion:

$$net_j = \sum_{i=1}^{n} o_i w_{ij} \tag{7.1}$$

[4]Angelehnt an (Janetzke und Lewandowski, 2012)

Die Berechnungsvorschrift der Summenbildung $F_I = net_j$ aggregiert die Ausgabesignale o_i aller verbundenen Neuronen in den Eingabewert net_j unter Berücksichtigung der variablen Verbindungsgewichte wij, die die Intensität der Beziehungen von Input- und Output-Neuron angeben.

So verstärkt ein positives Verbindungsgewicht mit $w_{ij} > 0$ die Aktivierung des Neurons und hemmt sie bei negativem Verbindungsgewicht mit $w_{ij} < 0$.

Es gibt eine große Auswahl alternativer Eingabefunktionen mit spezifischen Eingenschaften für die Eingabeverarbeitung in unterschiedlichen Netzarchitekturen (Crone, 2010), die in MLP aber selten Anwendung finden und deshalb hier keine weitere Erwähnung finden. Eine kompakte Übersicht der wichtigsten Funktionen kann in (Crone, 2010, S. 170) eingesehen werden.

Aktivierungsfunktion F_A
Anschließend an die Eingabe der Werte in ein Neuron berechnet die Aktivierungsfunktion[5] f_{act} den Aktivierungszustand a_j eines Neurons (Zell, 1997) durch

$$f_{act}(net_j) = a_j \tag{7.2}$$

Der Form der Aktivierungsfunktion kommt eine besondere Bedeutung bei der Modellierung KNN zu (Crone, 2010). Sie bestimmt nämlich, ob ein Neuron durch die Netzeingabe aktiviert wird oder nicht. Die Aktivierungsfunktion wird in einem zweidimensionalen Diagramm visualisiert, wobei auf der x-Achse die Netzeingabe und auf der y-Achse der entsprechende Aktivitätslevel abgetragen wird (Rey, 2013).

Man unterscheidet zwischen verschiedenen Aktivitätsfunktionen, die in Abbildung 7.4 (Seite 35) gezeigt werden, nach (Rey, 2013):

- **Lineare-Funktionen**: Der Zusammenhang zwischen Netzeingabe und Aktivitätslevel ist linear. Bei der hier dargestellten Funktion handelt es sich um die *Identitätsfunktion*, bei der der Ausgabewert gleich dem Eingabewert ist.
- **Binäre-Funktion**: Hier gibt es nur zwei Zustände des Aktivitätslevels, 0 (bzw. manchmal auch -1) oder 1.
- **Sigmoide-Funktionen**[6]: Ist die Netzeingabe groß und negativ, dann ist der Aktivitätslevel nahe 0 (**Logistische-Funktion**) bzw. -1 (**Tangens-Hyperbolicus-Funktion**). Sie steigt dann zunächst langsam an (eine Art Schwelle), woraufhin der Anstieg steiler wird und einer linearen Funktion gleicht. Bei einem hohen Netzinput nähert sich der Wert dann asymptotisch der 1 an. Sigmoide Aktivierungsfunktionen bieten zwei wesentliche Vorteile:

[5] In der Literatur auch als Transferfunktion bezeichnet, u.a. in (Duch und Jankowski, 02.03.2001)

[6] Die Namensgebung der beschriebenen Sigmoiden-Funktionen ist in der Literatur nicht eindeutig. Die Mehrzahl der betrachteten Quellen sehen Sigmoide-Funktionen als Funktionsklasse und die Logistische- und die Tangens-Hyperbolicus-Funktion als mögliche Ausprägungen dieser Funktionsklasse. Es gibt aber auch Autoren, die mit der Sigmoiden-Funktion die Logistische-Funktion meinen und die Tangens-Hyperbolicus-Funktion als bipolare Sigmoide-Funktion bezeichnen.

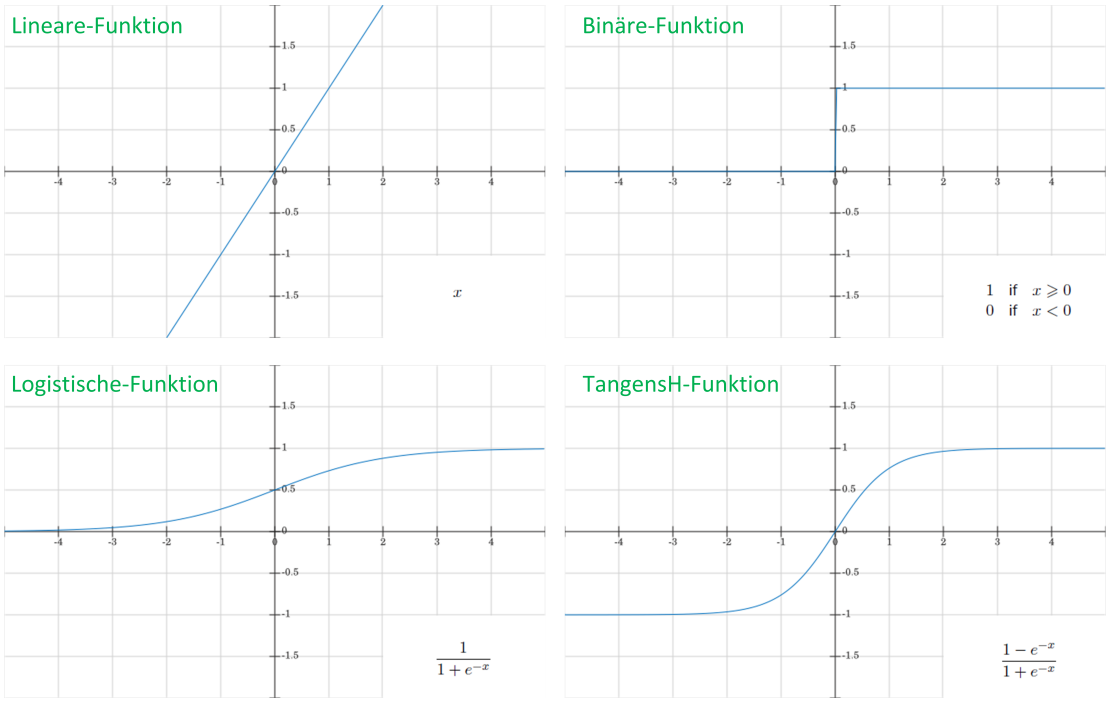


Abbildung 7.4.: Eine Auswahl der gängigsten Aktivierungsfunktionen

- *Begrenzung des Aktivitätslevels*: Im Gegensatz zu den linearen Aktivitätsfunktionen ist der Aktivitätslevel hier sowohl nach oben als auch nach unten begrenzt.
- *Mögliche Differenzierbarkeit*: Im Gegensatz zu der binären Schwellenfunktion ist die Funktion an allen Stellen differenzierbar, was beispielsweise eine notwendige Voraussetzung für das in Abschnitt 7.1.3 (Seite 39) vorgestellte Gradientenabstiegsverfahren (im Rahmen des Lernverfahrens Backpropagation) ist.

Darüber hinaus existieren weitere semilineare, trigonometrische und hyperbolische Aktivierungsfunktionen, die jedoch selten Anwendung bei der Prognose finden (Crone, 2010).

Zur Verdeutlichung der unterschiedlichen Aktivitätslevel zeigt die folgende Tabelle 3 exemplarische Netzeingaben net_j und die daraus resultierenden Aktivitätslevel der einzelnen Aktivierungsfunktionen f_{act}.

	Aktivitätslevel		
Netzeingabe	**Lineare-Funktion**	**Binäre-Funktion**	**Logistische-Funktion**
4	4	1	0,982
0	0	0	0,500
-4	-4	0	0,018

In MLP findet bei den Neuronen der Input- und Output-Schicht standardmäßig die lineare Identitätsfunktion Anwendung, wohingegen die Wahl der Aktivierungsfunktion bei den Neuronen der Hidden-Schicht vom Anwendungsfall abhängt und zu den wichtigen Stellschrauben der KNN-Konfiguration zählt.

Ausgabefunktion F_O
Die Ausgabefunktion *"[...] erlaubt die Kodierung der Ausgabe durch Skalierung der Aktivierungswerte in ein abweichendes Intervall sowie die Modellierung z.B. statistisch motivierter Verhaltensweisen spezifischer KNN-Architekturen"* (Crone, 2010, S. 175). Die Ausgabefunktion ist bei einem MLP festgelegt auf die im Folgenden dargestellte Identitätsfunktion und wird somit in den meisten Darstellungen und Modellierungen vernachlässigt (Crone, 2010).

$$f_{ident}(a_j) = a_j = o_j \tag{7.3}$$

7.1.2. Informationsverarbeitung und Propagierung (I)

"Die Informationsverarbeitung im KNN erfolgt in Analogie zum biologischen Vorbild sequentiell und unidirektional. Durch gewichtete, gerichtete Verbindungen eines Neurons mit anderen Neuronen entsteht ein Netzwerk von Neuronen, welches entsprechend als künstliches Neuronales Netz bezeichnet wird" (Crone, 2010, S. 197). Die drei Komponenten $[T, W, K]$ die daran beteiligt sind die Input-Werte durch das Netz zu propagieren und eine Menge an Ausgabewerten zu produzieren, sollen im Folgenden kurz erläutert werden.

Netzwerktopologie T
Im Rahmen dieser Arbeit werde ich die Netzwerktopologie über die Anzahl an Schichten und der darin enthaltenen Neuronen definieren. Die Topologie des Netzwerkes in Abbildung 7.2 (Seite 32) könnte tabellarisch somit wie folgt dargestellt werden:

	Schicht 1 (Input-Schicht)	Schicht 2 (Hidden-Schicht)	Schicht 3 (Output-Schicht)
Anzahl Neuronen p/Schicht	n	2	1

Bezüglich der Topologie gibt es also vier Entscheidungen zu treffen:

1. Anzahl der Neuronen in Input-Schicht
2. Anzahl der Hidden-Schichten
3. Anzahl der Neuronen pro Hidden-Schicht
4. Anzahl der Neuronen in Output-Schicht

Die Entscheidungen bzgl. 1. und 4. sind reine Modellierungsfragen und ausschließlich davon abhängig, welche Informationen man in das KNN hineingeben möchte und welche man herausbekommen möchte. Diese Fragen werden ausführlich in Abschnitt 11.2 behandelt.

Für die Entscheidungen bzgl. der Anzahl an Hidden-Schichten und der Menge an Neuronen pro Hidden-Schicht gibt es weder harte fachliche noch harte technische Kriterien und trotzdem kommt ihnen eine sehr große Bedeutung zu. *"Certain problems are more easily solved by particular network [topologies]. For MLP networks, the number of hidden neurons can mean the diffenrence between success and failure"* (Masters, 1993, S. 173). Mit jedem Hidden-Neuron und/oder jeder Hidden-Schicht steigt sowohl die Trainingszeit des Netzes als auch die Gefahr des Overfittings (beide Konzepte werden in den Abschnitten 7.1.3 und 8.4 ausführlich behandelt). Die Wahl einer zu einfachen Netzwerktopologie wiederum vermindert die Lernfähigkeit des Netzes und kann dazu führen, dass das modellierte Problem gar nicht gelöst werden kann.

Auch wenn es keine festen Regeln für die zu wählende Anzahl an Hidden-Schichten gibt, so sind sich die Experten doch zumindest einig, dass so wenige Hidden-Schichten wie möglich verwendet werden sollten. Boyd und Kaastra geben nach eigener Literaturrecherche die Empfehlung, dass *"[...] all neural networks should start with preferably one or at most two hidden layers. [...] Both theory and virtually all empirical work to date suggests that networks with more than four layers will not improve the results"* (Kaastra und Boyd, 1996). Diese Auffassung wird von Klimasauskas in (Klimasauskas, 1996) ebenfalls vertreten. Masters geht sogar noch weiter und sagt in (Masters, 1993), dass es keinen theoretischen Grund gibt, mehr als zwei Hidden-Schichten einzusetzen und die Praxis zeigt, dass selbst zwei in fast allen Fällen schon zu viele sind und eine Hidden-Schicht meist ausreicht.

Bezüglich der Anzahl Hidden-Neuronen pro Hidden-Schicht weisen alle Autoren darauf hin, dass dieser Schritt der KNN Konfiguration viel Experimentieren und Ausprobieren erfordert - vor allem, weil sich die aufgestellten 'Daumenregeln' aus der Literatur deutlich von einander unterscheiden.

Baily und Thompson in (Bailey und Thompson, 1990) empfehlen bei einem 3-Schichten KNN (also bei einer Hidden-Schicht), 75% der Anzahl Input-Neuronen als Hidden-Neuronen einzusetzen. Masters hat in (Masters, 1993) die Geometrische-Pyramiden-Regel aufgestellt (englisch: *geometric pyramid rule*), bei der ein 3-Schichten KNN mit n Input-Neuronen und m Output-Neuronen $\sqrt[2]{n * m}$ Hidden-Neuronen haben sollte. Heaton und Moustafa gehen in ihren Werken (Heaton, 2011) und (Moustafa, 2011) davon aus, dass man meist mit einer Menge Hidden-Neuronen, die doppelt so groß ist wie die Anzahl Input-Neuronen, gut zurecht kommen sollte. Klimasauskas hingegen stellt in (Klimasauskas, 1996) eine Regel auf, die lediglich eine Obergrenze für die Anzahl Hidden-Neuronen in Bezug zu der Anzahl Trainingsdaten angibt. Er empfiehlt, dass die Anzahl Trainingspaare (genaue Definition des Begriffs in Abschnitt 7.1.3) mindestens so groß sein sollte wie die Anzahl der Verbindungsgewichte im Netzwerk.

Wenn man nun diese vier Empfehlungen auf ein exemplarisches Netz mit 20 Input-Neuronen und 3 Output-Neuronen anwendet, ergibt sich folgende Anzahl an Hidden-Neuronen und minimaler Anzahl Trainingspaare:

	Anzahl Neuronen in Hidden-Schicht	Minimal benötigte Anzahl Trainingspaare nach Klimas-auskas
Masters	$\sqrt[2]{20*3} = 7,7 =$ **8**	$(20*8)+(8*3) =$ **184**
Baily und Thompson	$20*0,75 =$ **15**	$(20*15)+(15*3) =$ **345**
Heaton, Moustafa	$20^2 =$ **40**	$(20*40)+(40*3) =$ **920**

Die Konsequenz der Annahme von Klimasauskas wäre somit, die Anzahl der Hidden-Neuronen zu reduzieren, sollten nicht genug Trainingspaare vorhanden sein.

Struktur der Verbindungsgewichte W
Die Abbildung des Verbindungsnetzwerkes der Zellen erfolgt anhand der Topologie des gewichteten Graphen oder in einer Gewichtsmatrix aller Verbindungsgewichte W, wobei der Eintrag w_{ij} die Art und Stärke der künstlichen synaptischen Verbindung zwischen dem sendenden Neuron u_i und dem empfangenden Neuron u_j bestimmt (Crone, 2010).

$W = [w_{ij}]$
mit $w_{ij} = 0$ keine Verbindung zwischen Neuron i und j
$w_{ij} < 0$ Neuron i *hemmt* seinen Nachfolger j durch ein Gewicht der Stärke $|w_{ij}|$
$w_{ij} > 0$ Neuron i *erregt* seinen Nachfolger j durch ein Gewicht der Stärke $|w_{ij}|$

"Die Verbindungsmatrix spezifiziert neben der grundlegenden Topologie des KNN auch die Struktur des Informationsflusses im Netzwerk und somit auch die Art der Informationsverarbeitung" (Crone, 2010, S. 181). Anhand der Struktur der Verbindungsmatrix unterscheidet man Netze ohne Rückkopplung (Feedforward-Netze) und Netze mit Rückkopplung (Rekurrente-Netze).

In Feedforward-Netzen existiert kein Pfad, der von einem Neuron u_i direkt oder indirekt wieder zum Neuron u_i zurück führt. Mathematisch entspricht dies einem azyklischen Graph. In der Matrixschreibweise ist lediglich die obere Dreiecksmatrix mit Werten $wij \neq 0$ besetzt (Zell, 1997) - wie in Abbildung 7.5 A dargestellt. Rekurrente-Netze finden in dieser Arbeit keine Anwendung, werden aber in Teil B von Abbildung 7.5 kurz dargestellt, um anzudeuten, dass es auch komplexere Ansätze gibt.

Aktivierungsstrategien der Informationsverarbeitung K
"Die Aktivierungsstrategie K spezifiziert die Reihenfolge der einzelnen Berechnungsschritte zur Informationsverarbeitung in KNN" (Crone, 2010, S. 183).

Grundsätzlich gibt es in Feedforward-Netzen eine eindeutig determinierte schrittweise Berechnungsvorschrift von der Input- zur Output-Schicht. Die Input-Werte, werden über einen n-dimensionalen Input-Vektor $(x_1, x_2, ..., x_n)$ angelegt und über die Neuronen der Input-Schicht weitergeleitet an die Hidden-Schicht und von dort zu den Neuronen der Output-Schicht. Dort wird das Ergebnis in Form eines m-dimensionalen Output-Vektors $(y_1, y_2, ..., y_m)$ an die Umwelt herausgegeben (Zell, 1997).

Die Kombination der vorgestellten Modellierungsparameter und ihrer Ausprägungen - der Ver-

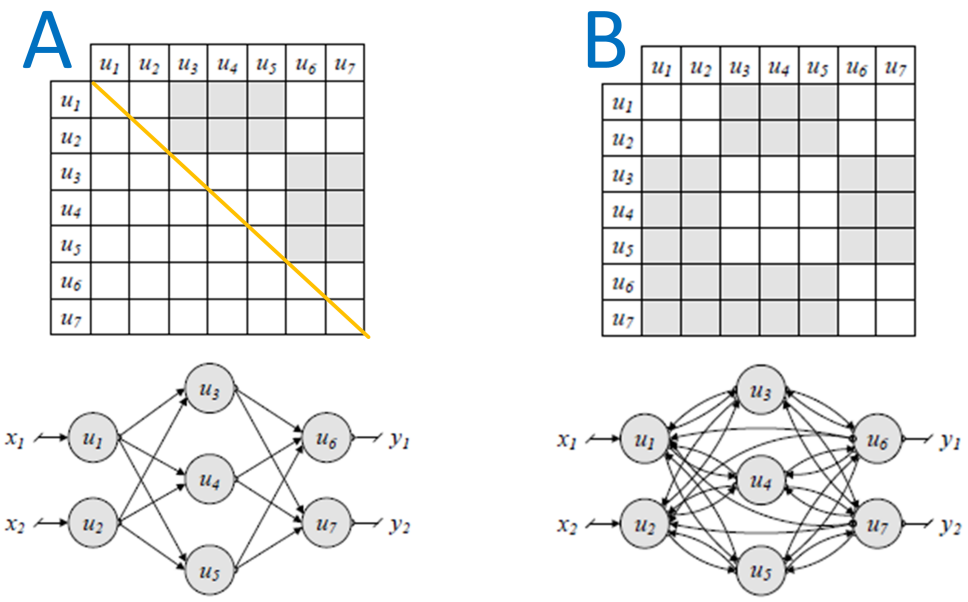


Abbildung 7.5.: Topologien und Verbindungsmatritzen für A) vorwärtsgerichtete KNN ohne schichtübergreifenede Verbindungen und B) indirekte rückgekoppelte KNN

arbeitungsfunktion für eingehende Signale im Neuron V, der Anordnung von Schichten und Neuronen in der Topologie T, der Verbindungsstruktur der Gewichtsmatrix W und die Aktivierungsstrategie zur Propagierung von Werten durch das KNN K - spezifizieren die Architektur A eines KNN fast vollständig (Alex, 1998). Zur Vervollständigung des Tupels $A = [V, I, L]$ fehlt nun noch die Festlegung des Lernprozesses L, der im nächsten Abschnitt beschrieben wird.

7.1.3. Parametrisierung des Lernprozesses (L)

Nachdem ein KNN durch seine Architektur spezifiziert wurde, wird es zur Anwendung einer konkreten Aufgabenstellung parametrisiert. Dies erfolgt durch die Propagierung von Trainingsdaten und die daraus resultierende Anpassung der Verbindungsgewichte und wird als Lernen oder Training bezeichnet. Der Lernprozess L wird durch das Lernziel Z, den Lernalgorithmus G, das gewählte Stopkriterium S und die Fehlerfunktion O charakterisiert und kann als Tupel $L = [Z, G, S, O]$ abgebildet werden.

Lernziel Z

Abbildung 7.6 (Seite 40)[7] verdeutlicht den Zusammenhang zwischen Lernzielen und Lernalgorithmen. Die Lernziele des assoziativen oder explorativen Lernens werden lt. T. Kohonen (Kohonen, 1989) durch die Anwendungsdomäne bestimmt und diese Lernziele bestimmen wiederum die Auswahl der Lernalgorithmen (Levine, 2000).

[7] Ursprünglich aus (Rojas, 1993), modifiziert von (Crone, 2010)

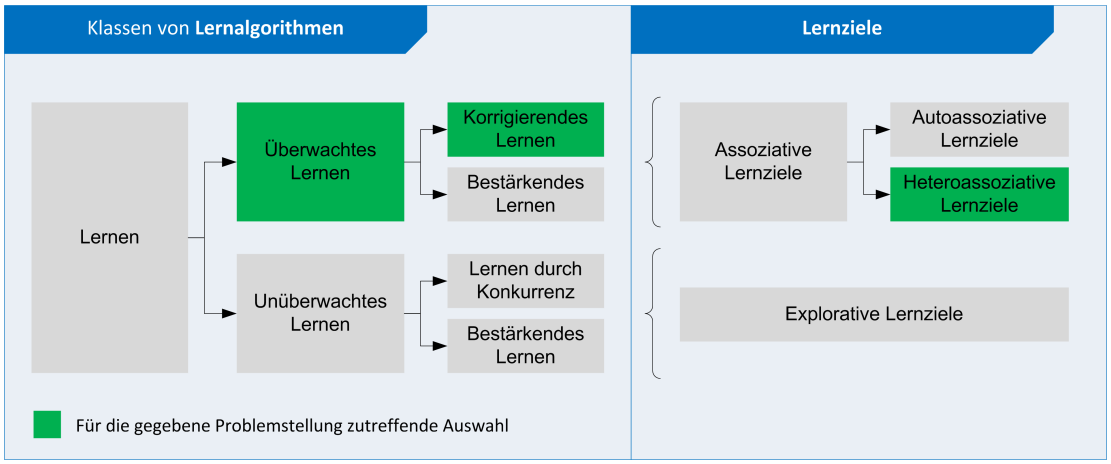


Abbildung 7.6.: Zusammenhang von Lernzielen und Lernalgorithmen

Bei einem *assoziativen* Lernen von Mustern werden freie Parameter derart verändert, dass nach wiederholter Präsentation von Paaren aus bekannten Eingabe- und Ausgabemustern die korrekte Assoziation für bekannte Muster selbständig vorgenommen werden kann. Darüber hinaus werden unbekannte, aber ähnliche Eingabemuster im Sinne einer Generalisierung korrekt zugeordnet (Zell, 1997). Dies entspricht der in Abschnitt 1.1 vorgestellten Problemstellung.

In den Anwendungen des *heteroassoziativen* Lernens sind die Vektoren der Eingabemuster und Ausgabemuster verschieden, $\vec{x} \neq \vec{y}$. Im Gegensatz dazu wird beim *homoassoziativen* Lernen, das Eingabemuster auf identische Ausgabemuster abbildet, also $\vec{x} \rightarrow \vec{y}$. Homoassoziatives Lernen erfordert also eine identische Anzahl von Neuronen in der Input- und Output-Schicht eines KNN (Anderson, 1995). Das heteroassoziative Lernziel entspricht der Modellierung und Parametrisierung von Prognosemodellen, da aus einer Menge abhängiger und/oder unabhängiger Variablen eine Prognose abgeleitet wird (Crone, 2010).

Lernalgorithmus G

Aus Abbildung 7.6 (Seite 40) wird bereits deutlich, dass Lernziele des assoziativen Lernens, durch überwachte Lernalgorithmen abgebildet werden (Anderson, 1995). *"Beim überwachten Lernen (engl. supervised learning) gibt ein externer 'Lehrer' zu jedem Eingabemuster der Trainingsmenge das korrekte bzw. beste Ausgabemuster an"* (Crone, 2010, S. 191), was im Allgemeinen als Trainingspaar bezeichnet wird.

Algorithmen für ein überwachtes, heteroassoziatives Training von KNN zur Lösung nicht linearer Modelle wurden erst 1986 durch die Forschergruppe um David E. Rummelhart (Rummelhart u. a., 1986) entwickelt. Sie erfanden den heute allgegenwärtigen Lernalgorithmus *Backpropagation*[8], ein im Vergleich zu den bisherigen Lernverfahren sehr schnelles und robustes Lernverfahren für MLP, das sich mathematisch elegant als Gradientenabstiegsverfahren des Abbildungsfehlers herleiten lässt (Lippe, 2006).

[8] In ihrer Veröffentlichung als "error back-propagation" bezeichnet.

Initial sind die Gewichte aller Verbindungen mit Zufallswerten vorbelegt. Wird ein Input-Vektor $\vec{x}$ angelegt, wird gemäß der Arbeitsweise der einzelnen Neuronen (siehe erneut Abbildung 7.3, Seite 33) in Abhängigkeit zu V (Verarbeitungsschritte im Neuron) und K (Aktivierungsstrategie) ein Output-Vektor $*\vec{y}$ ermittelt. Dieser weicht im Allgemeinen vom vorgegebenen Zielwert $\vec{y}$ ab. Damit das MLP die Abbildungsvorschrift $f(\vec{x})$→$\vec{y}$ erlernen kann, wird für jedes Trainingspaar $(\vec{x}, \vec{y})$ während des Vorwärtsdurchlaufs ein $*y$ mit einem Fehler $\delta = y - y*$ ermittelt. Dieser Fehler δ wird nun im Rückwärtsdurchlauf des Trainings zurück propagiert, von der Output-Schicht in Richtung Input-Schicht (Thiesing, 1996). Ein kompletter Durchlauf eines Trainingspaares besteht aus Vorwärtsdurchlauf und Rückwärtsdurchlauf und wird im Allgemeinen als *Epoche* oder *Iteration* bezeichnet.

Das Trainingsset wird vom KNN so oft durchlaufen, bis das **Stopkriterium** S erreicht ist. Dieses ist in der Regel eine vorgegebene maximale Anzahl an Iterationen oder ein gewünschter minimaler Abbildungsfehler. Am Ende einer Trainingsphase ist also immer der Abbildungsfehler (im Allgemeinen Trainingsfehler (TE) oder *Netzwerkfehler* genannt) des KNN bekannt. *"The error is the degree to which the neural network output matches the desired output"* (Heaton, 2011, S. 14). Es existieren viele verschiedene Algorithmen zur Berechnung des Trainingsfehlers u.a. Mean Square Error (MSE), Sum of Squares Error (ESS), Root Mean Square Error (RMS), wobei die Mean Square Error Berechnung häufig die Default-Einstellung in KNN-Frameworks ist (Heaton, 2011) und in den meisten Fällen den aussagekräftigsten Fehler liefert (Masters, 1993). Die Fehlerberechnung erfolgt nach der in Abbildung 7.4 dargestellten Formel, mit n = Anzahl der Trainingspaare.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - *y_i)^2 \tag{7.4}$$

Abbildungsfehler können sowohl positiv als auch negativ sein, in anderen Worten: der ermittelte Wert kann höher oder niedriger sein, als der Zielwert. Würde man diese einfach aufaddieren und den Mittelwert bilden, so würden sich hohe positive und hohe negative Fehler gegenseitig aufheben und einen falschen Anschein erwecken. Durch das Quadrieren des Abbildungsfehlers wird diese Fehleinschätzung umgangen.

Die **Fehlerfunktion** O wird aus dem Paar (Gewichtsmatrix W, MSE) gebildet, das nach jeder Iteration aktualisiert wird. *"Ziel des Lernens ist es nun ein - möglichst globales - Minimum dieser Fehlerfunktion zu erreichen"* (Lippe, 2006, S. 84). Der Backpropagation-Algorithmus beruht auf einem Gradientenabstiegsverfahren, das von folgender Annahme ausgeht: Ist die Ableitung der Fehlerfunktion einer Gewichtsmatrix $\ddot{\vec{w}}_{aktuell}$ ungleich Null, so muss das nächste Minimum $\ddot{\vec{w}}_{min}$ der Fehlerfunktion in Richtung des Gradientenabstieges liegen. Siehe dazu Abbildung 7.7 (Seite 42).

Auf eine detailliertere Darstellung des Gradientenabstiegsverfahren und des Backpropagation-Algorithmus wird im Folgenden verzichtet. Das Verfahren ist bestens bekannt und von allen Seiten beleuchtet worden und steht nicht im Fokus dieser Arbeit, kann jedoch sowohl im Ori-

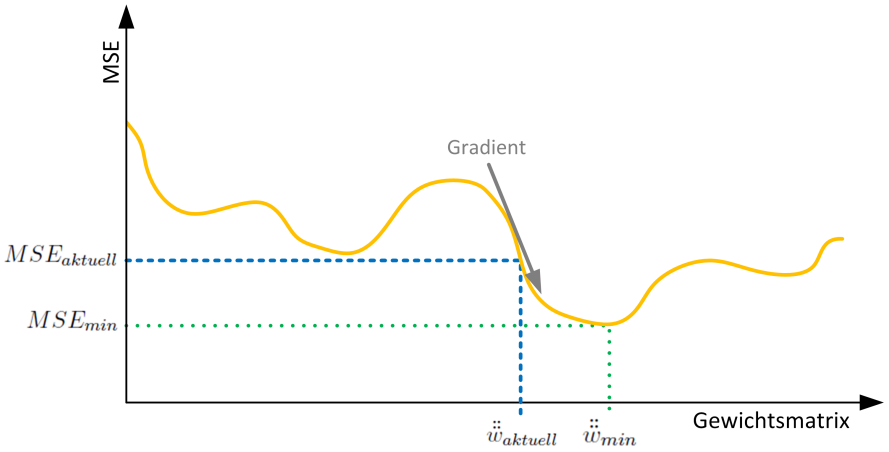


Abbildung 7.7.: Gewichtsmatrix und zugehörender Fehler für aktuelle und gewünschte Situation

ginalwerk von (Rummelhart u. a., 1986) als auch in einer kompakten und verständlichen Aufbereitung von (Lippe, 2006, S.43-54) nachgelesen werden.

Teil III.

Praktische Grundlagen

8. Praktische Anwendung Künstlicher Neuronaler Netze

Das Tupel $A = [V, I, L]$ definiert ein KNN, beschreibt aber noch nicht, wie es trainiert, validiert und getestet wird, welche Modellierungsentscheidungen der Netztopologie T zugrunde liegen und wie die historischen und zukünftigen Daten strukturiert und modelliert sein müssen. Dieses Kapitel bildet zusammen mit den Grundlagen der KNN (aus Kapitel 7) die Basis für die spätere Beschreibung der praktischen Umsetzung in Kapitel 11.

Das konkrete Vorgehen beim Einsatz von KNN wird in der Literatur übereinstimmend wie folgt beschrieben[1]:

1. Modelliere ein KNN gemäß der gegebenen Problemstellung.
2. Konfiguriere das Tupel $A = [V, I, L]$ dieses KNN.
3. Forme Datenpaare aus allen zur Verfügung stehenden historischen Daten, in der Form: (Input-Vektor $\vec{x}$ | Ziel-Vektor $\vec{y}$).
4. Unterteile die resultierenden Datenpaare in drei disjunkte Mengen, mit einer empfohlenen Verteilung von 50-70% Trainingsset, 20-30% Validierungsset und 10-20% Testset.
5. Trainiere das KNN auf dem Trainingsset, bis das Stopkriterium S erreicht ist.
6. Validiere das KNN gegen das Validierungsset.
7. Teste das KNN gegen das Testset.

Die folgenden Abschnitte werden auf die o.g. Punkte in der skizzierten Reihenfolge eingehen. Die Ausnahme bilden lediglich Punkt 2, der bereits im vorangegangenen Kapitel 7 abgedeckt wurde und Punkt 4, dessen Erläuterung ans Ende dieses Kapitels gestellt wurden, da Wissen über die Eigenschaften der Trainings-, Validierungs- und Testphasen benötigt wird, das vorher vermittelt werden soll.

[1] u.a. in Masters (1993), Kaastra und Boyd (1996), Callan (2003), Ng (2012b)

8.1. Modellierung von Prognosen mit Künstlichen Neuronalen Netzen

In Abschnitt 5.2 wurde bereits ausführlich auf die Unterscheidung zwischen Zeitreihen-, kausalen oder kombinierten Prognosen eingegangen. Die dort speziell in Unterabschnitt 5.2.5 (Seite 18) aufgestellten Annahmen und die dort etablierte Notation werden in diesem Abschnitt vorausgesetzt. Es soll nun gezeigt werden, wie diese theoretisch eingeführten Modellierungsformen durch ein KNN abgebildet werden. Im Gegensatz zu den herkömmlichen Verfahren, bei denen jede dieser Modellierungsformen einer eigenen Klasse an Berechnungsverfahren zugeteilt werden muss, unterscheidet eine KNN Modellierung diese Modellierungsformen lediglich durch die Wahl der Netztopologie T und die Semantik der Daten.

Die folgenden Abbildungen werden an einem formalen Beispiel zeigen, wie sich die drei gezeigten Modellierungsformen auf die Netztopologie, davon speziell auf die Anzahl der Input-Neuronen, auswirken.

Abbildung 8.1 (Seite 46) zeigt die Modellierung einer Zeitreihenprognose mit $n = 6$ und $m = 0$. Als prognoserelevant wurden hier also exemplarisch die vergangenen 6 Ausprägungen der abhängigen Variable y angenommen. Der obere Teil der Grafik zeigt die bereits bekannte Darstellungsform, die im unteren Teil konkretisiert und auf ein KNN abgebildet wird. Jede Realisierung von y wird also auf ein Input-Neuron abgebildet und die Netzausgabe entspricht der Prognose für den Prognosezeitpunkt $t + 1$.

Die Modellierungsform entscheidet also über die Anzahl der Input-Neuronen, die ein KNN haben muss, um einen Prognosegegenstand mit dem gewählten Modell prognostizieren zu können. Wie Abschnitt 7.1.2 gezeigt hat, gehen die meisten Experten davon aus, dass die Anzahl der Hidden-Neuronen wiederum von der Anzahl der Input-Neuronen abhängt. In den hier gezeigten Abbildungen wurde die Anzahl der Hidden-Neuronen anhand der Pyramiden-Regel von Masters (Masters, 1993) ermittelt.

Abbildung 8.2 (Seite 47) zeigt die Modellierung einer kausalen Prognose mit $n = 0$ und $m = 1$. Dargestellt sind exemplarisch zwei unabhängige Variablen x_1 und x_2, aus deren Verlauf jeweils die Ausprägungen zum Zeitpunkt $t+1$ prognoserelevant sind. Hier wird deutlich, dass über die Input-Neuronen Werte in das Netz einfließen, die unabhängig vom historischen Verlauf des Prognosegegenstands y sind.

Abbildung 8.3 (Seite 48) zeigt die Modellierung einer möglichen Form der kombinierten Prognose mit $n = 2$ und $m = n+1$. Dem KNN werden also sowohl die abhängigen und unabhängigen Variablen der vergangenen 2 Tage zugeführt als auch die unabhängigen Variablen zum Prognosezeitpunkt $t + 1$.

Die kombinierte Prognose ist nicht auf die dargestellte Modellierung beschränkt. Abbildung 5.1 (Seite 14) hat bereits eine alternative Modellierungsmöglichkeit aufgezeigt und Abschnitt 11.2 wird weitere Möglichkeiten vorstellen.

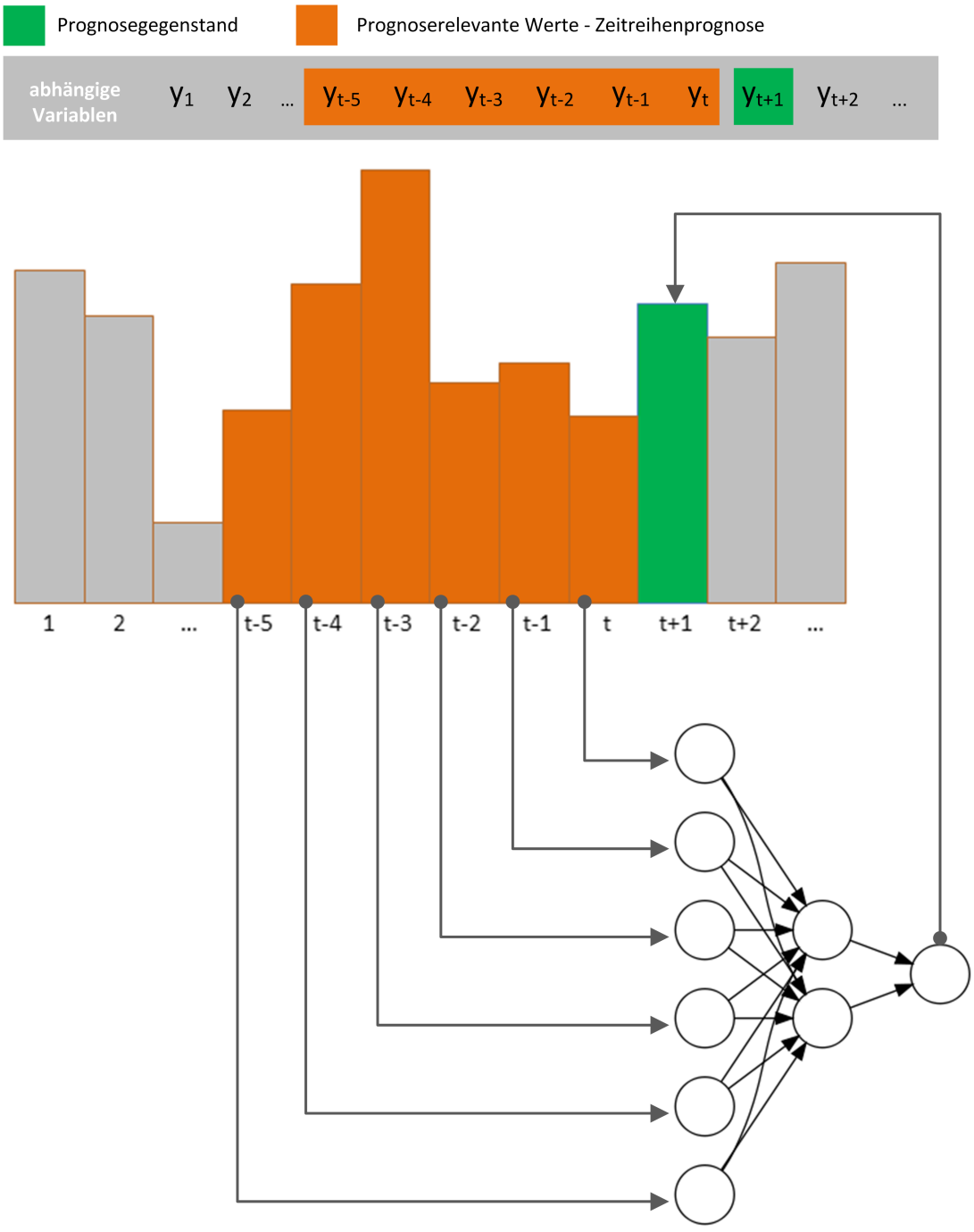


Abbildung 8.1.: **Zeitreihen**-Prognose

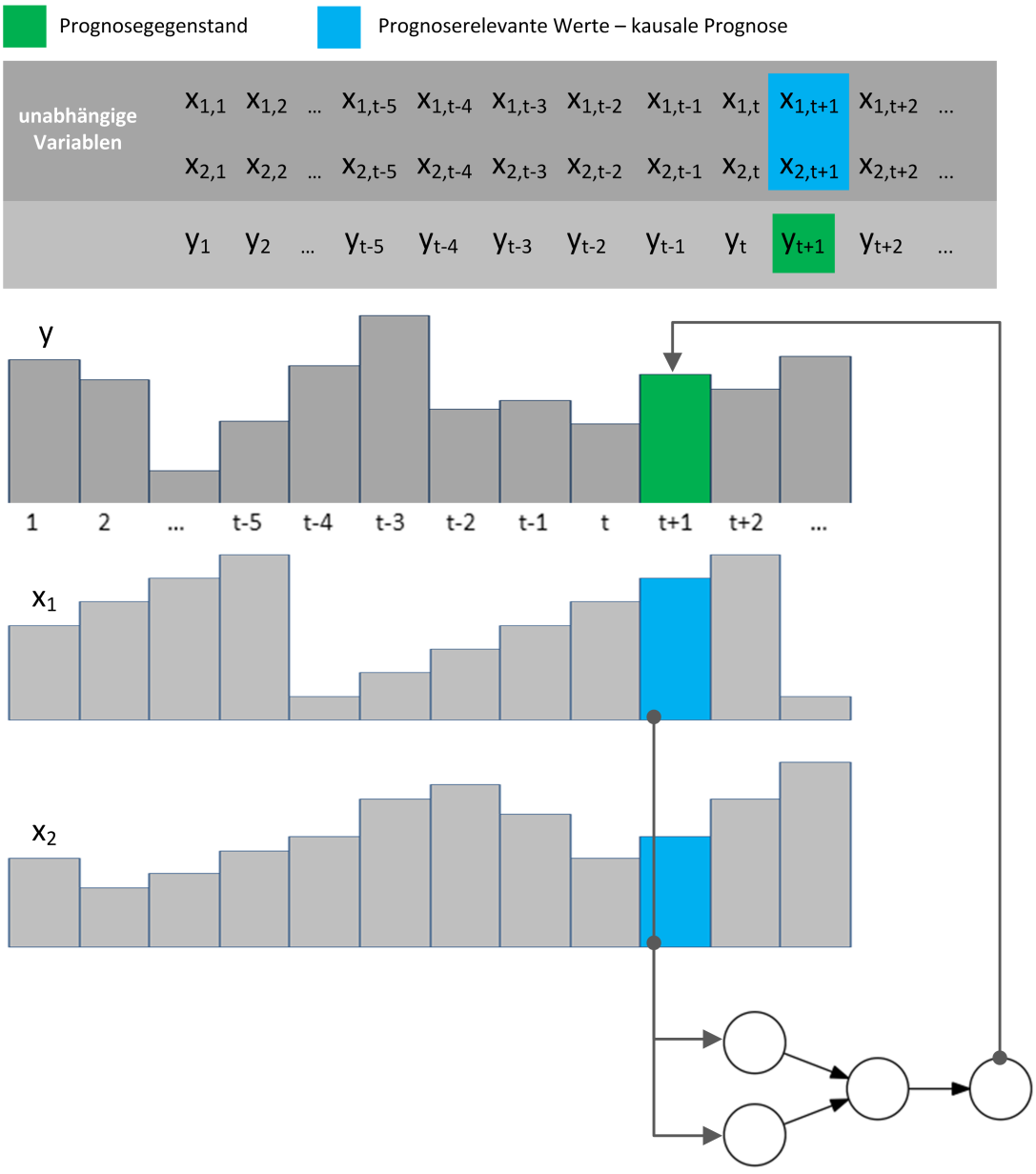


Abbildung 8.2.: **Kausale** Prognose

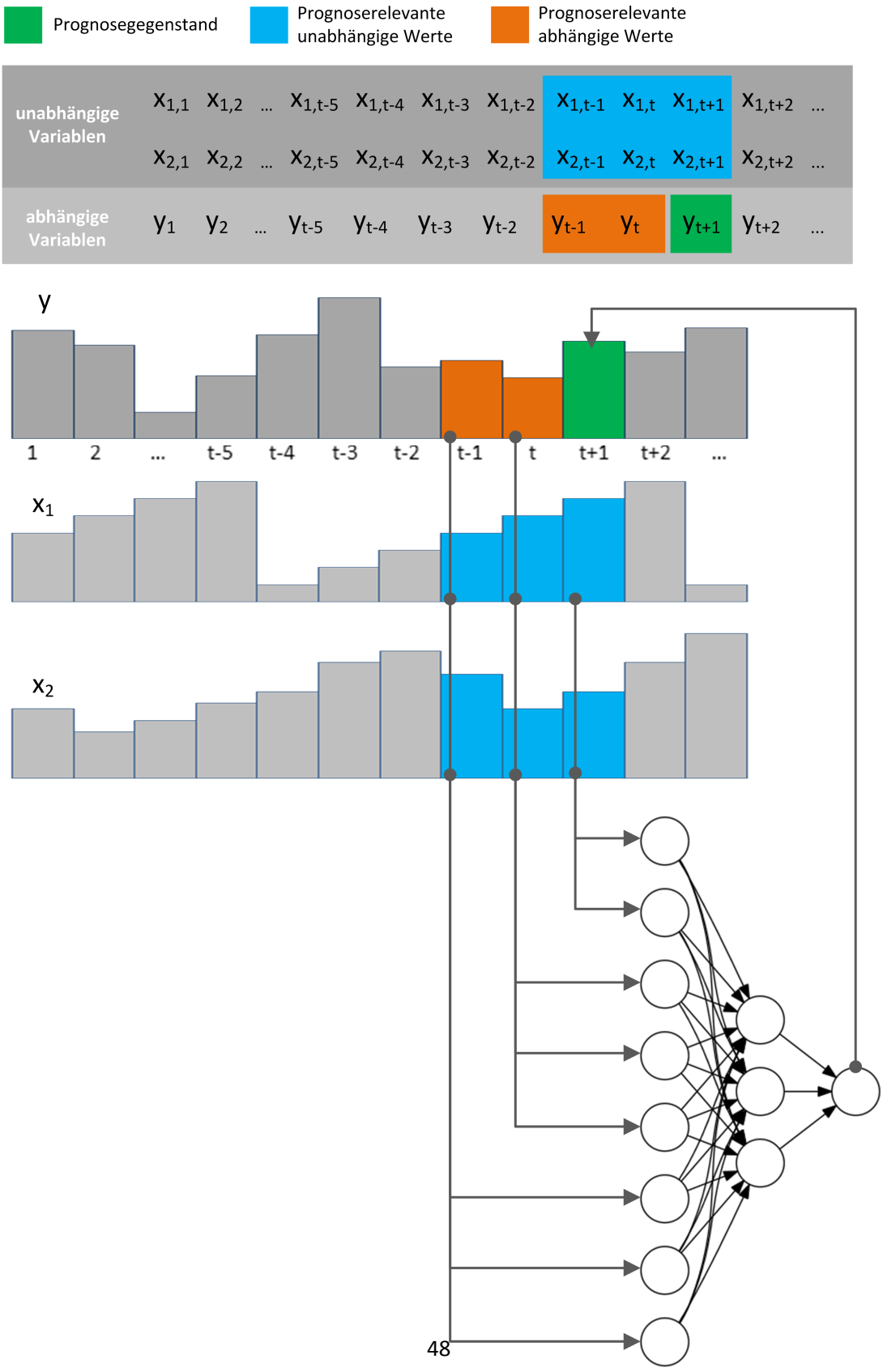


Abbildung 8.3.: **Kombinierte** Prognose

8.2. Strukturierung von historischen Datenpaaren

Durch die Wahl der Modellierungsform und ggf. die Festsetzung eines historischen Zeitfensters wird also die Anzahl der Input-Neuronen festgelegt. Die Anzahl der Output-Neuronen ergibt sich aus dem Prognosegegenstand. Somit ist sowohl die Struktur des Input-Vektors $\vec{x}$ als auch die des Output-Vektors $\vec{y}$ definiert. Ein KNN im produktiven Einsatz bekommt also die prognoserelevanten Informationen über den Input-Vektor geliefert und stellt über den Output-Vektor die ermittelte Prognose zur Verfügung. Ein KNN in der Entwicklungsphase benutzt hingegen den Output-Vektor historischer Daten als Ziel-Vektor für den bereits betrachteten Lernprozess L und für die in den folgenden Abschnitten beschriebenen Validierungs- und Testphasen. Der vorliegende Abschnitt wird deshalb zeigen, wie aus den vorhandenen historischen Daten eine Menge an Datenpaaren, bestehend aus Input- und Ziel-Vektor, geformt wird.

Angenommen der Prognosegegenstand y ist der Tagesbedarf eines beliebigen Produktes. Mögliche Einflussvariablen seien x_1, x_2, x_3 und x_4. Wenn nun Daten für z = 28 vergangenen Tage zur Verfügung stünden[2], dann könnten, je nach Modellierungsform und Festsetzung des historischen Zeitfensters n, u.a. folgende Datensets gebildet werden. Die Variablen vor dem | zeigen den Input-Vektor jedes Datenpaars, die danach den Ziel-Vektor.

8.2.1. Zeitreihenmodellierung

mit $n = 7$

Datenpaar 1 : $y_{28}, y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22}$ | y_{21}
Datenpaar 2 : $y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21}$ | y_{20}
Datenpaar 3 : $y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21}, y_{20}$ | y_{19}
Datenpaar i : ...
Datenpaar 20 : $y_9, y_8, y_7, y_6, y_5, y_4, y_3$ | y_2
Datenpaar 21 : $y_8, y_7, y_6, y_5, y_4, y_3, y_2$ | y_1

Bei 28 zur Verfügung stehenden historischen Tagen und einem prognosrelevanten historischen Zeitfenster n von 7 Tagen ist der frühestmögliche Tag, für den ein vollständiger Input-Vektor gebildet werden kann, y_{21}. Die Werte des Input-Vektors wandern dann pro Datenpaar einen Tag in die Zukunft, so dass aus z zur Verfügung stehenden Tagen $z - n$ Datenpaare geformt werden können. Das gleiche wird im Folgenden nochmals mit n = 14 veranschaulicht.

mit $n = 14$

Datenpaar 1 : $y_{28}, y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21}, y_{20}, y_{19}, y_{18}, y_{17}, y_{16}, y_{15}$ | y_{14}
Datenpaar 2 : $y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21}, y_{20}, y_{19}, y_{18}, y_{17}, y_{16}, y_{15}, y_{14}$ | y_{13}
Datenpaar 3 : $y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21}, y_{20}, y_{19}, y_{18}, y_{17}, y_{16}, y_{15}, y_{14}, y_{13}$ | y_{12}
Datenpaar i : ...

[2] Ein Zeitraum von 28 Tagen wäre für die meisten Prognosen zu kurz, soll hier aber angenommen werden, um das Beispiel überschaubar zu halten.

Datenpaar 13 : $y_{16}, y_{15}, y_{14}, y_{13}, y_{12}, y_{11}, y_{10}, y_9, y_8, y_7, y_6, y_5, y_4, y_3$ | y_2
Datenpaar 14 : $y_{15}, y_{14}, y_{13}, y_{12}, y_{11}, y_{10}, y_9, y_8, y_7, y_6, y_5, y_4, y_3, y_2$ | y_1

8.2.2. Kausale Modellierung

Die kausale Modellierung unterscheidet sich von der Zeitreihenmodellierung dahingehend, dass bei der kausalen Modellierung alle zur Verfügung stehenden Tage z zu Datenpaaren verarbeitet werden können.

mit $m = 1$ und x_1, x_2, x_3 und x_4 als Einflussvariablen
Datenpaar 1 : $x_{1,28}, x_{2,28}, x_{3,28}, x_{4,28}$ | y_{28}
Datenpaar 2 : $x_{1,27}, x_{2,27}, x_{3,27}, x_{4,27}$ | y_{27}
Datenpaar 3 : $x_{1,26}, x_{2,26}, x_{3,26}, x_{4,26}$ | y_{26}
Datenpaar i : ...
Datenpaar 26 : $x_{1,3}, x_{2,3}, x_{3,3}, x_{4,3}$ | y_2
Datenpaar 28 : $x_{1,2}, x_{2,2}, x_{3,2}, x_{4,2}$ | y_1

8.2.3. Kombinierte Modellierung

Die kombinierte Modellierung verhält sich wie die Zeitreihenmodellierung, da auch hier die Größe n des historischen Zeitfensters die Anzahl der formbaren Datenpaare beschränkt - auch hier ist die Anzahl der möglichen Datenpaare also $z - n$.

mit $n = 7$, $m = n + 1$ und x_1 und x_2 als Einflussvariablen
Datenpaar 1 : $y_{28}, y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22},$
$x_{1,28}, x_{1,27}, x_{1,26}, x_{1,25}, x_{1,24}, x_{1,23}, x_{1,22}, x_{1,21},$
$x_{2,28}, x_{2,27}, x_{2,26}, x_{2,25}, x_{2,24}, x_{2,23}, x_{2,22}, x_{2,21}$ | y_{21}
Datenpaar 2 : $y_{27}, y_{26}, y_{25}, y_{24}, y_{23}, y_{22}, y_{21},$
$x_{1,27}, x_{1,26}, x_{1,25}, x_{1,24}, x_{1,23}, x_{1,22}, x_{1,21}, x_{1,20},$
$x_{2,27}, x_{2,26}, x_{2,25}, x_{2,24}, x_{2,23}, x_{2,22}, x_{2,21}, x_{2,20}$ | y_{20}
Datenpaar i : ...
Datenpaar 21 : $y_8, y_7, y_6, y_5, y_4, y_3, y_2,$
$x_{1,8}, x_{1,7}, x_{1,6}, x_{1,5}, x_{1,4}, x_{1,3}, x_{1,2}, x_{1,1},$
$x_{2,8}, x_{2,7}, x_{2,6}, x_{2,5}, x_{2,4}, x_{2,3}, x_{2,2}, x_{2,1}$ | y_1

Bevor nun beschrieben werden kann, wie diese maximal formbare Menge an Datenpaaren auf die Trainings-, Validierungs- und Testsets aufgeteilt wird, werden die folgenden Abschnitte zeigen, welche Aufgaben die drei Phasen - Training, Validierung und Test - haben und wozu diese Datensets eingesetzt werden.

8.3. Trainingsphase

Aufgabe der Trainingsphase ist es, das KNN anhand historischer Daten so zu trainieren, dass es in der Lage ist zukünftige Daten vorherzusagen. Die historischen Daten werden dem Netz in Form der dargestellten Datenpaare präsentiert. Ziel ist es nun, die Gewichtsmatrix W eines KNN so zu verändern, dass der Ziel-Vektor jedes Trainingspaares zuverlässig angenähert werden kann. Dazu wird der Input-Vektor jedes Trainingspaares an das KNN angelegt, die Werte werden gemäß der Strategien und Konfigurationen in V und I durch das Netz propagiert und die Gewichtsmatrix W wird gemäß der Entscheidungen des Lernprozesses L verändert. In einer (Trainings-)Iteration werden diese Schritte für alle Trainingspaare eines Trainingssets durchlaufen. Am Ende jeder Iteration stehen also die aktuelle Gewichtsmatrix und der Abbildungsfehler zwischen gewünschtem und tatsächlichem Output zur Verfügung.

Das Stopkriterium S bestimmt, wie oft das Trainingsset durchlaufen wird. In der Praxis wird meist eine Anzahl an Iterationen festgelegt, die durchlaufen werden muss oder ein minimaler Abbildungsfehler, der erreicht werden soll. Im zweiten Fall wird das Trainingsset so oft durchlaufen, bis der Trainingsfehler unter den gesetzten Abbildungsfehler gesunken ist. Dies kann natürlich zu einer Endlositeration führen, sollte das KNN nicht in der Lage sein den Trainingsfehler auf das gewünschte Niveau zu senken. Deshalb wird auch hier eine maximale Anzahl an Iterationen festgelegt, bei der das Training abgebrochen wird, egal wie hoch der Trainingsfehler zu diesem Zeitpunkt sein sollte.

Die praxisnahe Literatur wie Sarle (2002) und Masters (1993) empfiehlt, die Trainingsphase pro KNN mehrmals mit unterschiedlichen Startgewichten zu durchlaufen. Hintergrund dieser Empfehlung ist, dass Gradientenabstiegsverfahren[3] die Entscheidung, wie und ob ein Gewicht verändert werden soll, anhand des aktuellen Gewichts treffen. Es gibt also günstige und ungünstige initiale Gewichte. Durch manche kommt ein KNN schneller oder überhaupt zum gewünschten Ergebnis und durch andere verfehlt es das globale Optimum gänzlich, da es durch die initialen Gewichte bereits so tief in einem lokalen Optimum 'feststeckt' und nicht mehr heraus kommt. Die empfohlene Vorgehensweise ist: *"Train the network using several (anywhere from 10 to 1000) different sets of random initial weights. For the operational network, you can then use the weights that produce the smallest training error"* (Sarle, 2002, o.S.).

8.4. Validierungsphase

"It would be foolhardy to train a network then immediately place it into service. Its competence must be evaluated first. This process is called validation. [...] Nerver underestimate the importance of validation, in many respects, proper validation is more important than proper training." (Masters, 1993, S. 10)

[3] Wozu in erster Linie das Verfahren der Backpropagation und viele ihrer Derivate und Verbesserungen zählen.

Die Validierung eines KNN ist die Überprüfung des Gelernten durch zuvor nicht präsentierte Daten. Dem Netz werden Datenpaare präsentiert, die strukturell und semantisch mit den Trainingspaaren übereinstimmen, aber Daten enthalten, die das Netz in der Trainingsphase nicht gesehen hat. Der Ziel-Vektor wird in der Validierungsphase nicht zur Optimierung, sondern zur Überprüfung der errechneten Werte herangezogen.

Der ermittelte Validierungsfehler (VE) ist also ein Indikator dafür, ob das Netz die vorhandenen Muster in den Trainingsdaten korrekt erkannt und soweit generalisiert hat, dass es auch auf unbekannten Daten zufriedenstellende Vorhersagen treffen kann. In der Validierungsphase werden keine Gewichte verändert, vielmehr wird auf Grundlage der bereits modifizierten Gewichte aus der Trainingsphase untersucht, ob das Netz eine Funktion gefunden hat, die sowohl bekannte als auch dem Netz unbekannte Daten abbilden kann (Rey, 2013, o.S.).

Im Folgenden möchte ich noch etwas näher auf die Empfehlung es Eingangs zitierten T. Masters eingehen, nämlicher der Validierung eines KNN eine größere Bedeutung beizumessen als dem Training.

8.4.1. Over- und Underfitting

Wie bereits mehrfach erwähnt, ist die große Stärke von KNN, Muster in den Daten zu erkennen und eine Gewichtung der neuronalen Verbindungen zu ermitteln, die in der Lage ist, diese Daten bestmöglich abzudecken (im englischen als *fitting* bezeichnet). Eine weitere Stärke ist die Fähigkeit Zusammenhänge zu erkennen und abbilden zu können die nicht linear sind. Im Folgenden soll nun anhand eines einfachen Beispiels gezeigt werden, dass eine der größten Herausforderungen beim Entwurf von KNN darin besteht, die richtige Balance dieser Fähigkeiten zu finden.

Die folgenden Beispiele und Erläuterungen sind an den Cousera-Kurs von Andrew Ng (Stanford University), Lecture 10 (Ng, 2012a) angelehnt.

Abbildung 8.4 (Seite 53) zeigt rein hypothetische Daten, die ein KNN erlernen bzw. vorhersagen soll. Die Beschriftung der Achsen mit Input und Output soll lediglich andeuten, dass bei einem gegebenen Input $\vec{x}$ die skizzierten Werte der optimale Output $\vec{y}$ wären. Die orange gezeichnete Linie stellt die Abbildungsfunktion dar, die das KNN ermittelt - die prognostizierten Werte also. Im Fall A werden die Daten underfittet, im Fall B overfittet und im Fall C werden sie ausreichend gut generalisiert, so dass sowohl Trainings- als auch Validierungsdaten gut getroffen werden.

Fall A stellt eine Situation dar, in der das zugrundeliegende Muster in den Trainingsdaten nicht genau genug erlernt wurde, so dass sich weder der Trainingfehler noch der Validierungsfehler in einem zufriedenstellenden Bereich bewegen. Es werden also weder die Trainingsdaten noch die Validierungsdaten ausreichend genau getroffen.

Ursache: Hier ist im Allgemeinen mindestens einer der folgenden zwei Gründe vorzufinden:

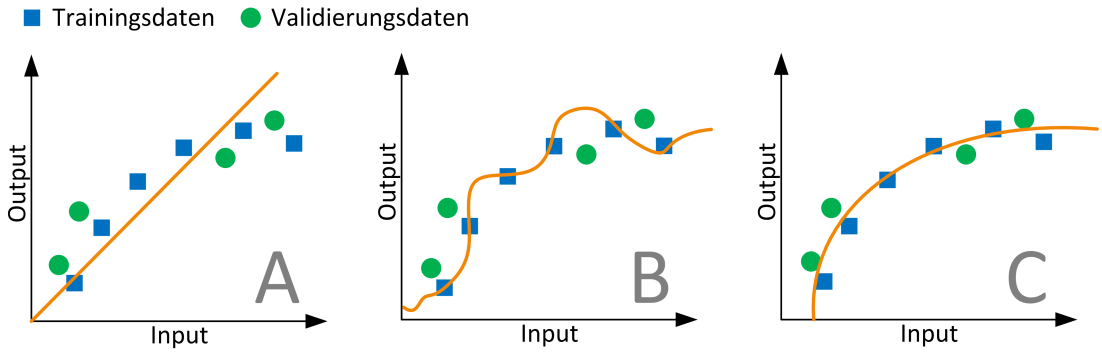


Abbildung 8.4.: A) Underfitting, B) Overfitting, C) "Genau richtig"

1. Die Inputwerte charakterisieren die Daten nicht ausreichend. Es ist also nicht möglich eine Abbildung von den gegebenen Inputwerten auf die gewünschten Outputwerte zu finden. Lösung: Ermittlung weiterer Input-Variablen, die mehr Informationen über die Outputwerte zur Verfügung stellen.
2. Die (Netz-)Komplexität ist für das zu lösende Problem unzureichend. Lösung: Die Netztopologie T erweitern und die Anzahl der Hidden-Neuronen erhöhen, Lernprozess L optimieren.

Fall B stellt eine Situation dar, in der das Muster in den Trainingsdaten zu genau erlernt wurde, so dass der Trainingsfehler besonders niedrig der Validierungsfehler aber besonders hoch ist. Die Trainingsdaten wurden besonders gut getroffen, die Validierungsdaten allerdings sehr unzureichend.

Ursache: Hier ist im Allgemeinen mindestens einer der folgenden zwei Gründe vorzufinden:

1. Das KNN hat zu viel Rechenleistung (im Allgemeinen zu viele Hidden-Neuronen oder gar Hidden-Schichten) im Vergleich zur Anzahl der Trainingspaare. Was sich im Extremfall darin bemerkbar macht, dass das KNN den gewünschten Output der Trainingsdaten einfach 'auswendig' lernt, statt Muster zu generalisieren. Lösung: Anzahl der Trainingsdaten erhöhen. Sollte dies nicht möglich sein, kann alternativ die Netztopologie T angepasst werden, indem die Anzahl der Hidden-Neuronen/ Schichten reduziert wird.
2. Es gibt Input-Variablen im Input-Vektor, die keine Abhängigkeit zum Output aufweisen, aber zufälligerweise die Trainingsdaten ganz genau treffen. Oder anders ausgedrückt: die Trainings- und/oder Validierungsdaten sind nicht repräsentativ

für die Domäne.[4] Lösung: Die meisten KNN-Frameworks bieten die Möglichkeit, die Entwicklung der Gewichte einzusehen. Sollte sich ergeben, dass ein Gewicht überproportionalen Einfluss auf das Netzergebnis hat, könnte dies ein Hinweis auf ein Overfitting sein. Eine fachlichere Herangehensweise wäre die Überprüfung der Trainings- und Validierungsdaten von einem Domänenexperten, um sicher zu gehen, dass es sich bei der benutzen Datenmenge um einen repräsentativen Ausschnitt der zu prognostizierenden Domäne handelt.

Fall C stellt eine Situation dar, in der das Muster in den Trainingsdaten genau genug ermittelt und generalisiert wurde, so dass der Validierungsfehler nur knapp über dem Trainingsfehler liegt und beide Datensets ausreichend gut getroffen werden.

Over-/Underfitting-Analysen

Bezug nehmend auf die im vorangegangenen Abschnitt gezeigten Problematiken stellt ebenfalls Andrew Ng in seinem Coursera-Kurs, Lectures 10 + 11 (Ng, 2012a,c) drei praktische Herangehensweisen vor, um sehr frühzeitig in einer KNN-Entwicklung zu erkennen, ob der gewählte Ansatz möglicherweise unter einem Over- oder Underfitting Problem leidet.

Analyse 1 wird in Abbildung 8.5 (Seite 55) dargestellt und versucht zu ermitteln, ob die gewählte Netzkomplexität zu einem Over- oder Underfitting führt, bzw. welches Setting am geeignetsten erscheint, um die Aufgabe optimal zu erfüllen. Mit Netzkomplexität ist in erster Linie die Netztopologie T gemeint. Je mehr Hidden-Neuronen (und ggf. Hidden-Schichten), desto höher die Netzkomplexität. Man berechnet und plottet also den Trainingsfehler (*TE*) und den Validierungsfehler (*VE*) des KNN nach jeder Komplexitätssteigerung und beobachtet das Verhalten der beiden Kurven.

Folgende Tabelle fasst diese Erkenntnisse zusammen:

Underfitting, bei zu geringer Netzkomplexität, mit:	*Overfitting*, bei zu hoher Netzkomplexität, mit:
TE = hoher Fehler $VE \approx TE$	*TE* = niedriger Fehler $VE \gg TE$

[4] Ein fast schon klassisches Beispiel, wird sehr unterhaltsam von Neil Fraser auf http://neil.fraser.name/writing/tank/ beschrieben. Bei einem US Amerikanischem Militärprojekt sollte ein KNN trainiert werden zu erkennen, ob sich Panzer auf einem Foto befinden oder nicht. Sowohl in der Trainings- als auch in der Validierungsphase lieferte das KNN perfekte Ergebnisse - alle Fotos wurden richtig klassifiziert. Das Pentagon gab dann jedoch noch einen unabhängigen Test mit neuen Fotos in Auftrag und die Antworten des KNN waren plötzlich komplett zufällig und ließen keinerlei Muster erkennen. Nach längerer Suche fand man die Ursache: die Fotos der Trainings- und Validierungsdaten enthielten alle ein unterscheidendes Merkmal: die Fotos ohne Panzer wurden an einem sonnigen Tag mit wolkenlosem blauen Himmel gemacht und die mit Panzer an einem grauen wolkenbehangenem Tag. Das KNN hat also nicht gelernt Panzer zu erkennen, sondern wolkenlose von bewölkten Tagen zu unterscheiden, was in dem gewählten Datenset aber zufällig mit dem gewünschten Zielwert übereinstimmte.

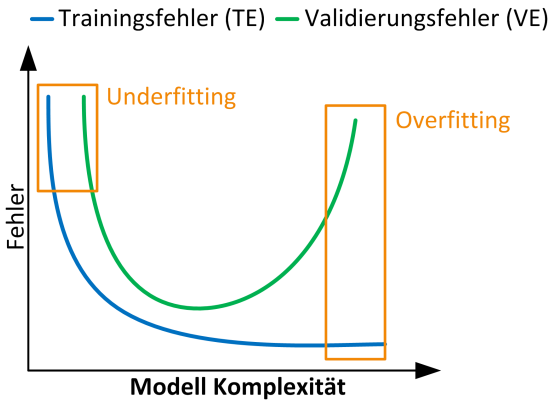


Abbildung 8.5.: Bei gleichbleibenden Trainings- und Validierungssets werden *TE* und *VE* nach jeder Komplexitätssteigerung des KNN berechnet und geplottet

Dieser Ansatz eignet sich nicht nur dazu herauszufinden, ob das gewählte Modell unter einem Over- oder Underfitting Problem leidet, sondern auch dazu die beste Anzahl an Hidden-Neuronen/Schichten zu ermitteln, wie T. Masters deutlich macht: *"The best approach to finding the optimal number of hidden neurons is time-consuming, but should always be followed for important tasks. Start with a number of neurons which is definitely too small. Train and [validate] the network, recording its performance. Then slightly increase the number of hidden neurons, and train and [validate] again. Repeat this until the error is acceptably small, or no significant improvement is noted, whichever comes first"* (Masters, 1993, S. 178).

Analyse 2 wird in Abbildung 8.6 (Seite 56) dargestellt. Man berechnet und plottet den Trainingsfehler (*TE*) und den Validierungsfehler (*VE*) des KNN nach jeder Trainingsiteration und beobachtet das Verhalten der beiden Kurven. Im Idealfall verhält sich *VE* so wie durch die gestrichelte Linie dargestellt, und konvergiert gegen das globale Minimum. Es gibt jedoch Verläufe bei denen ab einer bestimmten Anzahl an Trainingsiterationen der Validierungsfehler wieder ansteigt und die Prognosequalität des KNN somit mit zunehmender Anzahl an Trainingsiterationen sinkt.

Dieses Verhalten hat zu verschiedenen Herangehensweisen geführt. *"There are two schools of thought regarding the point at which training should be stopped"* (Kaastra und Boyd, 1996, S. 229). Die eine, initiiert von (Masters, 1993), geht davon aus, dass lokale Minima nur vermieden werden können, indem das Training eines KNN solange fortgesetzt wird, bis der Trainingsfehler konvergiert[5]. Die zweite Fraktion, u.a. repräsentiert durch (Tzafestas u. a., 1996) geht davon aus, dass ein KNN übertrainiert (engl. *overtrained*) werden kann und dass die maximale Anzahl an Iterationen ermittelt werden muss, die das Netzwerk laufen darf, bis es übertrainiert. *"Both schools of thought agree that generalization on the validation set is the ultimate goal"* (Kaastra und Boyd, 1996, S. 230). Sie unterscheiden sich aber in ihrer Interpretation der Ergebnisse und ihrer Lösungsansätze. Der Konvergenz-Ansatz geht davon aus, dass es kein Overtrai-

[5]Der Punkt ab dem der Trainingsfehler nicht mehr kleiner wird.

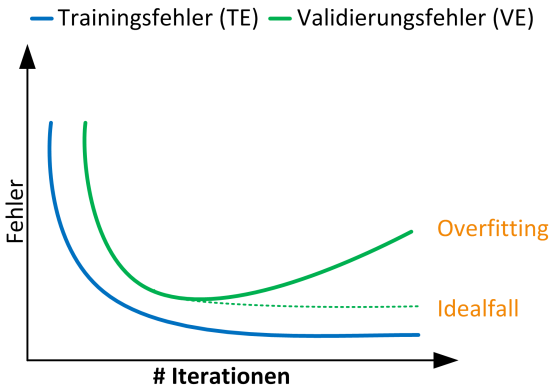


Abbildung 8.6.: Bei gleichbleibenden Trainings- und Validierungssets, werden *TE* und *VE* nach jeder Trainingsiteration des KNN berechnet und geplottet

ning, sondern nur Overfitting gibt und rät in diesem Fall, die Komplexität des KNN zu senken oder die Größe des Trainingssets zu erhöhen. Der Overtraining-Ansatz rät zur hier vorgestellten Analyse, um zu ermitteln, ab wie vielen Iterationen das Netz übertrainiert und somit nicht mehr ausreichend gut generalisiert. An diesem Punkt sollte das Training gestoppt werden. Das im vorangegangenen Abschnitt eingeführte Stopkriterium S würde somit auf die ermittelte maximale Anzahl an Iterationen gesetzt werden.

Analyse 3 Die Grundaussage, auf der die folgende Analyse basiert, ist, dass es einfacher ist, für wenige Trainingspaare eine Abbildungsfunktion zu finden, als für viele Trainingspaare - siehe dazu Abbildung 8.7 (Seite 56). Der Trainingsfehler steigt also mit zunehmender Anzahl an Trainingspaaren. Gleichzeitig wird sich aber der Validierungsfehler verringern, da die Abbildungsfunktion immer besser auf die Gesamtmenge der Daten passt.

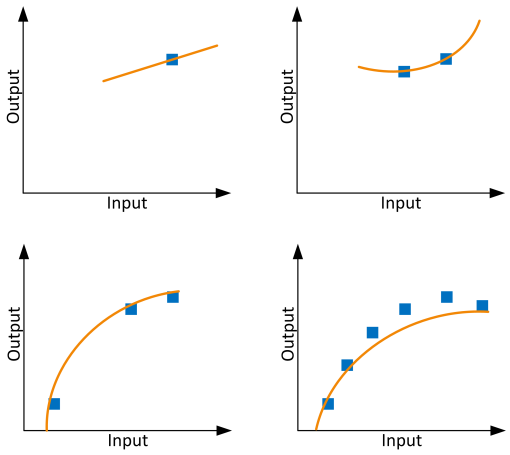


Abbildung 8.7.: Abbildungsfunktionen bei wenigen und bei vielen Trainingsdaten

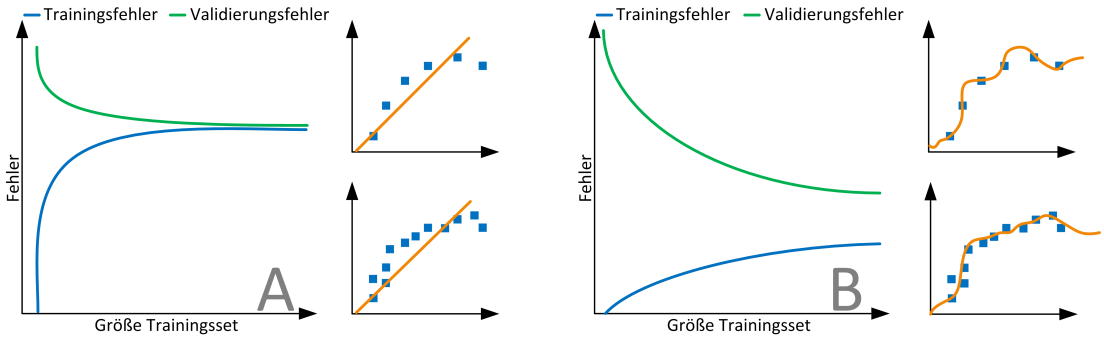


Abbildung 8.8.: Der Trainings- und Validierungsfehler wird für verschieden große Trainingsset geplottet. Fall A zeigt das Verhalten bei Underfitting und Fall B das Verhalten bei Overfitting

In der Analyse werden die zur Verfügung stehenden Trainingsdaten in verschieden große Untermengen eingeteilt (bswp. 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 Prozent (%) der gesamten Datenmenge). Mit jeder dieser Untermengen wird eine vollständige Trainings- und eine vollständige Validierungsphase durchlaufen und der jeweilige Trainings- und Validierungsfehler geplottet. Was zu den beiden Verhalten führen kann, die in Abbildung 8.8 (Seite 57) unter A und B veranschaulicht sind. Bei wenigen Trainingsdaten ist in beiden Fällen der Abstand zwischen *TE* und *VE* sehr hoch.

Im Fall A konvergieren beide Fehler jedoch sehr schnell und zwar auf einem sehr hohen Stand: mit $VE \approx TE$. Die beiden kleineren Graphen zur Rechten deuten an, dass es sich in diesem Fall um ein Underfitting handelt, um eine zu einfache Abbildungsfunktion, die nicht in der Lage ist, die Zielwerte zu treffen. Mehr Trainingsdaten würden diesen Fall *nicht* verbessern. Hier kann eine Verbesserung des Ergebnisses nur über eine Steigerung der Komplexität, mehr/andere Input-Variablen oder eine andere Netzarchitektur erlangt werden.

Fall B zeigt den Overfitting Fall, indem der große Abstand zwischen *VE* und *TE* sich viel langsamer verringert. Erst bei deutlicherer Vergrößerung der Trainingsdaten zeigen die beiden Kurven eine Tendenz zu konvergieren. Hier verdeutlichen die kleineren Graphen zur Rechten, dass eine Abbildungsfunktion die overfittet, durch mehr Trainingsdaten geglättet, also generalisiert werden kann.

Diese Analyse sollte lt. Andrew Ng (Ng, 2012a) bei jedem KNN-Entwurf vorgenommen werden, insbesondere jedoch wenn das Beschaffen weiterer Trainingsdaten kostspielig oder zeitaufwendig ist.

8.5. Testphase

Gründe für ein Validierungsset hat der letzte Abschnitt offensichtlich viele geliefert, doch auch die Begründung für ein weiteres Set, das Testset, liegt in den bereits gesehenen Argumentationen begründet.

Wenn man sich vorstellt, dass man mit einer Menge m an verschieden modellierten KNN startet, die alle antreten, um das gegebene Problem zu lösen, dann können evtl. schon nach der Trainingsphase einige dieser Modelle aussortiert werden, weil sie nicht in der Lage sind, einen annähernd zufriedenstellenden Trainingsfehler zu erreichen. Spätestens nach der Validierungsphase werden weitere KNN aus m herausfallen. Übrig bleibt in der Regel ein KNN M, das sowohl auf den Trainingsdaten als auch auf den Validierungsdaten am besten abgeschnitten hat. Derselbe Grund, weshalb man Modelle mit geringem Trainingsfehler durch ein Validierungsset überprüft, greift nun auch für die Notwendigkeit eines unabhängigen Testsets. Das Modell M ist auf zwei konkrete Untermengen der Domäne optimiert und sollte vor der Produktivsetzung durch eine weitere, bisher vollkommen unbekannte Untermenge, überprüft werden. Wenn man an das in Abschnitt 8.4.1 erwähnte Beispiel der Panzererkennung auf Fotos denkt, leuchtet diese Argumentation sehr schnell ein.

Dieses Vorgehen ist bereits ratsam, wenn man die Zeit und Ressourcen hat, eine große Bandbreite aller möglichen Modelle zu implementieren und diese zu trainieren, zu validieren und zu testen, ohne zwischenzeitlich Änderungen an ihnen vorzunehmen. Es ist jedoch erst recht zu empfehlen, wenn man mit einer sehr kleinen Modellmenge (1-3 Modelle) beginnt und diese iterativ optimiert. In Abbildung 8.9 (Seite 59) wird ein sehr typisches Vorgehen skizziert, in dem eine kleine Modellmenge sukzessive modifiziert und reduziert wird. In Menge $**m$ befindet sich am Ende ein Modell, dass nicht nur auf das Trainingsset optimiert wurde, sondern bei dem das Validierungsset ebenfalls unmittelbar in die Optimierung eingeflossen ist. Somit bildet das Testset die einzig objektive Überprüfungsmöglichkeit des finalen Modells.

Es sei darauf hingewiesen, dass viele Autoren vor diesem Vorgehen warnen (Kaastra und Boyd, 1996; Callan, 2003). Beziehungsweise merken sie an, dass die Notwendigkeit, weitere Optimierungen nach der Validierung vornehmen zu müssen, ein Indiz dafür ist, dass das Validierungsset in das Trainingsset integriert werden sollte und ein neues (unbekanntes) Validierungsset beschafft werden muss. Denn offensichtlich enthält das Validierungsset Informationen, die dem KNN während des Trainings nicht zur Verfügung standen (Masters, 1993). Gleichzeitig wird aber erwähnt, dass das Beschaffen neuer Daten in der Realität oft mit einem hohen zeitlichen Aufwand oder nicht unerheblichen Kosten verbunden ist und das beschriebene Vorgehen somit oft unausweichlich (Braun u. a., 1997; Ng, 2012b). Umso größer ist jedoch die Notwendigkeit eines repräsentativen Testsets.

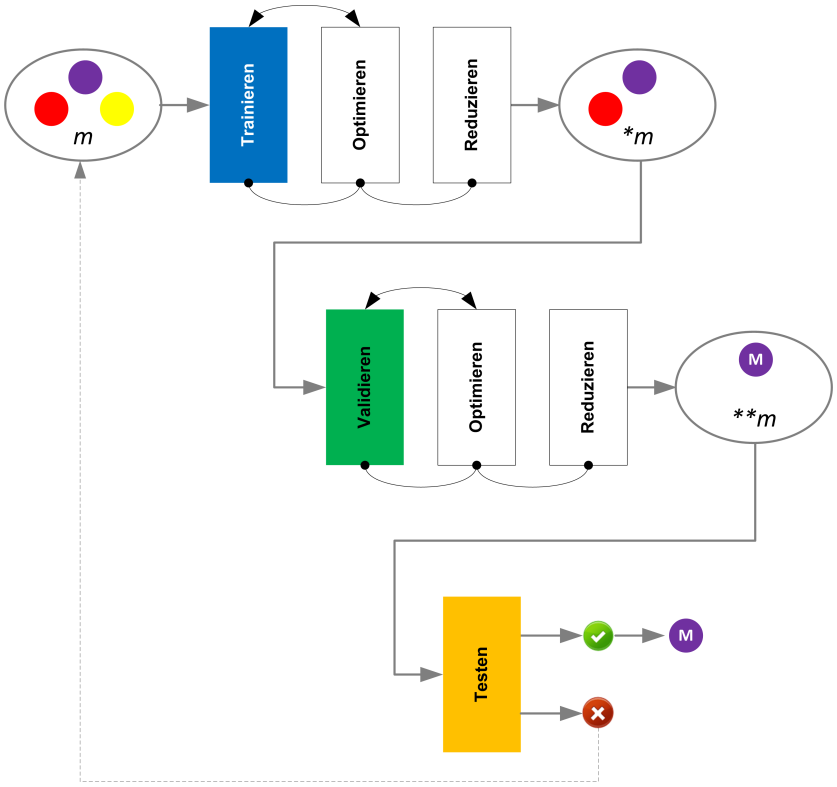


Abbildung 8.9.: Iteratives Optimieren und Reduzieren einer initialen Modellmenge

Unabhängig vom vertretenen Ansatz können folgende Zuständigkeiten der einzelnen Datensets zusammengefasst werden:

- **Trainingsdaten**: Die Trainingsdaten dienen der Findung einer initialen Konfiguration und der Parametrisierung der Verbindungsgewichte - also primär dem Lernen. Mit Hilfe der Trainingsdaten wird eine initiale Modellmenge entwickelt.
- **Validierungsdaten**: Die Validierungsdaten dienen der Optimierung der initialen Konfiguration, im Allgemeinen durch Veränderungen an der Netztopologie und Anpassungen des Trainingsalgorithmus. Mit Hilfe der Validierungsdaten wird also das beste Modell bzw. die besten Modellparameter bestimmt.
- **Testdaten**: Die Testdaten dienen der abschließenden Abschätzung des Prognosefehlers. Mit Hilfe der Testdaten wird das Modell überprüft, das während der Validierung am besten abgeschnitten hat.

8.6. Erstellung von Trainings-, Validierungs- und Testsets

Die Menge der zur Verfügung stehenden Datenpaare wird nun nach der genannten prozentualen Verteilung in Trainings-, Validierungs- und Testsets zusammengefasst - jedes Datenset besteht also aus einer Teilmenge aller Datenpaare. Der Schnitt dieser Teilmengen ist allerdings alles andere als trivial und wird in der Literatur unterschiedlich behandelt.

Der Hintergrund der Problematik ist, dass die Prognosegüte des benutzten KNN, und somit seine Fähigkeit zu Generalisieren, anhand des Validierungs- bzw. Testfehlers bestimmt wird. Wenn aber beispielsweise ein KNN zur Absatzprognose von Regenschirmen trainiert werden soll und das Trainingsset nur Daten von regenfreien Tagen enthält, dann wird das KNN auf einem Validierungsset, das viele Regentage enthält, schlecht abschneiden. Das Gleiche gilt, wenn eine Abhängigkeit zu den Wochentagen vermutet wird und das Trainingsset beispielsweise keine oder kaum Mittwoche enthält, das Validierungs- und/oder Testset hingegen viele. *"It is critical to have both the training and test sets representative of the population or underlying mechanism"* (Zhang u. a., 1998, S. 50).

Doch wie erreicht man eine repräsentative Aufteilung der vorhandenen Daten, ohne dass die Daten dadurch eine ungewollte Tendenz (engl. *bias*) entwickeln? Die Literatur hat darauf zwei Antworten:

- **Sequentielle Verteilung**: Aus allen Datenpaaren werden die ersten x% dem Trainingsset zugeordnet, die nächsten y% dem Valisierungsset und die letzten z% dem Testset. Bei diesem Vorgehen besteht das Trainingsset also immer aus den ältesten zur Verfügung stehenden Daten und das Testset aus den neuesten.
- **Zufällige Verteilung**: Die vorhandenen Datenpaare werden mit einem Zufallsalgorithmus gemischt und dann prozentual auf Trainings-, Validierungs- und Testset verteilt.

Puma-Villanueva et.al. haben diese beiden Ansätze in (Puma-Villanueva u. a., 2006) miteinander verglichen und kamen zu dem Ergebnis, dass die seltener eingesetzte zufällige Verteilung eindeutige Vorteile gegenüber der sequentiellen hat. *"The obtained results indicate with high confidence that the rarely adopted random partition [...] overcomes the contestants in the whole set of experiments"* (Puma-Villanueva u. a., 2006, S. 4740). Bei der Zusammenstellung des Testsets berufen sich allerdings sowohl Puma-Villanueva u. a. (2006) als auch Zhang u. a. (1998) auf eine Konvention aus dem M3-Wettbewerb, die festlegt, dass das Testset bei monatlichen, vierteljährlichen und jährlichen Prognosen respektive aus den letzten 18, 8, 6 Punkten der Zeitreihen bestehen sollte. Die Idee ist, dass für den abschließenden Test eines KNN die neuesten zur Verfügung stehenden Daten verwendet werden sollten, unter der Annahme, dass diese den Daten im Produktiveinsatz auf Grund ihrer zeitlichen Nähe am ähnlichsten sind.

Abbildung 8.10 verdeutlicht den Unterschied der beiden Verteilungen und bezieht die Empfehlung bezüglich des Testsets mit ein. Der linke Teil repräsentiert die Menge aller zur Verfügung stehenden Datenpaare. In der Mitte ist die sequentielle Verteilung abgebildet und rechts die zufällige. Die zufällige Verteilung mit festem Testset wird in dieser Arbeit Anwendung finden.

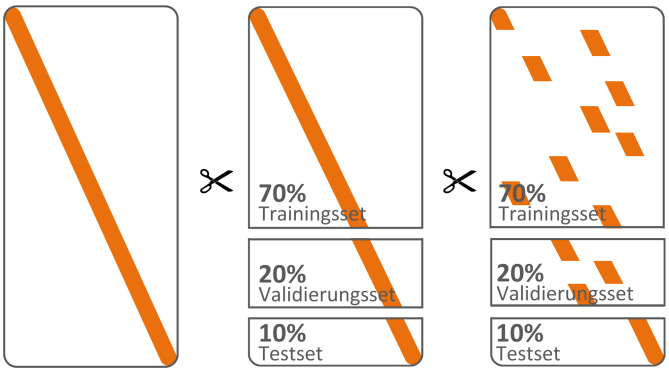


Abbildung 8.10.: Sequentieller (mitte) und zufälliger (rechts) Schnitt einer Menge von Datenpaaren (links)

Teil IV.

Praktische Umsetzung

9. Wahl des Frameworks

Davon ausgehend, dass die Einstiegshürde sowohl für mich als auch für eine mögliche Integration in die bestehende Java-Softwarelösung geringer sein würde, war mein Bestreben, ein **Java**-Framework für KNN zu finden. Die drei bekanntesten Open Source Frameworks für Java sind: Neuroph[1], JOONE[2] und Encog[3], die in einem aufschlussreichen Benchmark in (taheretaheri, 2010) gegenüber gestellt werden. Mit dem Ergebnis, dass Encog sowohl erheblich weniger Iterationen braucht, um die gestellten Aufgaben zu lösen, als seine Mitbewerber und die einzelnen Iterationen auch noch deutlich performanter implementiert sind. Für den zugrundeliegenden Anwendungsfall, eine Vorhersage Geothermischer-Daten, schnitten die genannten Frameworks folgendermaßen ab:

Framework	Dauer der Vorhersage (in Sekunden)
Encog	3,1280
JOONE	17,8510
Neuroph	39,7450

9.1. JOONE

JOONE habe ich nicht getestet, da die Entwicklung des Frameworks eingestellt wurde und die API als sehr komplex und zu offen beschrieben wurde (taheretaheri, 2010)[4]. Offen im Sinne von fehlendem Information-Hiding. Als Benutzer des Frameworks ist man gezwungen sich mit internen Implementierungsdetails zu beschäftigen, die sowohl die Verwendung als auch das Verständnis deutlich erschweren. Sowohl der fehlende Support als auch die unnötige Komplexität der API sind ausreichende Gründe, das Framework nicht in die engere Wahl zu ziehen.

[1] http://neuroph.sourceforge.net

[2] http://sourceforge.net/projects/joone

[3] http://www.heatonresearch.com/encog

[4] Zur Verdeutlichung das wörtliche Zitat des Benchmarkers: "The last two frameworks [Encog and Neuroph] that we examined had fairly simple APIs that make it easy on the programmer to get the neural network created. Not so with JOONE. JOONE's API is a beast. It does not really hide much, and is very difficult to get working. Of all three examples, this one took me the most time. It is also the longest." (taheretaheri, 2010)

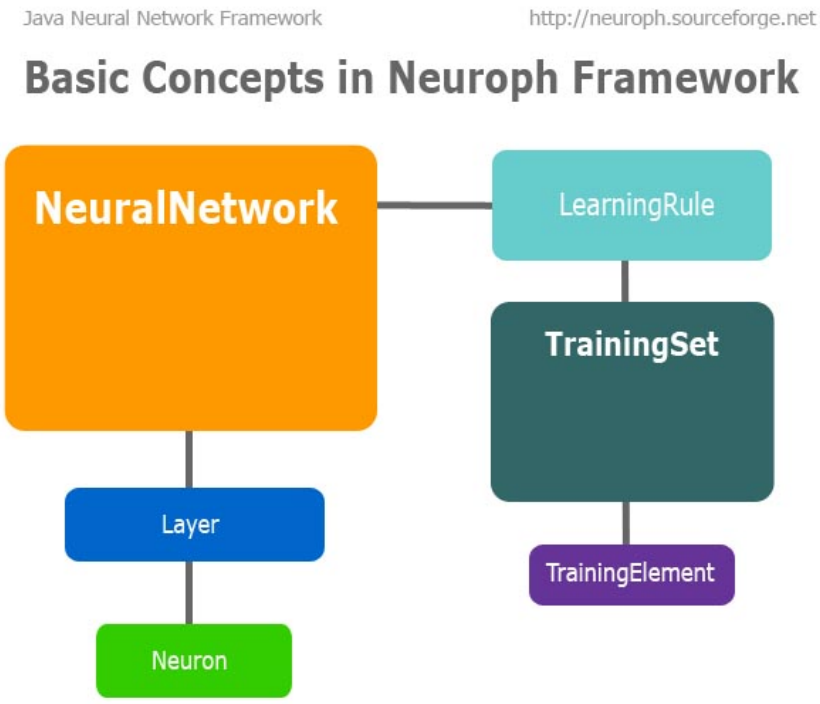


Abbildung 9.1.: Grundkonzepte des Open Source KNN Java-Frameworks Neuroph

9.2. Neuroph

Neuroph ist das erste Java KNN-Framework das ich getestet habe. Es ist ein leichtgewichtiges Framework mit einer sehr verständlichen und gut gekapselten API. Es erhebt nicht den Anspruch Werkzeuge für maschinelles Lernen im Allgemeinen zur Verfügung zu stellen, sondern bietet Funktionalität für die gängigsten KNN-Architekturen und ihre häufigsten Ausprägungen. Die Konzentration auf die gängigsten KNN-Konzepte halten das Framework erfreulich schlank, was den Einstieg in die KNN-Implementierung sehr erleichtert. Abbildung 9.1 (Seite 64) ist der Webseite des Entwicklerteams entnommen und verdeutlicht die übersichtliche Architektur. Die Implementierung eines KNN mit Neuroph erfordert lediglich das Verständnis der Grundkonzepte KNN, der Rest ist vorbildlich gekapselt, was auch die Framework-Implementierung selbst sehr nachvollziehbar macht.

Die Dokumentation besteht einerseits aus einer ausreichend genauen API-Dokumentation in Form von Javadocs und, zum größten Teil von Studenten erstellten, Tutorials und Anwendungsbeispielen. Diese Art der Dokumentation ist ausreichend, wenn das Ziel einfache KNN-Implementierungen sind, beispielsweise zu Übungs- oder Schulungszwecken. Sobald die Anforderungen jedoch umfangreicher werden, sind sie nicht komplex genug, um einen Großteil der entstehenden Fragen beantworten zu können.

9.3. Encog

Die mangelnde Dokumentation und die schlechten Benchmark-Ergebnisse von Neuroph haben mich bewogen ein weiteres Framework zu testen. *"Encog is a machine learning framework for Java and .NET. Initially, Encog was created to support only neural networks. Later versions of Encog expanded more into general machine learning"* (Heaton, 2011). Die Ausrichtung, ein Framework für maschinelles Lernen im Allgemeinen zu sein, macht Encog einerseits zu einem mächtigeren Framework im Vergleich zu Neuroph, aber gleichzeitig ist es dadurch deutlich schwergewichtiger und der Einstieg fällt schwerer. Encog verteilt 316 Klassen auf 75 Packages, im Gegensatz dazu implementiert Neuroph 124 Klassen in 17 Packages, was ein weiterer Indiz für die höhere Komplexität des Encog-Frameworks ist.

Die höhere Einstiegshürde bei Encog wird allerdings durch mehrere Vorteile aufgewogen:

- *Umfangreiche Dokumentation*: Zusätzlich zur obligatorischen API-Dokumentation als Javadocs existiert ein Wiki, eine Online-Forum, Beispielimplementierungen und zwei Bücher (Heaton (2011), Heaton (2008)), die sowohl eine umfangreichen theoretische als auch praktische Einführung in Encog und KNN im Allgemeinen liefern.
- *Performante Implementierung*: Encog stellt nicht nur die performantesten Lernalgorithmen zur Verfügung, sondern hat diese auch sehr performant implementiert, so dass Encog im vorgestellten Benchmark Neuroph um Größenordnungen übertroffen hat.
- *Umfangreiche Utility-Methoden*: Wie in den folgenden Kapiteln noch gezeigt werden wird, fließt ein Großteil des Arbeitsaufwandes bei der Implementierung KNN in die Vorverarbeitung der Daten. Datensets müssen erzeugt, normalisiert, zufällig verteilt und aufgeteilt werden. Encog liefert für diese Zwecke Methoden, die diesen Prozess vereinfachen und automatisieren.
- *Nützliche Workbench*: Sowohl Encog als auch Neuroph liefern eine sogenannte Workbench (also eine GUI-Client für das Framework). Die Encog-Workbench hat sich im Vergleich zur Neuroph-Workbench (die den gesamten Umfang der API nicht abbilden kann) als sehr hilfreich erwiesen, vor allem die graphische Auswertung von Trainings-, Validierungs- und Testdurchläufen.

Die grundlegende Bewertung von Encog fällt positiv aus und führte dazu, dass Encog im Rahmen dieser Arbeit eingesetzt wurde.

10. Analyse der Problemstellung

Ziel der Analyse der gegebenen Problemstellung ist die Beschreibung von Datenbasis und Prognosegegenstand, die Ermittlung des Algorithmus zur Bedarfsannäherung für historische Vergleichstage und die Gewährleistung der Vergleichbarkeit der Bedarfsprognosen des neuronalen und bestehenden Ansatzes.

10.1. Beschreibung von Datenbasis und Prognosegegenstand

Während der Analyse- und Entwicklungsphase standen im Umfang eingeschränkte aber inhaltlich vollständige Daten zur Verfügung:

Parameter	**Wert**
Anzahl prognoserelevanter Artikel	16
Anzahl Filialen	60
Sitz der Filialen	Tschechische Republik
Zeitraum der Daten	27.02 - 24.06.2012 (= 117 Tage)

Der vorliegende Anwendungsfall sieht vor, für jeden Artikel in jeder Filiale eine Bedarfsprognose zu ermitteln, was durch verschiedene Modellierungen und Implementierungen erreicht werden kann. Es könnte für jeden Artikel in jeder Filiale ein eigenes KNN entwickelt und trainiert werden, was bei den vorhandenen Daten bereits 960 KNN zu Folge hätte. Das andere Extrem wäre ein KNN mit allen Artikeln und allen Filialen zu trainieren. Die Ansätze, die sowohl die Anzahl der benötigten KNN als auch die Datenmenge in Grenzen halten, sind:

A) 60 KNN, um für jeweils eine Filiale den Bedarf aller dort verkauften Artikel zu prognostizieren

B) 16 KNN, um für jeweils einen Artikel den Bedarfs in allen Filiale zu prognostizieren.

Ich habe mich für Variante **B** entschieden, in der Annahme, dass die Gemeinsamkeiten für einen Artikel filialübergreifend größer sind, als die Gemeinsamkeiten für eine Filiale artikelübergreifend. Mit anderen Worten: der Verkauf desselben Artikels wird in unterschiedlichen Filialen durch ähnliche Faktoren motiviert. Der Verkauf unterschiedlicher Artikel in einer Filiale kann hingegen durch viele verschiedene Faktoren begründet sein.

Von den 16 möglichen Artikeln wurde ein Artikel ausgewählt, für den der Bedarf im Rahmen dieser Arbeit prognostiziert werden soll. Es handelt sich dabei um **Laugenbrötchen**, für die es die meisten historischen Daten in der zur Verfügung stehenden Datenbasis gibt.

Die Anzahl der einbezogenen Filialen wurde auf Grund einer Vereinfachung in der Implementierung reduziert. Das folgende Kapitel wird auf die konkrete Modellierung des Input-Vektors und der in Frage kommenden Einflussfaktoren eingehen, hier sei lediglich vorweggenommen, dass Wettereinflüsse eine Rolle bei der Prognose spielen werden. Um die Implementierung für den Rahmen einer Bachelorarbeit überschaubar zu halten, wurden historische Wetterdaten nur für Prag ermittelt und nicht für jede der 60 Filialen individuell. Das hat zur Folge, dass von den 60 maximal möglichen Filialen 24 Filialen ausgewählt wurden, die sich in und um Prag befinden.

Die folgende Tabelle fasst die Datenbasis zusammen:

Parameter	**Wert**
Zu prognostizierender Artikel	Laugenbrötchen
Zeitraum der Daten	27.02 - 24.06.2012
Anzahl Tage	117
Anzahl verkaufsoffene Tage	114
Anzahl Artikel	1
Anzahl Filialen	24
Anzahl Datensätze	114*24 = 2736

Tabelle 10.1.: Datenbasis

Aufgrund der Tatsache, dass Supermärkte in Tschechien auch an Sonntagen geöffnet haben und in den Zeitraum der Daten lediglich 3 gesetzliche Feiertage fallen, bleiben von den 117 Tagen, 114 verkaufsoffene übrig.

10.2. Algorithmus zur Bedarfsannäherung

Kapitel 4 hat die Eigenschaften von Bedarfsprognosen bereits erläutert, insbesondere ihre Abgrenzung zu Absatzprognosen und der damit verbundenen Notwendigkeit, einen Algorithmus zu entwickeln, der den historischen Bedarf eines Produktes annähern kann. Ein KNN das den Bedarf von Laugenbrötchen vorhersagen soll, kann nicht trainieren werden, wenn der Bedarf der Vergangenheit nicht bekannt ist. Selbst wenn die Prognose als kausale Prognose modelliert wird und der historische Bedarf somit nicht Bestandteil des Input-Vektors ist, so ist ein überwachter Lernalgorithmus nicht in der Lage zu lernen, wenn der Zielwert der Prognose nicht bekannt ist.

Das bestehende System implementiert einen Algorithmus zur Annäherung des Bedarfs von Backwaren, der sich beim Kunden über Jahre hinweg entwickelt und bewährt hat. Aus lizenz-

rechtlichen Gründen darf dieser hier nicht im Detail dargestellt werden, jedoch die fachlichen Hintergründe:

- Für jeden Artikel ist eine erwartete letzte Abverkaufszeit (eLAZ) vorgegeben.
- Der Verkaufspreis eines Artikel kann im Laufe eines Verkaufstages reduziert (diskontiert) werden, wenn der Filialleiter feststellt, dass zu einer gegebenen Zeit noch zu viele Einheiten eines Artikels vorhanden sind, in der Annahme, dass dies den Abverkauf steigern wird (diskontierter Abverkauf (DA)).
- Am Ende des Verkaufstages liefern die Scannerkassen sowohl die Anzahl der Abverkäufe (A) für jeden Artikel, als auch die letzte Abverkaufszeit (LAZ) des Artikels.

Die grobe Idee, um den Bedarf über *A* anzunähern ist:

- bei $LAZ < eLAZ$, je nach Höhe der Abweichung einen prozentualen Anteil auf *A* zu addieren
- bei $DA > 0$ einen prozentualen Anteil von *A* zu subtrahieren

In den folgenden Abschnitten wird der hier angedeutete Algorithmus als *Algorithmus zur Bedarfsannäherung (BA)* bezeichnet. Da es sich bei der erwarteten letzten Abverkaufszeit *eLAZ* um eine feste Größe handelt, wird im weiteren davon ausgegangen, dass *eLAZ* keine Input-Variable für *BA* ist, im Gegensatz zu den Variablen *A*, *DA* und *LAZ*.

10.3. Vergleichbarkeit der angestrebten mit der bestehenden Lösung

Ein Ziel dieser Arbeit ist, zu überprüfen, ob ein KNN bessere Bedarfsprognosen für Backwaren ermitteln kann, als die bisherige Lösung. Um sicher zu stellen, dass nicht die sprichwörtlichen Äpfel mit Birnen verglichen werden, muss sowohl die Funktionsweise der bestehenden Implementierung untersucht und verstanden werden als auch die Anforderungen, die eine Implementierung mit KNN an die Vergleichbarkeit stellt.

10.3.1. Übersicht über bestehende Architektur

Abbildung 10.1 (Seite 69) zeigt den groben Ablauf der bisherigen Implementierung[1]. Die Komponente wird zur zukünftigen Referenz, als Komponente **1** bezeichnet. Weiß unterlegte Elemente werden im Verlauf dieser Arbeit wieder aufgegriffen, weil sie für den Vergleich zwischen den beiden Lösungen von Bedeutung sind. Die grau unterlegten Elemente dienen lediglich der Verdeutlichung der Abläufe.

[1] Da es sich bei dem hier untersuchten System um eine proprietäre Softwarelösung handelt, kann die Implementierung verständlicherweise nur in Ansätzen dargestellt werden.

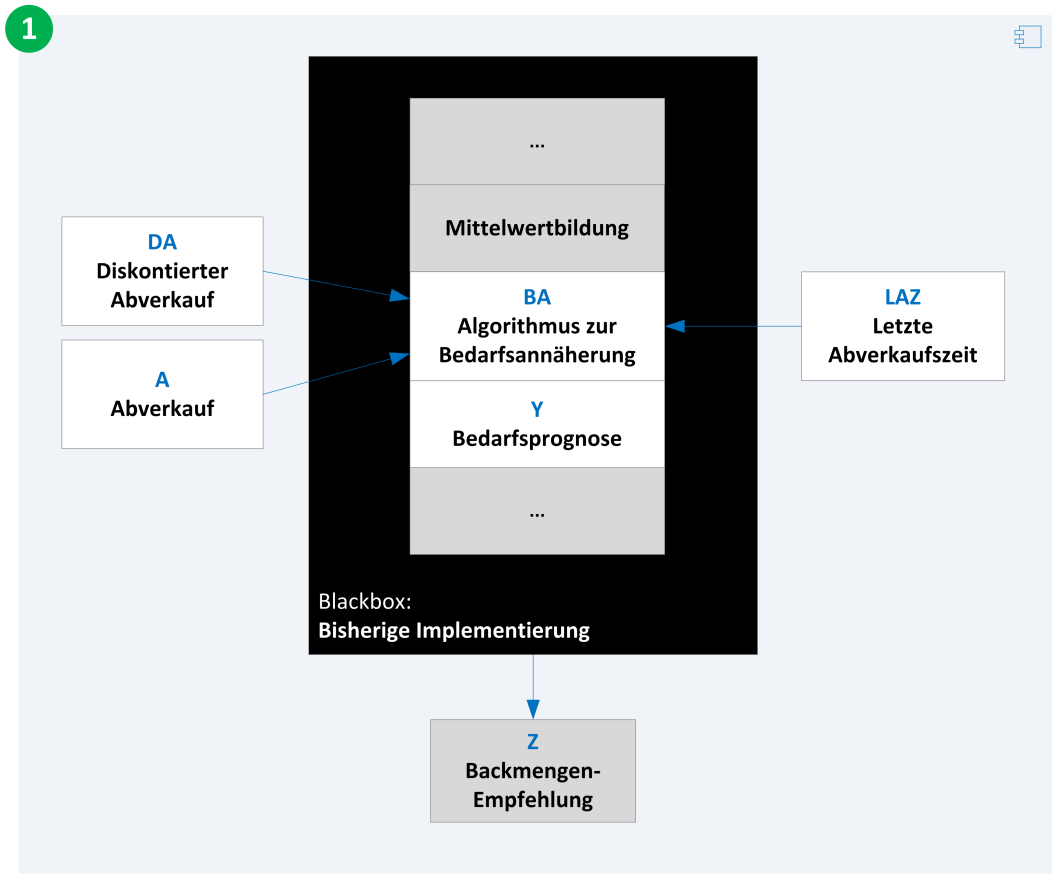


Abbildung 10.1.: Bisherige Implementierung: Komponente zur Ermittlung von Backmengenempfehlungen

Die zugrundeliegende Berechnungslogik ist hier als Blackbox dargestellt und besteht u.a. aus einem Berechnungsstack der nacheinander abgearbeitet wird und an dessen Ende eine Backmengenempfehlung steht [2]. Grundlage der Bedarfsprognose *Y* ist die in Abschnitt 1.1 skizzierte Mittelwertbildung über die Abverkaufszahlen der historischen Vergleichstage. Um die daraus resultierende Absatzprognose zu einer Bedarfsprognose zu erweitern, wird der Mittelwert mit Hilfe des Algorithmus *BA* entsprechend angepasst.

[2] Wie bereits erwähnt, ist die Generierung von Backmengenempfehlungen nicht Gegenstand dieser Arbeit, wird hier jedoch aufgeführt, um auf die unterschiedliche Zielsetzung der bisherigen und der, im nächsten Abschnitt beschriebenen, neuronalen Implementierung hinzuweisen.

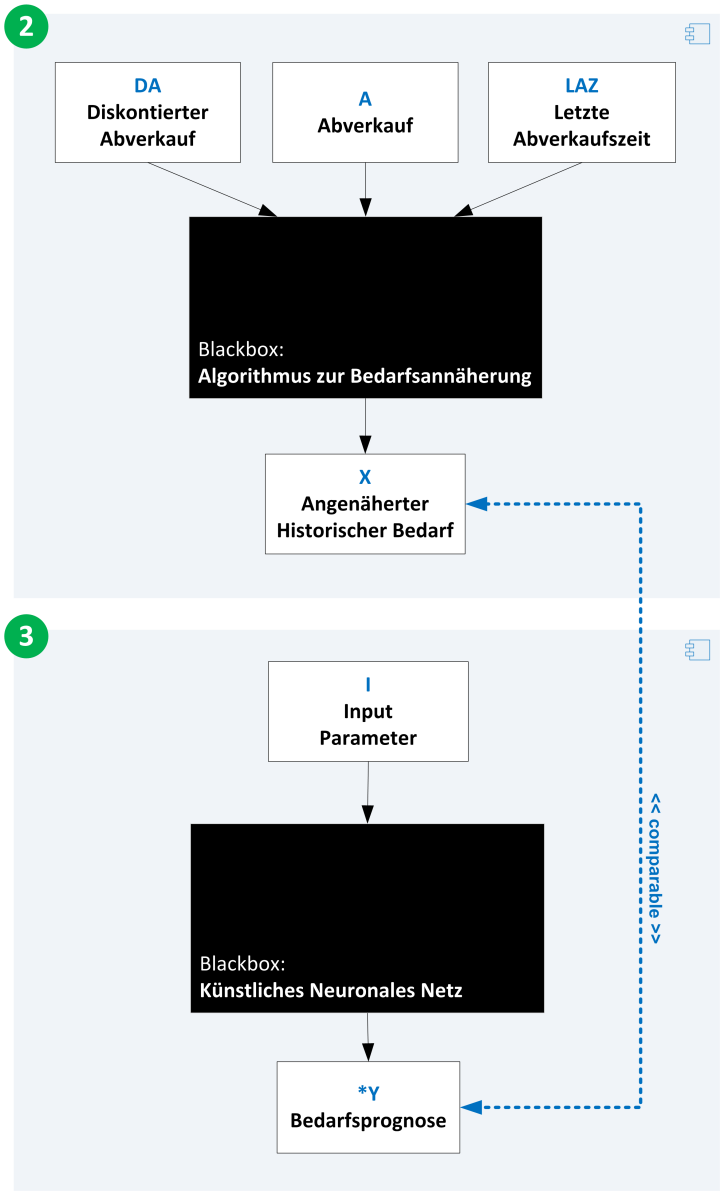


Abbildung 10.2.: Implementierung mit KNN: Komponenten zur Ermittlung und Validierung von Bedarfsprognosen

10.3.2. Übersicht über KNN-Architektur

Die Umsetzung der Bedarfsprognostizierung mit KNN, sieht die beiden in Abbildung 10.2 (Seite 70) skizzierten Komponenten vor:

- Komponente **2** besteht aus dem Algorithmus zur Bedarfsannäherung *BA* und kann für

alle $t \leq$ *heute* den tatsächlichen historischen Bedarf *X* aus den Eingabegrößen *A*, *DA* und *LAZ* annähern.

- Die Komponente **3** steht für das zu entwickelnde KNN, das aus einem gegebenen Input *I* den Bedarf $*Y$ generiert. Während der Trainings-, Validierungs- und Testphase des KNN stellt *X* den idealen Zielwert für das überwachte Lernen zur Verfügung. Im Produktivbetrieb, also bei Prognosen in die unbekannte Zukunft, ist der Vergleich erst am Folgetag möglich, wenn der Abverkauf (*A*), der diskontierte Abverkauf (*DA*) und die letzte Abverkaufszeit (*LAZ*) bekannt sind.

10.3.3. Vergleichbarkeit

Ziel ist es nun die Bedarfsprognose *Y* die das bisherige System in Komponente **1** ermittelt, mit der Bedarfsprognose $*Y$ aus der KNN-Komponente **3** zu vergleichen. Voraussetzung dafür ist, dass *BA* und *Y* aus dem Berechnungsstack in Komponente **1** isoliert werden. In Bezug auf den Algorithmus zur Bedarfsannäherung bedeutet dies, dass die Berechnungslogik zur Annäherung des Bedarfs in eine eigenständige Komponente **2** ausgelagert werden muss, die von beiden Implementierungen benutzt werden kann. Die Bedarfsprognose *Y* existiert aktuell nicht außerhalb des Berechnungsstacks und muss persistiert werden.

Abbildung 10.3 (Seite 72) zeigt die angesprochene Isolierung der benötigten Komponenten aus der bisherigen Implementierung und die daraus resultierende Vergleichbarkeit mit der hier angestrebten Lösung. Der Algorithmus *BA* steht als eigenständige Komponente **2** zur Verfügung, so dass die Bedarfsprognosen beider Implementierungen gegen den selben Wert *X* validiert werden können. Außerdem wurde der Rest der bisherigen Berechnungslogik in zwei Komponenten gesplittet: Komponente **4** berechnet nun ausschließlich die Bedarfsprognose, so dass diese isoliert von den Aspekten der Backmengenempfehlung betrachtet werden kann. Komponente **5** rechnet Bedarfsprognosen, unabhängig von ihrer Ermittlung, zu Backmengenempfehlungen um.

Die Berechnungslogik des bisherigen Systems, wie eingangs in Komponente **1** dargestellt, muss also refaktoriert werden und auf 3 Komponenten (**2**, **4** und **5**) aufgeteilt werden, um die Vergleichbarkeit mit einer neuronalen Lösung zu ermöglichen.

10.3.4. Definition einer Teillösung

Die in Abbildung 10.3 (Seite 72) skizzierte Vergleichbarkeit der beiden Lösungsansätze setzt eine Refaktorierung der bisherigen Implementierung voraus, die den praktischen Rahmen dieser Arbeit gesprengt und gleichzeitig die eigentliche Zielsetzung verfehlt hätte. Die Isolierung des Algorithmus zur Bedarfsannäherung *BA* und die Persistierung der Bedarfsprognose *Y* in der benötigten Form sollte zukünftig durch das Projektteam von *mgm technology partners* umgesetzt werden.

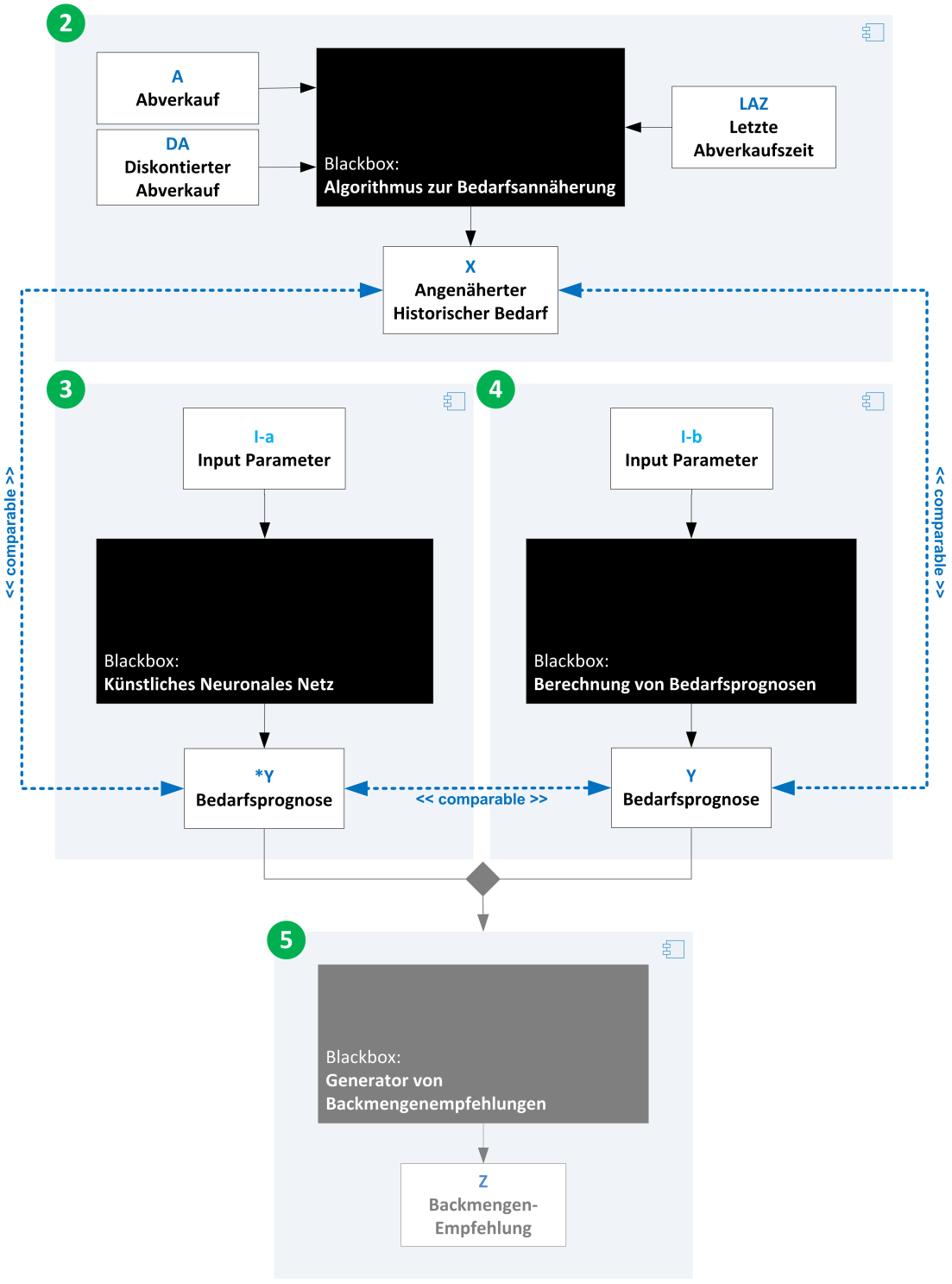


Abbildung 10.3.: Refaktorierung der bisherigen Implementierung zur Gewährleistung der Vergleichbarkeit beider Ansätze

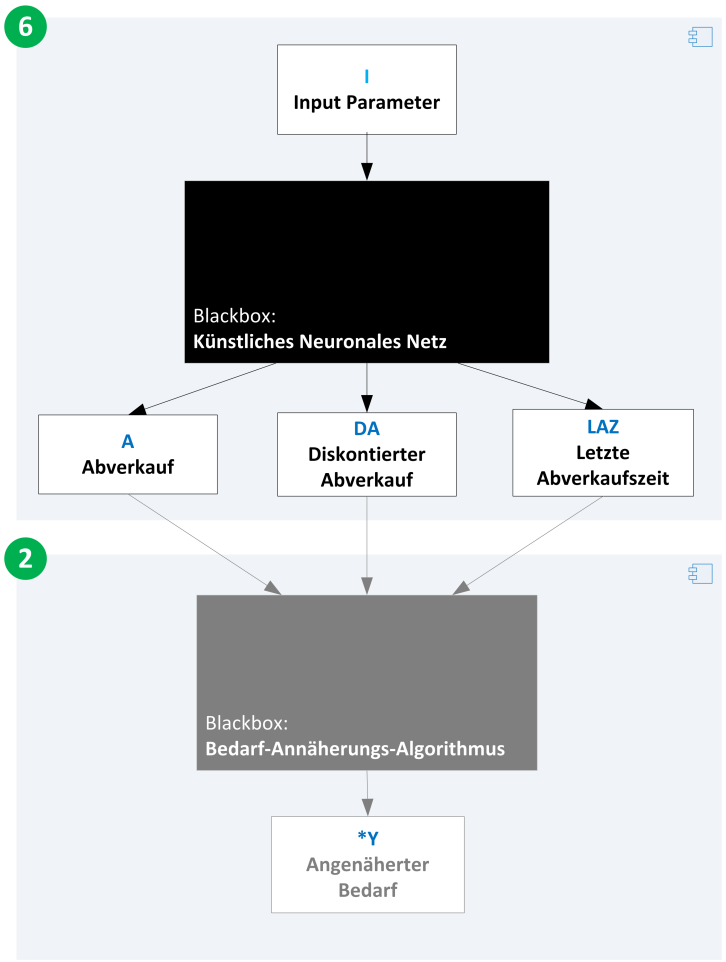


Abbildung 10.4.: Modifizierte und in dieser Arbeit umgesetzte fachliche Architektur

Durch die im Folgenden skizzierte und im Rahmen dieser Arbeit umgesetzte Teillösung, bleibt aber sowohl die Zielsetzung als auch eine zukünftige Erweiterbarkeit erhalten. Dazu wurde der Ansatz, wie in Abbildung 10.4 (Seite 73) gezeigt, modifiziert. Ohne die Möglichkeit den historischen Bedarf *X* anzunähern, ist es nicht möglich, ein KNN zu trainieren, das den Bedarf prognostiziert, da kein Zielwert vorhanden ist, an dem sich der Lernalgorithmus orientieren kann.

Eine mögliche Alternative wäre nun, die Ermittlung einer Absatzprognose statt der angestrebten Bedarfsprognose. Der Nachteil dieser Einschränkung wäre jedoch, dass sie nicht erweiterbar wäre. Eine Absatzprognose wäre ausreichend, um eine grundsätzliche Tauglichkeit KNN bei der Ermittlung von Prognosen für Backwaren zu überprüfen, müsste jedoch durch eine andere Implementierung ersetzt werden, sobald die benötigten Komponenten aus dem bestehenden System zur Verfügung stehen.

Die Modifizierung sieht deshalb vor, die Werte vorherzusagen, die der Algorithmus zur Bedarfsannäherung zukünftig benötigt, um den Bedarf anzunähern (Komponente **2**). Damit bleibt die Möglichkeit erhalten - nach der Isolierung von *BA* - den Bedarf aus den prognostizierten Variablen *A*, *DA* und *LAZ* zu extrapolieren. Sobald dann auch die Bedarfsprognose *Y* von allen Einflüssen der Backmengenempfehlung isoliert wurde, können *Y* und $*Y$ auf Basis der hier prognostizierten Werte miteinander verglichen werden.

Ob der Lösungsansatz mit KNN für den Kunden also einen Vorteil gegenüber seiner bisherigen Implementierung bringen würde, muss in einer Folgearbeit oder firmenintern untersucht werden. Was diese Arbeit aber nach wie vor leisten kann, ist:

- Die hier angedeutete Refaktorierungsanalyse[3] für das bestehende System.
- Eine genaue Analyse der Problemstellung, im Hinblick auf ihre Modellierung mit KNN
- Eine Prototypische Implementierung der ermittelten Modellierung mit Erkenntnissen über:
  - die Plausibilität der erzielten Prognosen
  - die Handhabbarkeit der Technologie in Verbindung mit Java-Softwarelösungen.

[3] Firmenintern hat diese Arbeit eine umfangreichere Analyse geliefert, als hier aus lizenzrechtlichen Gründen dargestellt werden kann.

11. Konfiguration, Modellierung und Implementierung

Aufbauend auf der in Abbildung 10.4 (Seite 73) skizzierten fachlichen Architektur wird das vorliegende Kapitel den konkreten Entwurf KNN zur Bedarfsprognose des konkreten Artikels - Laugenbrötchen - vorstellen.

Der Fokus der praktischen Ausarbeitung liegt auf der Modellierung der Bedarfsprognose durch KNN, speziell auf den verschiedenen Modellierungen des Input-Vektors und deren Auswirkungen. Diese Thematik wird in Abschnitt 11.2 behandelt. Zuvor soll jedoch die konkrete Konfiguration des Tupels $A = [V, I, L]$ vorgestellt werden.

11.1. Konfiguration des KNN

Die im Folgenden vorgestellten Ausprägungen des Tupels $A = [V, I, L]$, setzen die in Abschnitt 7.1 (Seite 30) vermittelten Grundlagen voraus und beziehen sich auf die dort vorgestellten Parameter. Sollte die Ausprägung eines Parameters bereits durch die Wahl des Multilayer Perceptron (MLP), als grundlegende Netzarchitektur, festgelegt sein, wird der Parameter im Folgenden nicht mehr aufgeführt.

Bezug nehmend auf die in Abbildung 8.9 (Seite 59) vorgestellte iterative Optimierung und Reduzierung einer initialen Menge verschieden modellierter KNN, übernimmt die folgende Eingrenzung der Konfiguration das Optimieren und Reduzieren im Rahmen der Trainingsphase. Statt viele verschiedene KNN mit verschiedenen Konfigurationen zu trainieren und die mit den besten Trainingsresultaten zu wählen, wird die Auswahl aus der empfohlenen Konfiguration in der Literatur getroffen.

11.1.1. Konfiguration der Verarbeitungsschritte im Neuron V

Aktivierungsfunktion F_A :
Bei der zu ermittelnden Abbildungsfunktion zwischen den abhängigen und/oder unabhängigen Variablen und den Prognosegegenständen A, DA und LAZ handelt es sich mit Sicherheit um eine nicht-lineare Funktion. Somit kommt als Aktivierungsfunktion in den Hidden-Neuronen

eine der beiden Sigmoiden-Funktionen in Frage: die Logistische-Funktion oder die Tangens-Hyperbolicus-Funktion. Im Vergleich zur Logistischen-Funktion, ermöglicht die Tangens-Hyperbolicus-Funktion die Ausgabe negativer Werte, was wünschenswert ist, wenn der Prognosegegenstand negative Werte annehmen kann. In der vorliegenden Problemstellung sind allerdings keine negativen Werte sinnvoll, so dass die Wahl auf die **Logistische Funktion** gefallen ist.

11.1.2. Konfiguration der Informationsverarbeitung und Propagierung I

Netzwerktopologie T:
Die Netztopologie wird definiert über:

- *Anzahl Neuronen in der Input-Schicht*: Diese Anzahl wird, wie bereits gezeigt, über die Modellierung des Input-Vektors definiert. Abschnitt 11.2 wird konkrete Zahlen hinsichtlich der Wahl der verschiedenen Modellierungsformen liefern.
- *Anzahl der Hidden-Schichten*: Hier wird auf die Empfehlungen von (Masters, 1993), (Klimasauskas, 1996) und (Kaastra und Boyd, 1996) zurückgegriffen, so wenige Hidden-Schichten wie möglich einzusetzen. Testdurchläufe werden mit einer Hidden-Schichten durchgeführt.
- *Anzahl Neuronen pro Hidden-Schicht*: Bei der Anzahl der Hidden-Neuronen pro Schicht wird auf die Empfehlung und Argumentation von Masters (1993) zurückgegriffen. Diese besagt, dass man mit einer zu kleinen Anzahl an Hidden-Neuronen beginnen sollte, um sie dann, je nach Bedarf, sukzessive zu erhöhen. Da zum jetzigen Zeitpunkt nicht bekannt ist, wie wenige Hidden-Neuronen zu wenige sind, wird die Anzahl über die Geometrische-Pyramiden-Regel von Masters ermittelt, da diese im Vergleich zu den anderen Daumenregeln aus der Literatur, die geringste Anzahl errechnet. Und zwar mit $\sqrt[2]{n * m}$ bei n Input-Neuronen und m Output-Neuronen. Sollten die Notwendigkeit entstehen, die Anzahl der Hidden-Neuronen zu erhöhen, kann auf die Empfehlungen von Bailey und Thompson (1990) und Heaton (2011)/Moustafa (2011) zurückgegriffen werden:
  - Empfehlung von Bailey und Thompson (1990) : Mindestens 75% der Anzahl Input-Neuronen als Hidden-Neuronen einzusetzen -> $n * 0,75$.
  - Empfehlung von Heaton (2011)/Moustafa (2011): Doppelt so viele Hidden-Neuronen wie Input-Neuronen pro Hidden-Schicht zu verwenden -> $2 * n$.
- *Anzahl Neuronen in der Output-Schicht*: Wie die Definition der Teillösung in Abschnitt 10.3.4 bestimmt, ist das Ziel der hier implementierten Prognosen, die Werte vorherzusagen, die später dazu benutzt werden können den vermeintlichen Bedarf für $t + 1$ anzunähern. Der Output-Vektor ist also für alle Modellierungsformen:
  $\overrightarrow{o} = [A_{t+1}, DA_{t+1}, LAZ_{t+1}]$, mit A = Abverkauf, DA = Diskontierter Abverkauf, LAZ = Letzte Abverkaufzeit, jeweils zum Prognosezeitpunkt $t+1$. Alle im Rahmen dieser Arbeit eingesetzte KNN haben somit 3 Output-Neuronen.

11.1.3. Konfiguration der Parametrisierung des Lernprozesses L

Lernalgorithmus G:
Der bekannteste, am besten erforschte und mit MLP und überwachtem Lernen fest in Verbindung gebrachte Lernalgorithmus ist der Backpropagation Algorithmus (Riedmiller und Braun, 1993), (Moustafa, 2011). Da es sich dabei um ein Gradientenabstiegsverfahren handelt, entsteht eine Vielzahl typischer Probleme. *"Die Konvergenz des Algorithmus in lokale Minima, die langsame Konvergenz des Trainings auf flachen Plateaus, die Oszillation in steilen Schluchten und das Verlassen vorzuziehender lokaler zugunsten schlechterer Minima"* (Crone, 2010, S. 200). Diese Problematiken können ausführlich in (Zell, 1997, S. 112ff) nachgelesen werden.

Die Auswirkungen dieser Probleme lassen sich durch die individuelle Anpassung verschiedener Lernparameter abschwächen. Dazu gehört die Lernrate μ und das Momentum ϵ. Die Lernrate bestimmt, wie schnell der Algorithmus konvergiert: eine zu kleine Lernrate führt dazu, dass der Algorithmus nur sehr langsam konvergiert und im schlimmsten Fall zu viele Schritte braucht um zu einer akzeptablen Lösung zu kommen. Eine zu hohe Lernrate kann hingegen zu Oszillationen führen und verhindern, dass der Trainingsfehler unter einen akzeptablen Wert fällt . Die Einführung eines Momentums soll diese Instabilität wiederum abschwächen und den Abstieg in globale Minima beschleunigen (Riedmiller und Braun, 1993). Das Problem beider Parameter ist jedoch *"[...] that the optimal value of the momentum parameter ϵ is equally problem dependent as the learning rate μ, and that no general improvement can be accomplished"* (Riedmiller und Braun, 1993, S. 586). Nur durch umfangreiches und zeitaufwendiges Testen verschiedener Kombinationen aus μ und ϵ, für jeden Anwendungsfall individuell, kann ansatzweise sicher gestellt werden, dass der minimal mögliche Fehler während des Trainings eines KNN ermittelt wird.

Riedmiller und Braun (1993), Igel und Hüsken (2003) und Heaton (2011) plädieren auf Grund dieser Problematiken dafür, statt des Backpropagation Algorithmus, den Resilient Propagation Algorithmus (RPROP) zu verwenden - eine Erweiterung und Verbesserung des Backpropagation Algorithmus. Heaton, der Chefentwickler des verwendeten KNN-Frameworks *encog*, beschreibt die Unterscheidung zwischen Backpropagation und Resilientpropagation wie folgt: *"The resilient propagation training (RPROP) algorithm is often the most efficient training algorithm for supervised feedforward neural networks. One particular advantage to the RPROP algorithm is that it requires no parameter setting before using it. There are no learning rates, momentum values or update constants that need to be determined. This is good because it can be difficult to determine the exact optimal learning rate. Resilient propagation will typically outperform backpropagation by a considerable factor"* (Heaton, 2011, S. 74). Die Implementierung von RPROP in *encog* erfolgte ursprünglich wie von (Riedmiller und Braun, 1993) erstmals vorgestellt und wurde dann konform der Verbesserungen in (Igel und Hüsken, 2003) erweitert.

Auf Grund der besseren Performance[1] und der einfacheren Handhabung wird in dieser Arbeit der RPROP als Lernalgorithmus eingesetzt.

Stopkriterium S:
Als Stopkriterium wird stets eine maximale Anzahl an Trainingsiterationen definiert. Diese maximale Anzahl wird im Rahmen vergleichender Testläufe variiert, um einerseits zu ermitteln, welche Modellierungen die niedrigsten Abbildungsfehler erreichen und andererseits Over- und Underfitting-Analysen durchzuführen.

11.1.4. Zusammenfassung der Konfiguration

Die folgende Tabelle 11.1 gibt einen Überblick über die gewählten Konfigurationsparameter und dient als Referenz für den weiteren Verlauf der Arbeit:

F_A Aktivierungsfunktion	Logistische Funktion	
T Netztopologie	n = # Input-Neuronen	gemäß Input-Vektor
	m = # Output-Neuronen	3
	# Hidden-Schichten	1
	# Hidden-Neuronen	$\sqrt[2]{n * m}$ $(2 * n, n * 0,75)$
G Lernalgorithmus	Resilient Propagation (RPROP)	
S Stopkriterium	max. # Trainingsiterationen	

Tabelle 11.1.: Übersicht über die, für dieser Arbeit, gewählten Konfigurationsparameter

11.2. Modellierung

Die Entscheidungsfindung, ob die Bedarfsermittlung für Backwaren des Folgetages am genausten durch eine Zeitreihen- eine kausale oder eine kombinierte Prognose ermittelt werden kann, ist ein zentraler Bestandteil dieser Arbeit. Die beschriebenen Vorteile eines datenbasierten Prognoseverfahrens, unter Einsatz KNN, sind in Fällen wie diesen sehr vorteilhaft. Den unterschiedlichen Modellierungsformen muss nicht durch die Verwendung verschiedener Prognoseverfahren Rechnung getragen werden, wie es bei den modellbasierten Verfahren der Fall wäre, sondern lediglich durch die unterschiedliche Modellierung des Input-Vektors.

[1] In einigen der ersten Testversuche, im Rahmen dieser Arbeit, wurde die Performance von Backpropagation und RPROP verglichen. Der Vergleich der Trainingszeit zweier KNN bei einer einfachen Zeitreihenprognose mit n=7 und einem Stopkriterium von 5000 Trainingsiterationen zeigte, dass der RPROP-Algorithmus knapp doppelt so schnell war wie der Backpropagation Algorithmus und einen um 37% niedrigeren Trainingsfehler erzielte.

Für das vorliegende Kapitel gelten nach wie vor die in Abschnitt 5.2.5 (Seite 18) getroffenen Annahmen und vereinbarte Notation, sowie die Grundlagen der Modellierung von Prognosen mit KNN der Abschnitte 8.1 und 8.2.

Der Modellierung des Input-Vektors kommt bei der Entwicklung KNN mindestens eine so große Bedeutung zu, wie der Konfiguration der Architektur A. Egal wie viel Zeit und Ressourcen man in die Optimierung der Konfiguration investiert, wenn es keine Abbildung zwischen dem gelieferten Input und dem gewünschten Output gibt, kann kein noch so gut konfiguriertes KNN diese finden. *"Success in designing a neural network depends on a clear understanding of the problem. Knowing which input variables are important in the market being forcasted is critical"* (Kaastra und Boyd, 1996, S. 219).

Einer Handelsprognose mit KNN muss also stets eine Domänenanalyse vorausgehen, im Rahmen derer ermittelt wird, welche möglichen Einflussvariablen auf den Prognosegegenstand existieren. Die ermittelten Variablen werden dann, abhängig von der gewählten Modellierungsform, in Input-Vektoren modelliert.

11.2.1. Bestimmung prognoserelevanter Variablen

Es gilt zu ermitteln, von welchen Variablen funktional auf die Prognosegegenstände A, DA und LAZ geschlossen werden kann. Die Ermittlung von Einflussfaktoren ist im Normalfall ein sehr zeitaufwendiger Prozess, der einerseits eine große Erfahrung mit Prognosen im Allgemeinen und andererseits ein umfangreiches Domänenwissen voraussetzt.

In diesem Zusammenhang werden die Vorzüge einer Zeitreihenprognose deutlich. Sollte es sich bei der vorliegenden um eine Problemstellung handeln, bei der die zu prognostizierenden Werte durch ihre Historie hinreichend bestimmt werden können, dann ist die Wahl der prognoserelevanten Variablen sehr einfach. Die Input-Variablen wären lediglich die historischen Realisierungen des Prognosegegenstandes:

A:	Abverkaufszahl
DA:	Diskontierte Abverkaufszahl
LAZ:	Letzte Verkaufszeit

Tabelle 11.2.: Abhängige Variablen für eine Modellierung einer Zeitreihenprognose

Im Falle einer kausalen oder kombinierten Prognose, hängt die Genauigkeit der Prognose von A, DA und LAZ davon ab, ob dem KNN Einflussvariablen geliefert werden, die ausreichend Informationen über die Prognosegegenstände beinhalten. Hier eine Liste der Werte, die im Rahmen dieser Arbeit als Einflussvariablen ermittelt wurden:

T:	durchschnittliche Tagestemperatur
N:	durchschnittliche Niederschlagsmenge
TW:	Tag der Woche
TM:	Tag des Monats
M:	Monat
zuF:	Abstand zum nächsten Feiertag
vonF:	Abstand vom letzten Feiertag
PV:	Preisveränderung ja/nein

Tabelle 11.3.: Unabhängige Einflussvariablen für eine Modellierung einer kausalen oder kombinierten Prognose

Es wird also vermutet, dass der Bedarf an Brötchen, für den Folgetag, von vier Einflusskategorien abhängt:

1. **W** : Wettereinflüsse, repräsentiert durch **T** und **N**
2. **K** : Kalendarische Einflüsse, repräsentiert durch **TW**, **TM** und **M**
3. **F** : Abstand zu Feiertagen, repräsentiert durch **zuF** und **vonF**
4. **P** : Preisveränderung durch eine Rabatt- oder Werbeaktion, repräsentiert durch den boolschen Wert **PV**

Viele weitere Einflussvariablen sind denkbar, wurden in dieser Arbeit aber nicht berücksichtigt, um einerseits die Komplexität der Modellierung in einem handhabbaren Rahmen zu halten und andererseits, weil der Zugriff auf diese Variablen nicht möglich oder zu aufwendig gewesen wäre. Zu nennen wären in diesem Zusammenhang sicherlich:

- Ferien: Befindet sich der Prognosezeitpunkt innerhalb von Schulferien oder (bei Filialen in touristischen Gegenden) in einer Haupt- oder Nebensaison?
- Einschätzungen von Filialmitarbeitern: Es gibt zu viele mögliche Einflussfaktoren, die sich nicht alle im Vorwege modellieren lassen, die aber durch eine persönliche Einschätzung von Filialmitarbeitern in die Prognose einfließen könnten. Dazu gehören beispielsweise Faktoren wie:
  - Baustelle vor Supermarkt
  - Rockfestival in Filialnähe
  - Umbauten in Filiale

11.2.2. Modellierungsformen

Ziel dieser Arbeit ist es die bestmögliche Modellierungsform für die Prognose von Backwaren im Einzelhandel zu finden. Dazu werden im Folgenden die Modellierungen vorgestellt, die, basierend auf der Theorie in Kapitel 5, als Kandidaten in Frage kommen. Es wird wieder auf die in Kapitel 5 eingeführte Darstellungsform zurückgegriffen, da diese kompakter ist, als die zuletzt in den Abbildungen 8.1, 8.2 und 8.3 verwendete. Es wird aber nach wie vor impliziert, dass jede Ausprägung einer Variable durch ein Input-Neuron modelliert wird. Um die Modelle einigermaßen übersichtlich zu halten, werden in den folgenden Diagrammen lediglich die Kategorien von Einflussvariablen **W**, **K**, **F** und **P** statt der einzelnen Variablen abgebildet.

- **Fall A - Zeitreihenprognose** (Abbildung 11.1, Seite 82): Diese Modellierung wird gute Prognoseergebnisse produzieren, wenn die zukünftige Entwicklung von A, DA und LAZ von ihren Werten in der Vergangenheit abhängt. Eine Konfigurationsmöglichkeit ist hier die Größe des Zeitfensters in die Vergangenheit n - als sinnvolle Größen wurden hier 7, 14 und 28 Tage angenommen.
- **Fall B - Kausale Prognose** (Abbildung 11.2, Seite 82): Sollten hingegen die unabhängigen Variablen den größten Einfluss auf A, DA und LAZ haben, dann wäre die kausale Modellierungsform am erfolgversprechendsten. Hier wird davon ausgegangen, dass die Wettereinflüsse, die kalendarischen Einflüsse, der Abstand zu Feiertagen und die Information über die Preisveränderung am Prognosetag ausschlaggebend für eine gute Prognose sind.
- **Fall C - Kombinierte Prognose C** (Abbildung 11.3, Seite 82): Für den Fall, dass weder durch die Zeitreihen- noch die kausale Modellierung gute Vorhersagen ermitteln werden können, kann dem KNN die Vereinigungsmenge beider Input-Vektoren präsentiert werden. In der Annahme, dass die zukünftigen Realisierungen von A, DA und LAZ weder durch ihre Historie alleine, noch durch die Einflussfaktoren am Prognosetag alleine hinreichend bestimmt werden.
- **Fall D - Kombinierte Prognose D** (Abbildung 11.4, Seite 83): Die kombinierte Prognose D geht noch einen Schritt weiter und stellt dem KNN nicht nur die historischen Realisationen von A, DA und LAZ zur Verfügung, sondern auch der Einflussvariablen der Kategorien W, K, F und P. Dies ermöglicht dem Netz, auch die historischen Entwicklungen der Einflussvariablen in die Prognoserechnung einzubeziehen.

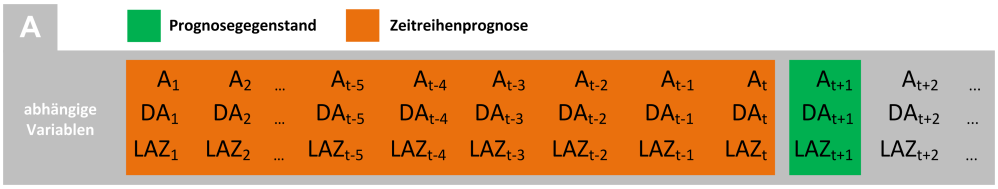


Abbildung 11.1.: Zeitreihenprognose mit n > 1 und m = 0 und allen abhängigen Variablen D, DA und LAZ

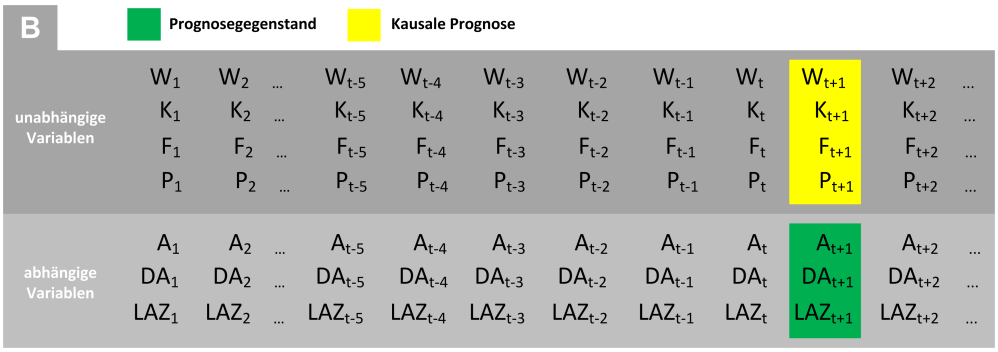


Abbildung 11.2.: Kausale Prognose mit n = 0 und m = 1 und allen unabhängigen Variablen der Kategorien W, K, F und P

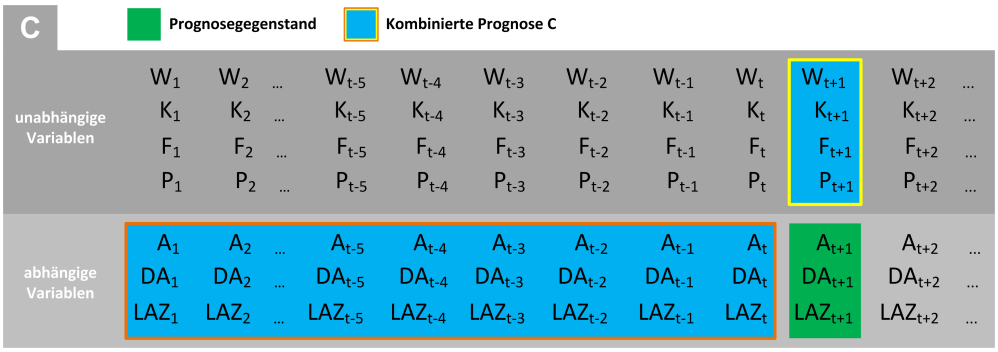


Abbildung 11.3.: Kombinierte Prognose mit n > 1 und m = 1 und allen abhängigen Variablen D, DA und LAZ und allen unabhängigen Variablen der Kategorien W, K, F und P

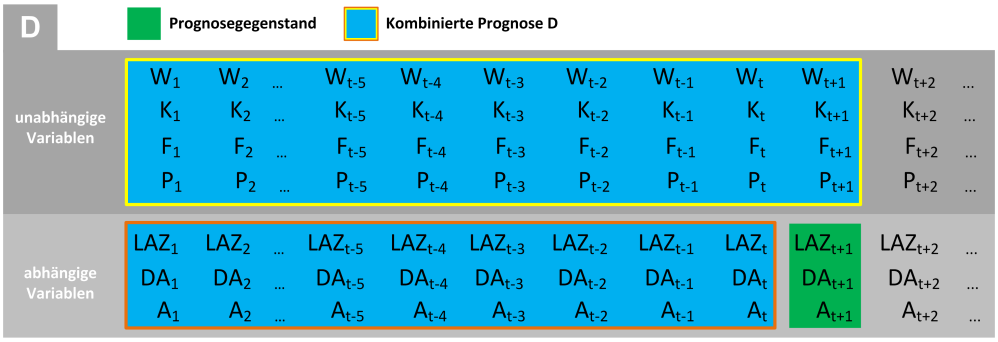


Abbildung 11.4.: Kombinierte Prognose mit n>1 und m=n+1 und allen abhängigen Variablen D, DA und LAZ und allen unabhängigen Variablen der Kategorien W, K, F und P

11.3. Implementierung

11.3.1. Normalisierung

Die gewählten abhängigen und unabhängigen Variablen bestehen aus semantisch und numerisch sehr unterschiedlichen Werten. Die Abverkaufszahlen, in der zur Verfügung stehenden Datenbasis, sind Ganzzahlen in einem Intervall von 1 bis 162, die Niederschlagsmenge wird in mm Regen pro Quadratmeter angegeben und bewegt sich zwischen 0.00 und 2.00, die Preisveränderung ist ein boolscher Wert und die letzte Abverkaufszeit wird in Minuten gerechnet, so dass eine typische LAZ zwischen 1200 (20:00h) und 600 (10:00h) Minuten liegt. Würde man all diese Werte nun in ihrer Rohform in das KNN einspeisen, dann würden die numerisch hohen Werte, wie beispielsweise die LAZ, eine viel höhere Gewichtung bekommen, als die niedrigen Werte, wie beispielsweise die Niederschlagsmenge. Ein Neuron würde durch eine Netzeingabe von [A=32, N=1.9, PV=0, LAZ=1145] und der Logistischen Aktivierungsfunktion (Intervall von 0..1) auf jeden Fall aktiviert werden und die Reduzierung des Niederschlags auf 0.0, was semantisch ein enormer Unterschied wäre, würde sich auf den Zustand des Neurons kaum auswirken. In der Literatur wird dieses Phänomen meist als *Saturation* bezeichnet - ein Neuron wird durch sehr hohe Input-Werte saturiert (Sarle, 2002).

Damit ein Neuron (der Hidden-Schicht) die verschiedenen Eingabewerte trotzdem ähnlich gewichten kann, müsste das KNN zunächst lernen, die unterschiedlichen Wertebereiche durch entsprechende Verbindungsgewichte zu kompensieren (vergleiche Abschnitt 7.1.1 zum Einfluss der Verbindungsgewichte auf die Aktivierung eines Neurons). Diese Art der Kompensation kann jedoch beliebig viel Zeit in Anspruch nehmen und ist nur schwer steuerbar.

Eine gängige Lösung für dieses Problem ist eine Vorverarbeitung aller Werte des Input-Vektors - die Normalisierung. Ziel dabei ist es alle Input-Werte um den Nullpunkt der Aktivierungsfunktion herum zu skalieren (Heaton, 2011). Der höchste Wert, den eine bestimmte Variable annehmen kann, soll dem höchsten Aktivierungswert der Aktivierungsfunktion entsprechen und der niedrigste Wert der Variable dem niedrigsten Aktivierungswert. So ist garantiert, dass

jeder Variable die gleiche Gewichtung zukommt[2].

Es existieren verschieden Formeln zur Normalisierung. In dieser Arbeit wurde die von Heaton (Heaton, 2011) und Kaastra und Boyd (Kaastra und Boyd, 1996) empfohlene Berechnung verwendet:

$$f(x) = \frac{(x - d_L)(a_H - a_L)}{(d_H - d_L)} + a_L \tag{11.1}$$

Hierbei definiert x den zu normalisierenden Wert, d den höchsten (H) und niedrigsten (L) Wert in den Daten, den eine Variable vom Typ x annehmen kann und a steht für den höchsten und niedrigsten Wert der Aktivierungsfunktion. Wenn wie hier die Logistische Aktivierungsfunktion gewählt wurde, kann natürlich auf die Addition von a_L am Ende verzichtet werden.

Daraus ergibt sich, dass der höchste Wert in den Daten immer 1 ist, der niedrigste 0 und der mittlere 0.5. Der oben genannte Beispielvektor [A=32, N=1.9, PV=0, LAZ=1145] würde normalisiert also folgendermaßen aussehen: [0.1925, 0.9500, 0.000, 0.9083].

Eine Besonderheit stellt die Filialnummer dar. Wie in Abschnitt 10.1 (Seite 66) gezeigt, bestehen die verwendeten Datensets aus Datenpaaren verschiedener Filialen. Jedes Datenpaar enthält also die Information zu welcher Filiale die Daten gehören. In den zugrundeliegenden Stammdaten werden alle Filialen durch eine 6 stellige Filial-ID identifiziert. Würde man diese nun einfach mit der oben genannten Formel normalisieren und in den Input-Vektor integrieren, würde man dem KNN suggerieren, dass eine Filiale mit der ID 900000 eine größere Relevanz hat, als eine Filiale mit der ID 100000. Die Filial-IDs dienen aber ausschließlich dem Zweck, die jeweiligen Datenpaare einer Filiale zuordnen zu können und nicht, um die Daten der Filialen unterschiedlich gewichten zu können. Eine Lösung für diesen Fall ist die *Binärverteilung*[3] der Filial-IDs (Sarle, 2002) wie im Folgenden veranschaulicht:

Angenommen es gibt 5 Filialen mit den IDs 100000, 200000, 300000, 400000, 500000 oder 1, 2, 3, 4, 5 oder 'schöne Filiale', 'große Filiale', 'Stadt-Filiale', 'Land-Filiale', 'Flaggschiff', ganz egal. Dann wird die Filiale in jedem Datenpaar, statt mit einem Feld, mit 5 Feldern repräsentiert, wovon jeweils eins den Wert 1 hat und alle anderen den Wert 0. Die 5 genannten Filialen würden also folgendermaßen repräsentiert werden:

[2] Sollte im Vorwege bekannt sein, dass manche Variablen einen größeren Einfluss auf den Prognosegegenstand haben, so kann die Normalisierung dahingehend beeinflusst werden, dass die normalisierten Werte dieser Variablen mit einem entsprechenden Faktor multipliziert werden (Sarle, 2002).

[3] In (Sarle, 2002) wird der Begriff der 'Binärverteilung' nicht verwendet, sondern lediglich das Konzept dahinter erklärt. In Ermangelung anderer zuverlässiger Quellen und einer festgelegten Terminologie habe ich für den Rahmen dieser Arbeit den Begriff der Binärverteilung gewählt.

	Binärverteilte Filial-ID				
Filial-ID	**Feld 1**	**Feld 2**	**Feld 3**	**Feld 4**	**Feld 5**
100000	1	0	0	0	0
200000	0	1	0	0	0
300000	0	0	1	0	0
400000	0	0	0	1	0
500000	0	0	0	0	1

Die vorgestellte Binärverteilung wird auch für die Variablen Tag der Woche (TW), Tag des Monats (TM) und Monat (M) durchgeführt, da auch in diesen Fällen keine unterschiedliche Wertigkeit impliziert werden soll, sondern lediglich eine Kategorisierung. TW besteht somit aus 7 Feldern, TM aus 31 und M aus 12.

11.3.2. Maximale Input-Matrix

Die vorangegangenen Abschnitte haben nun alle Informationen und Werte geliefert, die benötigt werden, um alle im Rahmen dieser Arbeit geplanten Versuche durchführen zu können. Dieser Abschnitt wird abschließend auf die Implementierung und Zusammensetzung einer Datenstruktur eingehen, die für die durchzuführenden Versuche alle notwendigen Daten liefert - eine maximale Input-Matrix.

Als maximale Input-Matrix bezeichne ich eine Datenstruktur, die die maximal benötigten Werte für alle Modellierungsformen enthält. Selbst die komplexeste Modellierung soll in *einer Zeile* dieser Matrix alle benötigten Werte für ein Datenpaar vorfinden. Es werden also alle verfügbaren Daten in einer maximalen Datenstruktur zusammengefasst und je nach Versuchsaufbau und Modellierungsform werden Spalten und Zeilen aus dieser 'herausgeschnitten'. Alle Versuchsaufbauten und Modellierungsformen bedienen sich somit der gleichen maximalen Datenstruktur und können die gleiche Implementierung zur Bildung von Datenpaaren nutzen. Der einzige Unterschied besteht in der Wahl der benötigten Spalten.

Von den vorgestellten Modellierungsformen ist die kombinierte Prognose D, aus Abbildung 11.4 (Seite 83), die komplexeste, diejenige mit dem größten Input-Vektor. Damit gewährleistet ist, dass bei einem historischen Zeitfenster n, mit der maximalen Größe 28, aus einer Zeile alle nötigen Werte für ein Datenpaar extrahiert werden können, ist *eine Zeile* der maximalen Input-Matrix wie in Abbildung 11.5 (Seite 87) aufgebaut.

Die Anzahl der Zeilen ergibt sich aus der Anzahl vorhandener historischer Tage und der Anzahl der Filialen, für die diese historischen Zahlen existieren. Wie in Tabelle 10.1 (Seite 67) nachzulesen ist, existiert für diese Arbeit eine Datenbasis von 114 verkaufsoffenen Tagen in 24 Filialen. Die maximale Input-Matrix hat somit 2736 Zeilen[4] und 1706 Spalten[5].

[4] $114 * 24 = 2736$

[5] $24 + 58 + (28 * 58) = 1706$ (siehe 11.5 (Seite 87))

Abbildung 11.6 (Seite 88) verdeutlicht, wie sich diese Zeilen zu einer Matrix zusammensetzen. Die weißen Felder deuten an, dass diese Felder nicht mit Werten gefüllt werden können. Die Matrix hat stets die Form:

$(a * b) \times (c * d)$
mit a = Anzahl Filialen
b = Anzahl maximal verfügbarer historischer Tage p/Filiale
c = Prognosehorizont, also hier immer 1
d = Größe des maximal verfügbaren historischen Zeitfensters

Im vereinfachten Beispiel ist a = 2 (statt 24), b = 28 (statt 114), c = 1 und d = 14 (statt 28). Die Variablen c und d stehen stellvertretend für die Anzahl von 58er Blocks aus Abbildung 11.5 (Seite 87). Für die ersten $d - 1$ Datensätze jeder Filiale existieren nicht alle historischen Tage, um alle Felder füllen zu können. Beim Datensatz b jeder Filiale, also dem jeweils letzten verfügbaren Tag, existiert kein Zielwert $t + 1$. Im Fall einer Zeitreihen- oder kombinierten Prognose können somit lediglich $a * ((b - (h - 1)) - c)$ Datenpaare geformt werden, mit h = Größe des jeweils genutzten historischen Zeitfensters (7, 14 oder 28). Im Fall einer kausalen Prognose können stets $a * (b - c)$ Datenpaare geformt werden. Folgende Tabelle gibt einen Überblick über die mögliche Anzahl von Datenpaaren je nach Modellierungsform. In der mittleren Spalte ist die Anzahl Datenpaare aufgeführt, die im vereinfachten Beispiel 11.6 (Seite 88) je nach Modellierungsform geformt werden können:

Modellierungsform	# Datenpaare (Beispiel)	# Datenpaare (tatsächliche Prognose)
kausal	54	2712
Zeitreihe + kombiniert:		
$h = 7$	42	2568
$h = 14$	28	2400
$h = 28$	-	2064

Der Vorteil dieser maximalen Input-Matrix ist, dass sie nur einmal erstellt werden muss und ab dann alle möglichen Modellierungen, Test- und Versuchsfälle die benötigten Daten aus dieser Datenquelle beziehen können. Die Logik, welche Daten aus welcher Datenquelle in welcher Form zusammengesetzt werden, muss also nur an einer Stelle implementiert werden und ist gut gekapselt. Die Auswahl, welche Spalten für welchen Testfall benötigt werden, kann dann mit einfachsten Mitteln, über ein indizierte Liste, gelöst werden.

Abbildung 11.5.: Darstellung einer Zeile der maximalen Input-Matrix

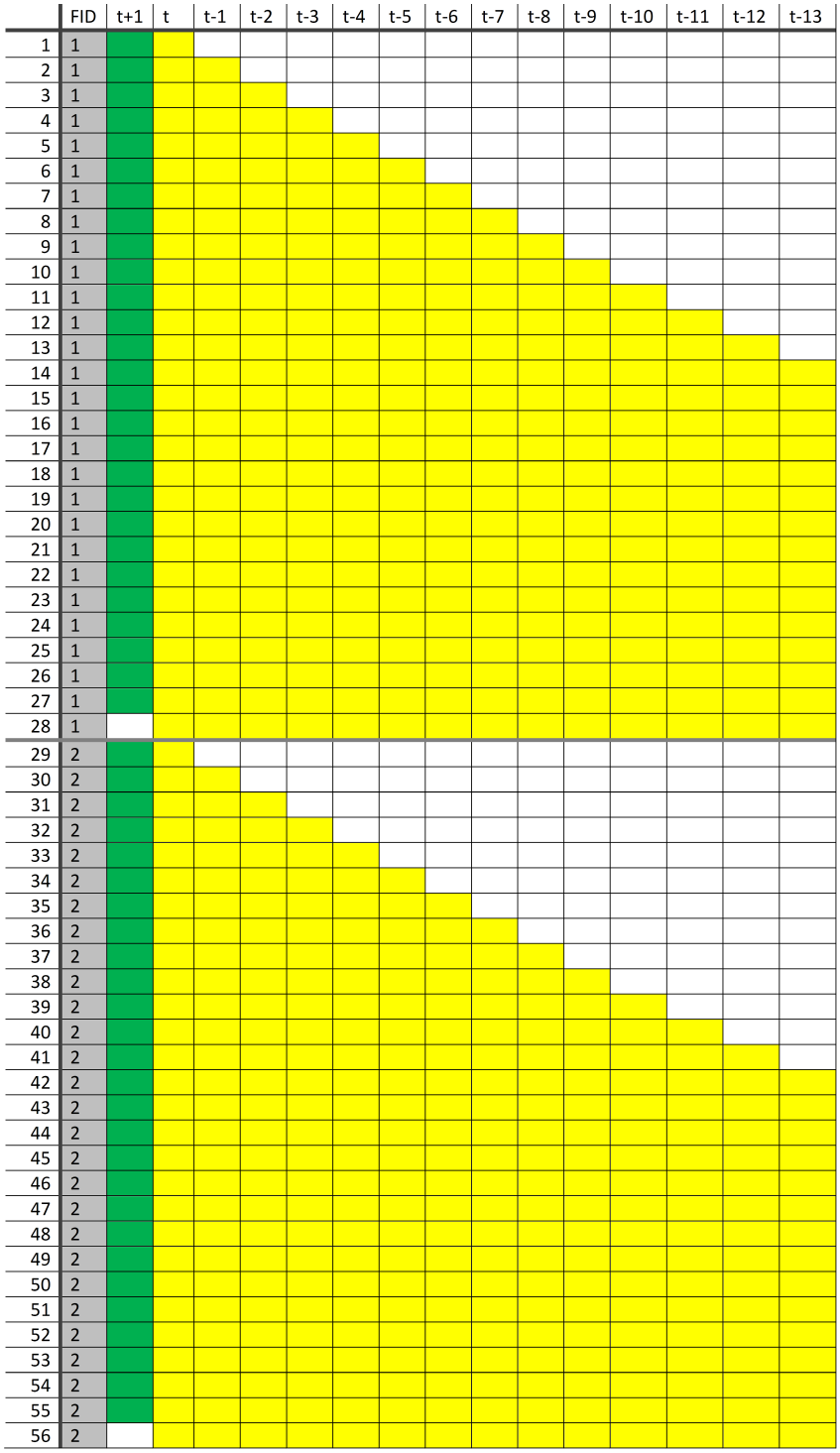


Abbildung 11.6.: Schematische Darstellung der gefüllten und nicht gefüllten Felder der maximalen Input-Matrix, mit n = 14 und 2 Filialen mit jeweils 28 Tagen

11.4. Bewertung der Handhabbarkeit der Implementierung KNN

Ein Sekundärziel dieser Arbeit ist die Bewertung der Handhabbarkeit einer neuronalen Lösung, zur Bedarfsprognose von Backwaren, im Rahmen einer Java-Softwareentwicklung. Dazu gehört einerseits eine Bewertung des eingesetzten Frameworks und andererseits eine Einschätzung der Komplexität der Implementierung und der Handhabbarkeit im täglichen Betrieb einer produktiven Softwarelösung.

11.4.1. Bewertung des KNN-Frameworks Encog

Wie bereits in Kapitel 9 beschrieben, fiel die Vorabbewertung des eingesetzten Frameworks Encog positiv aus. Drei Kritikpunkte möchte ich jedoch aufführen, die sich im Laufe der Nutzung herauskristallisiert haben und dazu führten, dass die Implementierungszeit deutlich länger ausgefallen ist, als sie nach allgemeinem Ermessen hätte ausfallen müssen:

1. Encog befindet sich gerade im Versionswechsel, von Version 3.0 auf 3.2, der erhebliche Änderungen mit sich bringt. Diese Änderungen sind zwar durchweg positiv, doch leider ist die Dokumentation fast ausschließlich noch auf dem Stand 3.0. Das führte dazu, dass viele wichtige Nutzungsinformationen nur über das Durchforsten der Implementierung Encogs herausgefunden werden konnten. Von meinem zwischenzeitlichen Plan, auf Version 3.0 zurück zu gehen, um mit der Dokumentation konform zu bleiben, wurde mir von Jeff Heaton, dem Chefentwickler Encogs, abgeraten. Version 3.2 behebt einige Performanceeinschränkungen, die bei den zum Teil sehr großen Input-Vektoren, meiner Anwendung, zu nicht unerheblichen Performanceeinbußen geführt hätten. Die Diskrepanz zwischen Implementierung und der veralteten Dokumentation, hat somit zu einigen unnötigen Hürden und Verzögerungen geführt.
2. Die Parametrisierung vieler Primärklassen ist derart gekapselt, dass nicht ersichtlich ist, welche fundamentalen Entwurfsentscheidungen mit der Nutzung dieser Klassen implizit getroffen werden und auch die jeweilige Klassen oder Schnittstellendokumentation geht nicht ausreichend auf diese Entscheidungen ein. Der Hintergedanke bei dieser Kapselung ist sicherlich, den Einstieg zu erleichtern und den Anwender nicht sofort mit zu vielen Implementierungsdetails zu konfrontieren, die bei der Verwendung von KNN sehr vielfältig sind. Doch das ist gleichzeitig der Kritikpunkt, denn diese vielfältigen Details entscheiden bei KNN oft über Erfolg oder Misserfolg und sollten bewusst eingesetzt werden und nicht mühsam aus dem Code herausgesucht werden müssen.
3. Encog bietet viele sinnvolle Utilityklassen und Methoden zur Vorverarbeitung der Daten und zur Zusammenstellung von Trainings-, Validierungs- und Testdaten. Leider sind diese zum Großteil auf ganz spezifische (und sicherlich gängige) Anwendungsfälle ausgerichtet und nicht generisch genug, um Abweichungen von diesen Standardfällen zu tolerieren. Dies führte dazu, dass ich wichtige Teile der Anwendung nachträglich umstellen und

selbst implementieren musste, da die versprochene Funktionalität der Utilityklassen und Methoden nicht auf den vorliegenden Anwendungsfall zutraf.

Im Großen und Ganzen handelt es sich bei diesen Kritikpunkten um Problematiken die beim Einsatz fremder APIs häufig auftreten und speziell im Open-Source Bereich mit lückenhafter Dokumentation einhergehen. Die Bewertung des Frameworks fällt somit gemischt aus: die angebotene Funktionalität ist sehr gut und umfangreich, die Umsetzung jedoch an vielen Stellen zu einseitig, so dass selbst minimale Abweichungen vom Standardfall oft nicht toleriert werden.

Als lohnende (nicht Java-) Alternativen könnten sich die KNN-Komponenten von matlab und R erweisen.

11.4.2. Bewertung der Komplexität und Handhabbarkeit der Implementierung

Die Komplexität und Handhabbarkeit der Implementierung kann natürlich nicht unabhängig vom eingesetzten Framework betrachtet und bewertet werden. Jedoch zeichnen sich sehr deutliche Arbeitsschritte ab, die unabhängig vom Framework durchgeführt werden müssen und eine Einschätzung der Handhabbarkeit, unabhängig vom Framework, zulassen. Zu den identifizierten Arbeitsschritten gehören:

Entwicklungsphase	**Produktivphase**
1. Abrufen der prognoserelevante Daten von zuverlässigen Datenquellen	
2. Modellierungsformspezifische Zusammenführung der Daten	
3. Normalisierung der Daten	
4E. Zusammenstellung von Trainings-, Validierungs- und Testsets	*4P.* Zusammenstellen des Input-Vektors
5E. Trainieren, Validieren, Testen	*5P.* Erstellung der Prognose
6E. Auswertung der Ergebnisse	*6P.* Periodisches Training mit neuesten Daten (*4E*, *5E* und *6E*)

Die Punkte *5E* und *5P* sind für die jeweilige Phase entscheidend, ihr Implementierungs- und Wartungsaufwand ist jedoch gering. Hier besteht lediglich die Anforderung an das Framework, die Konfiguration des KNN und die Parametrisierung des Lernprozesses zu gewährleisten und die Arbeit performant zu verrichten. Dafür gibt es für jedes Framework Implementierungsbeispiele, da es sich um die zentralen Aufgaben einer KNN-Implementierung handelt. Bei *5E* ist lediglich zu beachten, dass die Trainingszeit, je nach Größe des Trainingssets und der Komplexität des Modells, sehr lange dauern kann. Die längste in dieser Arbeit benötigte Trainingszeit betrug immerhin 4 Stunden und 15 Minuten und das, obwohl die Größe des Trainingssets eher als klein zu bezeichnen ist.

Deutlich schwieriger und aufwendiger sind die Punkte *1*, *2* und *3*. Für den Produktivbetrieb müssen diese Anforderungen zuverlässig und automatisiert erfolgen und erfordern Fallback-Strategien für den Fall, dass Datenquellen nicht zur Verfügung stehen oder fehlerhafte Daten liefern. Während der Entwicklungsphase ist es wichtig, dass die Punkte *2* und *3* nicht für jedes zu entwickelnde und zu testende Modell einzeln durchgeführt werden, da die Prozesse zur Zusammenführung und Normalisierung der Daten, schon bei einer vermeintlich kleinen Menge an Variablen, schnell unübersichtlich und fehleranfällig werden. Diese Prozesse stabil und performant zu implementieren, sollte für ein erfahrenes Entwicklerteam jedoch keine große Hürde darstellen, man sollte sich lediglich darüber im Klaren sein, dass die Vorverarbeitung der Daten mehr Zeit, Ressourcen und guter Ideen benötigen wird, als die Implementierung der KNN selbst.

Die Zusammenstellung von Trainings-, Validierungs- und Testsets in *4E* ist in erster Linie eine fachliche Anforderung. Es muss je nach Größe und Beschaffenheit der Datenbasis entschieden werden, welche Teile davon in welcher Reihenfolge in welches Set einfließen sollen (vgl. dazu Abschnitt 8.6). Der Implementierungsaufwand hängt dann von der in *2.* gewählten Datenstruktur ab. Die Zusammenstellung der Datensets, aus der hier gewählten maximalen Input-Matrix, ist sehr einfach, die performante Implementierung der Input-Matrix selbst jedoch nicht.

Anforderung *6E* zählt, mindestens in den Anfängen der Entwicklungsphase, zu den zeitlich aufwendigsten und inhaltlich anspruchsvollsten. Wie das folgende Kapitel 12 noch ausführlich zeigen wird, sind die Trainings-, Validierungs- und Testergebnisse der KNN ohne die nötige Erfahrung im Umgang mit KNN nicht leicht zu interpretieren. Die Auswertung der Ergebnisse erfordert Zeit und gute Visualisierungs- und Aufbereitungsstrategien der Daten, an denen nicht gespart werden sollte. Denn der Erfolg des produktiven KNN, hängt zu 100% von der korrekten Interpretation der Ergebnisse während der Entwicklungsphase ab.

Das periodische neu Trainieren produktiver KNN, in Punkt *6P*, wurde im Rahmen dieser Arbeit nicht konzipiert oder durchgeführt. Es ist allerdings zu den großen Stärken der Prognose mit KNN zu zählten und soll hier als wichtige Anforderung zumindest aufgeführt werden. Mit dem Hinweis, dass der Aufwand dieses Teils der Implementierung sicherlich nicht unerheblich sein wird.

Zusammenfassend kann gesagt werden, dass die Hauptherausforderungen bei der Entwicklung einer Lösung zur Bedarfsprognose mit KNN folgende sind:

- Eine programmierhandwerkliche Herausforderung bei der Automatisierung von Beschaffung und Vorverarbeitung großer Datenmengen und die Bereitstellung dieser Daten in geeigneter/performanter Form.
- Die Ermittlung eines geeigneten KNN-Frameworks und die Einarbeitung in dieses.
- Das Sammeln von Erfahrung mit KNN und der korrekten Interpretation der Ergebnisse.

12. Versuchsaufbau und Auswertung

Im diesem Kapitel werden die verschiedenen Versuchsaufbauten zusammengefasst, die Ergebnisse präsentiert und analysiert.

Von den in Abschnitt 11.2.2 (Seite 81) vorgestellten Modellierungsformen werden die folgenden Versuchsreihen mit diesen Modellen durchgeführt:

- Fall A - **Zeitreihenprognose (ZrP)** (Abbildung 11.1, Seite 82)
- Fall B - **Kausale Prognose (KaP)** (Abbildung 11.2, Seite 82)
- Fall D - **Kombinierte Prognose (KoP)** (Abbildung 11.4, Seite 83)

Die Fälle A und D werden jeweils mit einem historischen Zeitfenster n von 7, 14 und 28 Tagen ausgeführt. Insgesamt stehen für die Versuche also 7 Modelle zur Verfügung: ZrP(7), Zrp(14), Zrp(28), KaP, KoP(7), KoP(14) und Kop(28).

Im ersten Schritt werden Anhaltspunkte gesucht, um die initiale Modellmenge einzugrenzen. Die verbleibenden Modelle werden dann im zweiten Schritt gemäß der ermittelten Lösungsvorschläge optimiert. Am Ende der Versuchsreihen bleibt das Modell übrig, das die Problemstellung unter den gegebenen Voraussetzungen am besten löst.

Die Versuchsaufbauten orientieren sich an den in Abschnitt 8.4.1 (Seite 54) vorgestellten Over- und Underfittinganalysen von Andrew Ng. Sie werden hier aber nicht nur zur Ermittlung von Over- und Underfitting Tendenzen eingesetzt, sondern liefern gleichzeitig Indikatoren für die Prognosequalität der jeweiligen Modelle

12.1. Eingrenzung der Modellmenge

Ziel der ersten Versuchsreihen ist die Eingrenzung der Modellmenge. Welche der 7 Modelle neigen zu einem Over- oder Underfitting der vorhandenen Datenbasis? Welche historische Zeitfenstergröße n liefert die besten Ergebnisse für ZrP und KoP? Welche Modelle haben das größte Optimierungspotenzial? Welche Modelle produzieren den niedrigsten Validierungsfehler?

Bezug nehmend auf den in Abbildung 8.9 (Seite 59) dargestellten iterativen Optimierungszyklus, repräsentiert dieser Abschnitt das Optimieren und Reduzieren im Rahmen der Validierungsphase.

Versuchsaufbau

Für jedes der 7 Modelle werden die maximal vorhandenen Trainingspaare aus dem jeweiligen Trainingsset in 10 verschieden große Teilmengen unterteilt - in 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 Prozent (%) der gesamten Datenmenge. Mit jeder dieser Teilmengen wird dann ein KNN trainiert und der Trainingsfehler (TE) berechnet. Das Stopkriterium S ist eine maximal vorgegebene Anzahl an Trainingsiterationen in 4 verschiedenen Abstufungen: 100, 500, 2500 und 10000 Iterationen. Alle trainierten KNN werden abschließend gegen das selbe Validierungsset validiert und der Validierungsfehler (VE) berechnet. Sowohl für den TE als auch für den VE werden die Output-Werte der jeweiligen KNN in ihrer normalisierten Form in die auf Seite 41 gelieferte Formel 7.4 zur Berechnung des MSE eingesetzt. Dabei ist der dargestellte TE und VE der Mittelwert aus den errechneten Mean Square Error (MSE) der drei Output-Werte Abverkauf (A_{nor}), Diskontierter-Abverkauf (DA_{nor}) und letzte Abverkaufszeit (LAZ_{nor})[1]. Der TE und VE eines KNN i berechnet sich also aus: $(MSE(A_i) + MSE(DA_i) + MSE(LAZ_i))/3$. Wie gut die drei Prognosegegenstände im Einzelnen prognostiziert werden, wird im Laufe der Ergebnisanalyse noch vorgestellt, zur Eingrenzung der Modellmenge ist der Mittelwert vorerst ein ausreichender Indikator.

Es ergibt sich also für jedes der 7 Modelle die folgende Ergebnistabelle:

Trainings	Trainingsfehler (TE)				Validierungsfehler (VE)				Zeilenmittel	
Daten	100	500	2500	10000	100	500	2500	10000	avg TE	avg VE
10%										
20%										
30%										
40%										
50%										
60%										
70%										
80%										
90%										
100%										

Tabelle 12.1.: Schematische Ergebnistabelle für Testdurchläufe zum Vergleich von verschiedenen Trainingssetgrößen und Anzahl von Trainingsiterationen

Für jedes der 7 Modelle wurden also 4 KNN, mit unterschiedlichen Stopkriterien, konfiguriert

[1]Wobei nor anzeigen soll, dass der MSE-Berechnung die normalisierten Werte zugrunde liegen und zum jetzigen Zeitpunkt noch nicht die realen de-normalisierten. Die De-Normalisierung der Daten/Ergebnisse wird erst zur Auswertung der Ergebnisse vorgenommen.

und mit 5 verschiedenen zufälligen Startgewichten[2] mit jeweils 10 verschiedenen Mengen an Trainingspaaren trainiert und gegen das selbe Validierungsset validiert. Insgesamt wurden also 28 verschiedene KNN jeweils 5*10 mal trainiert und validiert, was zu 1400 Versuchen in dieser Testreihe führte. Von den 5 Durchläufen mit verschiedenen Startgewichten wurde jeweils der kleinste erzielte Validierungsfehler gespeichert, so dass insgesamt 280 Testdurchläufe, in der oben abgebildeten tabellarischen Form, protokolliert wurden und als Grundlage für die folgenden Analysen dienen.

Aus Platzgründen werden die vollständigen Tabellen nicht in dieser Arbeit abgedruckt. Die folgenden Abschnitte präsentieren und diskutieren jedoch alle relevanten Ergebnisse und stellen die Auswertungen in zusammenfassenden Diagrammen zur Verfügung. Aus Gründen der Vollständigkeit enthält die CD, die dieser Arbeit beliegt, jedoch alle vollständigen Ergebnistabellen[3].

12.1.1. Analyse 1

In dieser Analyse wird der Trainings- und Validierungsfehler in Abhängigkeit zur Größe des Trainingssets geplottet. Aus der oben abgebildeten Tabelle 12.1 (Seite 93) wird das Zeilenmittel der Trainingsfehler gegen das Zeilenmittel der Validierungsfehler geplottet. Das Zeilenmittel wurde hier gewählt, um einerseits keine zu frühe Eingrenzung auf die Anzahl zu laufender Trainingsiterationen zu treffen und andererseits die Anzahl zu plottender Diagramme in einem überschaubaren Rahmen zu halten. Statt in 15 Diagrammen, konnten die Ergebnisse dieser Analyse in 3 Liniendiagrammen in Abbildung 12.1 (Seite 96) zusammengefasst werden, wobei jedes Diagramm die Ergebnisse einer Modellierungform enthält.

Ziel dieser Analyse ist zu ermitteln, welche Modelle zu einem Over- oder Underfitten der Datenbasis tendieren und welche Modelle unter den gegebenen Voraussetzungen die besten Er-

[2] Das Training eines KNN mit 5 verschiedenen zufälligen Startgewichten wäre in der Praxis lt. der bereits in Abschnit 8.3 erwähnten Empfehlung von Sarle (2002) zu wenig, um ein KNN produktiv setzen zu können. Dazu wird empfohlen, den Trainings-/Validierungszyklus 10 - 1000 mit verschiedenen Startgewichten zu durchlaufen. In diesem Abschnitt der Arbeit soll aber lediglich die allgemeine Tauglichkeit der gewählten Modelle begutachtet werden, so dass das beste Ergebnis einer Auswahl von 5 zufällig gewählten Startgewichten ausreichend ist. Der Grund für diese Einschränkung ist, dass der zeitliche Aufwand dieses Vorgehens beträchtlich ist und sich zum jetzigen Zeitpunkt der KNN-Entwicklung nicht lohnt. Das einmalige Durchlaufen von 10 verschiedenen Trainingssets bei einem KoP-Modell mit einem historischen Zeitfenster von 28 Tagen und 10000 Trainingsiterationen, ohne Validierung und Persistierung der Ergebnisse, dauert bereits 20 Stunden und 31 Minuten.

[3] Die folgenden Excel-Tabellen enthalten alle relevanten Ergebnisse aller durchgeführten Versuche:

- Analyse-1-KaP-Modelle.xlsx
- Analyse-1-KoP-Modelle.xlsx
- Analyse-1-ZrP-Modelle.xlsx
- Analyse-2-Optimierung-KaP-Modelle.xlsx
- Analyse-2-Optimierung-KoP-Modelle.xlsx
- Analyse-2-Optimierung-ZrP-Modelle.xlsx

gebnisse liefern.

Auswertung Analyse 1: Wie zu erwarten, steigt der Trainingsfehler mit zunehmender Anzahl an Trainingspaaren, wohingegen der Validierungsfehler sinkt, da die Generalisierungsfähigkeit der KNN zunimmt, je mehr Kombinationen von Input-Werten durch die Trainingsdaten abgedeckt werden. Auffallend ist jedoch, dass die Kurven unterschiedlich schnell konvergieren und dies auf unterschiedlichem Fehlerniveau.

Zur genaueren Betrachtung wurden die Kurven der Zeitreihen- und kombinierten Modellierung entzerrt, indem für jedes Zeitfenster n ein eigenes Diagramm geplottet wurde. Abbildung 12.2 (Seite 97) fasst diese zusammen. Zusätzlich liefert die folgende Tabelle in den Spalten *avg TE* und *avg VE* den jeweils letzten Wert jeder Kurve. Die Spalte *niedrigster VE - Wert* zeigt hingegen den niedrigsten gemessenen Validierungsfehler in allen Testläufen des jeweiligen Modells[4]. Die Spalte *niedrigster VE - Iteration* gibt an mit welchem Stopkriterium (100, 500, 2500 oder 10000 Trainingsiterationen) das jeweilige KNN diesen niedrigsten VE erreicht hat und die Spalte *niedrigster VE - Trainingsdaten* den dabei benutzten Prozentsatz des Trainingssets.

Modell	avg TE	avg VE	niedrigster VE		
			Wert	Iteration	Trainingsdaten
ZrP(7)	0,003910	0,004999	0,004709	100	90%
ZrP(14)	0,003519	0,004894	0,004655	100	100%
ZrP(28)	0,002656	0,005089	0,004705	500	100%
KaP	0,002535	0,006305	0,004702	100	100%
KoP(7)	0,000101	0,001098	0,000248	2500	100%
KoP(14)	0,000226	0,002264	0,000916	500	90%
KoP(28)	0,000334	0,005792	0,003529	100	100%

Was auffällt ist, dass der Abstand zwischen dem TE und VE, der Zeitreihenmodelle und des kausalen Modells, um eine Zehnerpotenz geringer ist, als bei den Modellen der kombinierten Prognose. Hinzukommt, dass der niedrigste gemessene VE beim besten KoP-Modell um einen Faktor $\sim$20 geringer ist, als bei den ZrP-Modellen und dem KaP-Modell. Dies deutet darauf hin, dass sich die Lernfähigkeit der gewählten Zeitreihen und kausalen Modelle ihrem Bestfall bereits stark angenähert hat und im Folgenden untersucht werden muss, ob diese durch eine Veränderung der Modelle gesteigert werden kann. Die Modelle der kombinierten Prognose zeigen jedoch schon in der gewählten Form, dass weiteres Lernpotential in ihnen steckt.

Betrachtung der Zeitreihenmodelle:
Teil A in Abbildung 8.8 (Seite 57) zeigte bereits, dass die schnelle Annäherung von TE und VE auf einem hohen Fehlerniveau zur Folge hat, dass das Hinzunehmen weiterer Trainingsdaten keine wesentliche Verbesserung des VE bewirken kann, da das zugrundeliegende Modell keine genauere Abbildung zwischen Input- und Output-Werten ermöglicht.

[4] Bei den ZrP- und KoP-Modellen ist das der niedrigste TE und VE aus jeweils 120 Testläufen und bei KaP-Modellen aus 40.

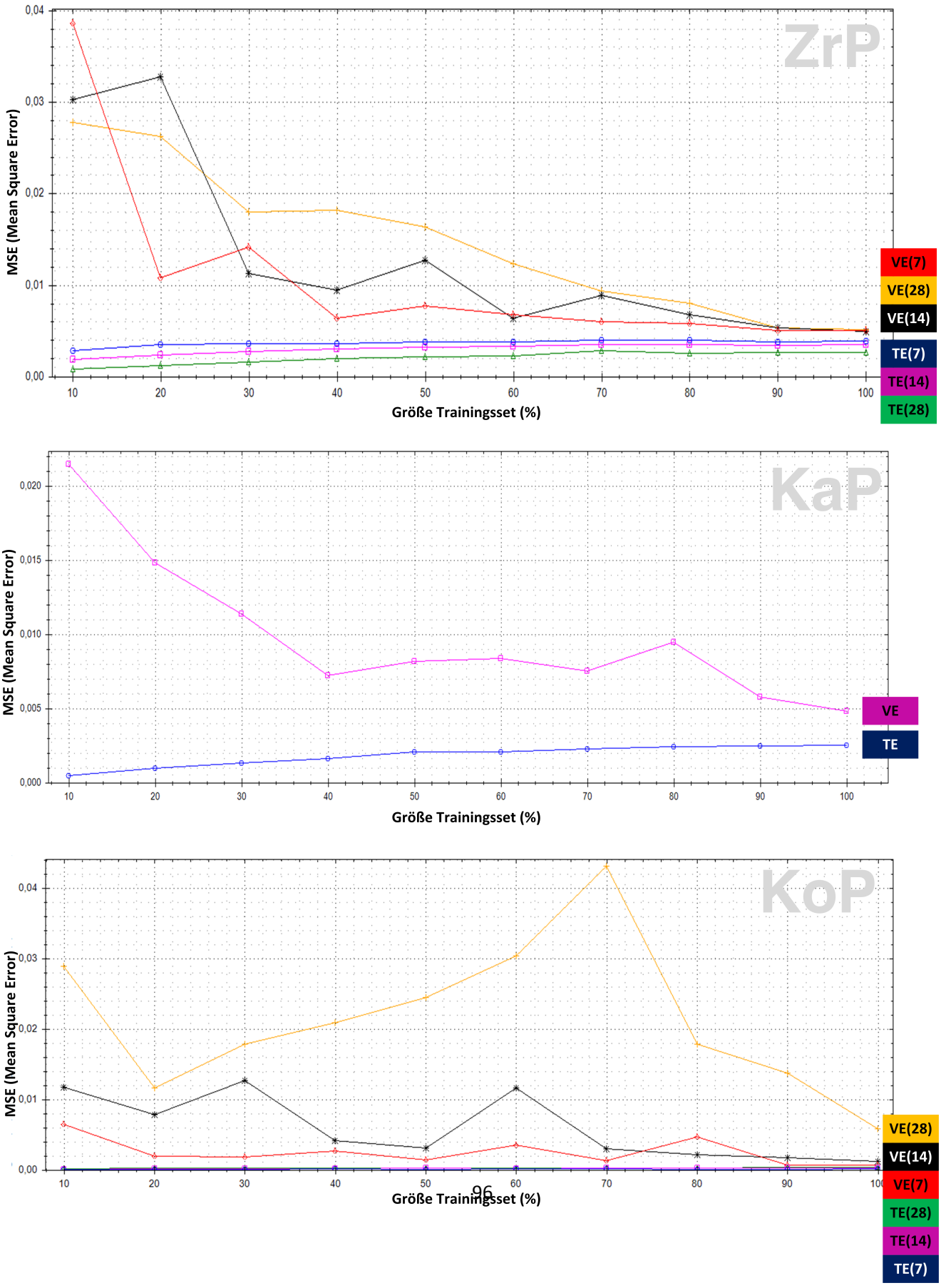


Abbildung 12.1.: Analyse 1: Trainings- und Validierungsfehler in Abhängigkeit zur Anzahl Trainingspaare

Abbildung 12.2.: Analyse 1: Zeitreihen- und Kombinierte Prognose, aufgesplittet

Es ist also anzunehmen, dass die gewählten Modelle der Zeitreihenprognose unter einem Underfitting-Problem leiden. Die Komplexität der Modelle ist somit entweder zu gering oder die Prognosegegenstände werden durch ihre historischen Ausprägungen alleine nicht ausreichend bestimmt. Die Annahme des Underfittings wird durch die Tatsache gestützt, dass kein einziger der 120 ZrP-Testversuche einen VE produzieren konnte, der sich von den Durchschnittswerten absetzen konnte.

Betrachtung des kausalen Modells:
Für das kausale Modell gelten die gleichen Überlegungen, die für die Zeitreihenmodelle soeben ausgeführt wurden. Der VE nähert sich dem TE auf einem hohen Fehlerniveau an. Der kleinste produzierte VE der 40 KaP-Testläufe ist lediglich 2,5 mal so groß wie der durchschnittliche TE und somit ist der beste VE um den Faktor $\sim$20 größer, als der beste VE der KoP-Testläufe.

Es ist also anzunehmen, dass auch das gewählten Modell der kausalen Prognose unter einem Underfitting-Problem leiden. Die Komplexität des Modells ist somit entweder zu gering oder die unabhängigen Variablen und die Prognosegegenstände sind nicht hinreichend korreliert und lassen somit keine zufriedenstellende Abbildung auf den Prognosegegenstand zu.

Betrachtung der kombinierten Modelle:
Teil B in Abbildung 8.8 (Seite 57) zeigte das Pendant zum Underfitting, das Overfitting. Die Fehlerkurven der KoP-Modelle ähneln diesem Fall: Der VE deutet eine Konvergenz mit dem TE erst bei einem vergleichsweise großen Trainingsset an, vorher ist der Abstand zwischen TE und VE groß bzw. beide Kurven fluktuieren sehr stark. Erst bei 80%iger Größe des Trainingssets deuten die Kurven an zu konvergieren, doch selbst bei Nutzung aller Trainingsdaten ist der Abstand zwischen beiden Kurven im Vergleich zum Underfitting-Fall, noch sehr groß. Im Fall der ZrP-Modelle war der VE im Schnitt 1,5 mal[5] so groß wie der des TE, bei dem KaP-Modell 2,5 mal und bei den KoP-Modellen 12,5 mal[6].

Dies deutet darauf hin, dass die gewählten Modelle der kombinierten Prognose dazu tendieren die vorhandene Datenbasis zu overfitten und somit ihr volles Potenzial nicht ausschöpfen können. Das Potenzial deutet sich aber bei dem niedrigsten ermittelten VE von 0,000248 durch das KoP(7)-Modell an.

Die in Abschnitt 7.1.2 erwähnte Daumenregel von (Klimasauskas, 1996) zur Bestimmung der Größe des Trainingssets in Abhängigkeit zur Anzahl Hidden-Neuronen, belegt die hier getroffene Annahme mit Zahlen. Klimasauskas empfiehlt, dass die Anzahl der Trainingspaare mindestens so groß sein sollte, wie die Anzahl der Verbindungsgewichte im Netzwerk. Tabelle 12.2 (Seite 99) zeigt, dass dies bei den KoP-Modellen bei weitem nicht der Fall ist.

Die hohe Diskrepanz zwischen der lt. Klimasauskas benötigten Anzahl an Trainingspaaren und der tatsächlich vorhandenen, schließt nicht aus, dass gute Prognosen durch dieses Modell erzielt werden können (wie die z.T. guten Ergebnisse des KoP(7)-Modells und mit Abstrichen des KoP(14)-Modells zeigen), aber steigert die Gefahr des Overfittings immens. Um die Robustheit

[5] ZrP(7) = 1,3; ZrP(14) = 1,4; ZrP(28) = 1,8
[6] KoP(7) = 11; KoP(14) = 10; KoP(28) = 17

Modell	# Output-Neuronen	# Input-Neuronen	# Hidden-Neuronen	# benötigter Trainingspaare	# vorhandener Trainingspaare
Kop(7)	3	488	38	18658	2277
KoP(14)	3	894	52	46454	2110
KoP(28)	3	1706	72	122262	1774
ZrP(7)	3	56	12	708	2277
ZrP(14)	3	84	15	1305	2110
ZrP(28)	3	140	20	2860	1774
KaP	3	79	15	1230	2445

Tabelle 12.2.: Gegenüberstellung benötigter und vorhandener Trainingspaare lt. Klimasauskas (1996)

der Prognosen zu steigern, sollte die Anzahl der Hidden-Neuronen bei den KNN der kombinierten Prognose reduziert werden, zumindest solange es nicht möglich ist, weitere Trainingspaare bereit zu stellen.

Abgeleitete Optimierungsmaßnahmen aus Analyse 1:

- ZrP + KaP: Steigerung der Komplexität der KNN
- KaP: Hinzunahme weiterer unabhängiger Variablen
- KoP: Vergrößerung der Datenbasis
- KoP: Verringerung der Komplexität der KNN

Abgeleitete Reduzierung der Modellmenge aus Analyse 1:
Betrachtet man erneut die Diagramme in Abbildung 12.1 (Seite 96) wird deutlich, dass ZrP(7) und KoP(28) die beschriebenen Under- bzw. Overfittingtendenzen am deutlichsten zeigen und aus der Modellmenge herausgenommen werden.

Der VE von ZrP(7) konvergiert sehr schnell gegen den angesprochen hohen TE und ist bei der maximalen Anzahl Trainingspaaren nur 1,4 so groß wie der entsprechende TE. Die Wahrscheinlichkeit, dass sich die Prognoseergebnisse dieses Modells durch eine Komplexitätssteigerung verbessert, wird als gering erachtet. Es ist davon auszugehen, dass die historischen Realisierungen der drei Prognosegegenstände, mit einem Zeitfenster von 7 Tagen in die Vergangenheit, als einziger Input in die Prognoserechnung nicht ausreichen.

Der VE von KoP(28) nähert sich von allen KoP-Modellen am spätesten an den TE an und der Abstand zwischen beiden ist mit $VE = TE * 17$ am größten. Was nicht überraschend ist, da es sich dabei um das mächtigste KNN der Modellmenge handelt. Somit ist auch die Diskrepanz zwischen der empfohlenen Anzahl an Trainingspaaren und der vorhandenen am größten. Das Potential dieses Modells ist weder ausgeschöpft noch ausreichend untersucht. Es ist aber eindeutig ersichtlich, dass das KoP(28)-Modell ohne die Hinzunahme weiterer Trainingsdaten nicht optimiert werden kann und scheidet somit aus der Modellmenge aus.

12.1.2. Analyse 2

In dieser Analyse wird der Trainings- und Validierungsfehler in Abhängigkeit zur Anzahl gelaufener Trainingsiterationen geplottet. Dazu wird jedes der 7 Modelle mit der jeweils maximalen Anzahl verfügbarer Trainingspaare über 100, 500, 2500 und 10000 Iterationen trainiert und der jeweilige Trainingsfehler (TE) berechnet. Anschließend werden alle Modelle gegen das gleiche Validierungsset validiert und der Validierungsfehler (VE) berechnet. Aus der oben abgebildeten Tabelle 12.1 (Seite 93) werden also die Ergebnisse der 100% Zeile (exklusive der avg-Werte) geplottet. Die drei Liniendiagramme in Abbildung 12.3 (Seite 101) fassen die Ergebnisse dieser Analyse zusammen, wobei jedes Diagramm wieder die Ergebnisse einer Modellierungform enthält.

Ziel dieser Analyse ist, zu ermitteln, welches Stopkriterium für die gewählten Modelle am sinnvollsten erscheint und welche Modelle zu schnell dazu tendieren, Muster in den Trainingsdaten zu fitten, die sich nicht auf die Validierungsdaten abbilden lassen.

Auswertung Analyse 2:
Die Modelle reagieren unterschiedlich auf die Erhöhung der Trainingsiterationen. Bei den ZrP- und KoP-Modellen sinken Trainings- und Validierungsfehler bis zu einer bestimmten Anzahl Iterationen gemeinsam, woraufhin der Validierungsfehler wieder ansteigt während der Trainingsfehler weiter sinkt. Dieses Verhalten stimmt mit den Erläuterungen zu Abbildung 8.6 (Seite 56) überein und beschreibt eine weitere Form des Overfittings. In diesem Fall ist die Ursache des Overfittens nicht eine zu hohe Netzkomplexität, gepaart mit zu wenigen Trainingsdaten, sondern die Tatsache, dass das KNN ab einer bestimmten Anzahl an Trainingsiterationen versucht unbeabsichtigte und unerwünschte Muster in den Daten - Rauschen - mit den Prognosegegenständen zu korrelieren. Das KNN beginnt also ab einem bestimmten Punkt, Muster zu fitten, die auf den Trainingsdaten den gewünschten Output zwar zufällig treffen, die Beziehung zwischen Input und gewünschtem Output aber nicht beschreiben können. Bei einem idealen Modell, sollten Trainings- und Validierungsfehler mit der Steigerung der Trainingsiterationen monoton fallen, bis sie gegen das globale Minimum konvergieren, egal wie viele Trainingsiterationen noch folgen.

Dies erklärt nun auch den bereits in Abschnitt 8.4.1 angesprochenen Unterschied zwischen dem Overtraining- und dem Overfitting-Ansatz genauer. Der Overfitting-Ansatz geht berechtigterweise davon aus, dass eine künstlich Beschränkung der Trainingsiterationen (im Englischen oft als *Early-Stopping* bezeichnet) nicht das grundlegende Problem der gewählten Modellierung behebt, sondern lediglich die Symptome mildert. Oft ist ein Early-Stopping jedoch nicht zu vermeiden, da es natürlich nicht immer möglich ist, ideale Modelle zu finden und diese mit idealen Bedingungen zu trainieren. Auch in dieser Arbeit wird der Overtraining-Ansatz Anwendung finden, da hier zum einen erst die Grundlagen der Modellierung erforscht werden und somit noch keine idealen Modelle vorliegen können und zum anderen nicht genug Trainingsdaten vorliegen, um frühen Formen des Overfittings zuverlässig zu entgehen.

Das KaP-Modell hingegen zeigt die gerade erläuterte Form des Overfittings nicht. Bis zu 2500

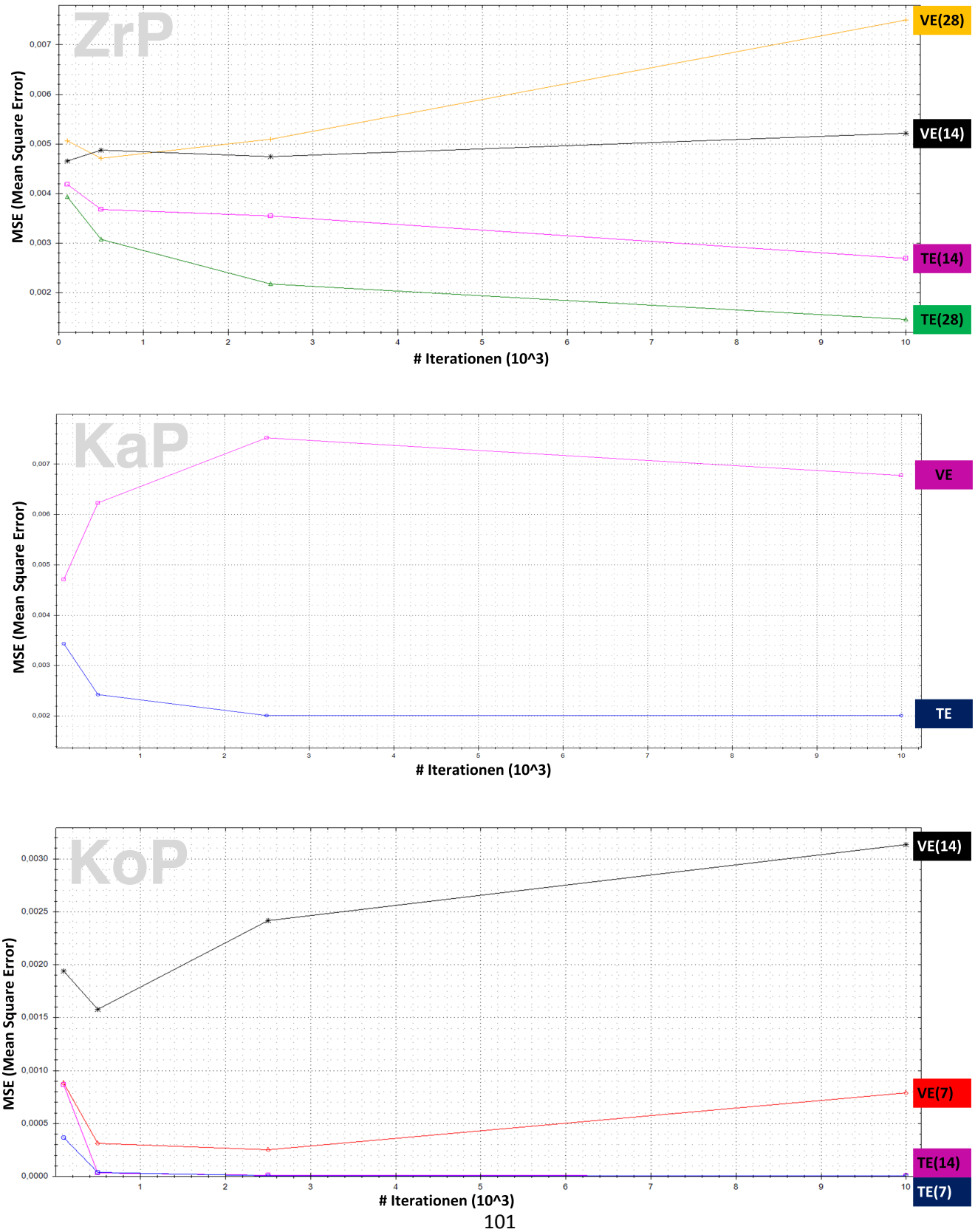


Abbildung 12.3.: Analyse 2: Trainings- und Validierungsfehler in Abhängigkeit zur Anzahl Trainingsiterationen

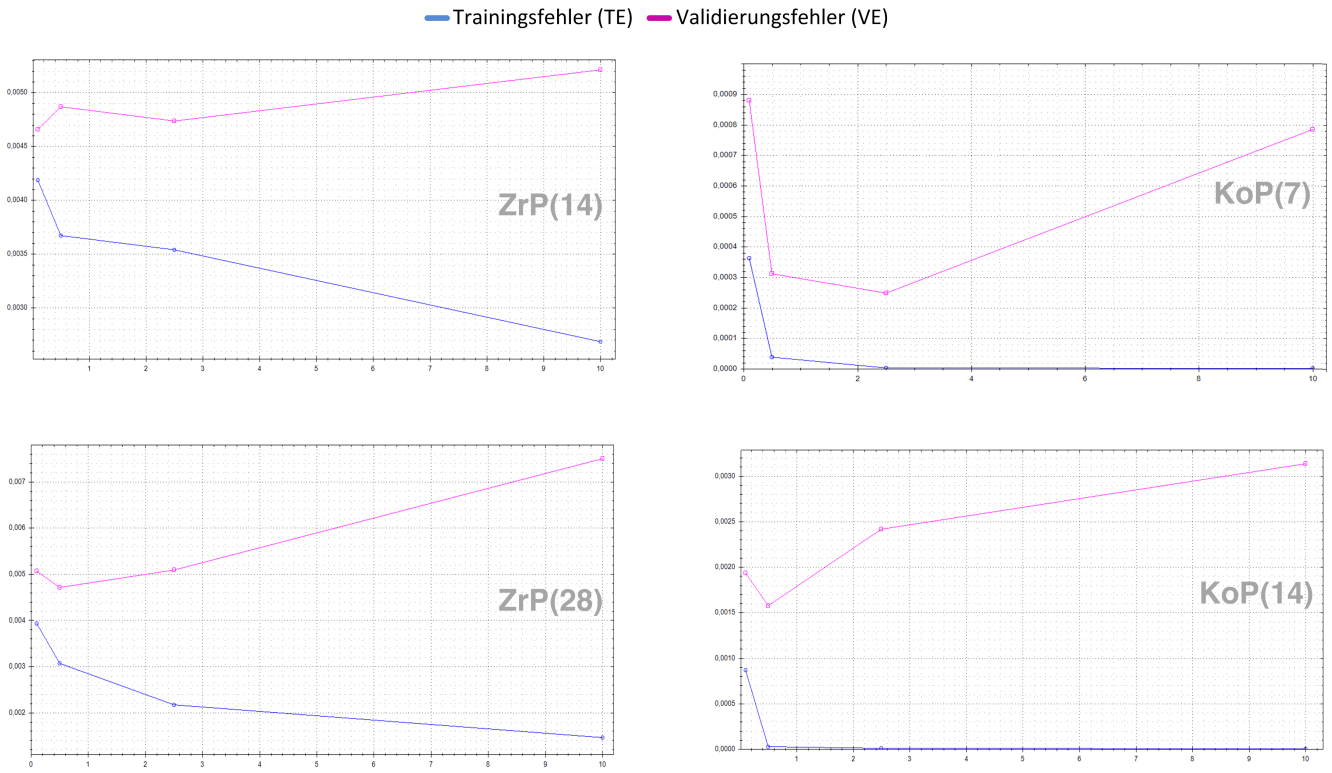


Abbildung 12.4.: Analyse 2: Zeitreihen- und Kombinierte Prognose, aufgesplittet

Trainingsiterationen zeigen TE und VE zwar das selbe Verhalten wie beispielsweise beim ZrP(14)-Modell, doch nach der Vervierfachung der Trainingsiterationen sinkt der VE, was bei den ZrP-KoP-Modellen nicht der Fall war. Was darauf hindeutet, dass das KaP-Modell robuster gegenüber dem Overfitting durch zu viele Trainingsiterationen ist, da der gewählte Input-Vektor weniger Rauschen zu enthalten scheint, als der der anderen Modelle.

Im Folgenden werden die drei Modellierungen genauer betrachtet, wozu die Diagramme der ZrP- und KoP-Modelle aufgesplittet wurden und einzeln in Abbildung 12.4 (Seite 102) dargestellt werden.

Betrachtung der Zeitreihenmodelle:
Das ZrP(14)-Modell erzielt seinen besten VE bei 100 Iterationen, wohingegen das ZrP(14) 500 Trainingsiterationen durchlaufen kann, bevor das KNN beginnt zu overfitten. Das ZrP(14) zeigt aber bei einem Stopkriterium von 2500 Iterationen, dass sich beide Fehler noch einmal annähern, bevor sie sich endgültig entfernen.

Wichtig ist, an dieser Stelle hervorzuheben, dass das in Analyse 1 festgestellte Underfitting der Zeitreihenmodelle in keinem Widerspruch zum hier gezeigten Overfitting steht. Analyse 1 zeigte, dass auf Grund einer zu geringen Netzkomplexität oder unzureichender Input-Variablen keine zufriedenstellende Abbildung zwischen Input und gewünschtem Output gefunden werden

konnte, das Netz also die Trainingsdaten zu ungenau trifft, underfittet, und somit die Validierungsdaten erst recht nicht gut treffen kann. Das hier in Analyse 2 gezeigte Overfitting bedeutet nun, dass die verwendeten KNN das Optimum, das sie unter den gegebenen Umständen erreichen können, bereits nach x Iterationen treffen und bei Trainingsiterationen > x beginnen die Trainingsdaten zu overfitten, indem sie den Trainingsfehler auf das Rauschen in den Daten optimieren.

Die Modelle der Zeitreihenprognose tendieren also zu einem Underfitting der Problemstellung, können aber durch zu viele Trainingsiterationen zu einem Overfitting der Trainingsdaten 'gezwungen' werden.

Betrachtung des kausalen Modells:
Anders verhält sich das kausale Modell. Auch diesem wurde durch Analyse 1 eine Tendenz zum Underfitting der Problemstellung nachgewiesen, doch der Input-Vektor des KaP-Modells scheint kein Overfitting der Trainingsdaten zu ermöglichen. Die Daten, die dem Netz zur Prognose präsentiert werden, enthalten also wenig bis kein Rauschen und/oder das Verhältnis zwischen Netzkomplexität und Größe der Datenbasis ist optimal. Salopp formuliert könnte man sagen: Das KNN versucht, mit den Daten die es bekommt, die bestmögliche Prognose zu ermitteln und, egal ob diese gut oder schlecht ist, sie wird durch eine Weiterführung des Trainings nicht verschlechtert. In dem vorliegenden Fall gilt sogar zu untersuchen, ob sie sich durch eine weitere Steigerung der Trainingsiterationen nicht gar verbessern lässt.

Das KaP-Modell hat den kleinsten VE nach 100 Trainingsiterationen erreicht.

Betrachtung der kombinierten Modelle:
Die KoP-Modelle ähneln den ZrP-Modellen, da sie auch zu einem Overfitting von verrauschten Trainingsdaten tendieren. Sie unterscheiden sich aber einerseits darin, dass der Trainingsfehler schneller konvergiert und dass andererseits zumindest der VE des KoP(7) länger als bei allen anderen Modellen mit dem TE fällt. Das Overfitten des KoP(7) beginnt also erst ab 2500 Iterationen, bis dahin optimiert das Modell den TE anhand von Mustern in den Trainingsdaten, die in den Validierungsdaten ebenfalls existieren und erreicht dadurch einen VE, der um eine Zehnerpotenz geringer ist als bei allen anderen Modellen.

Abgeleitete Optimierungsmaßnahmen aus Analyse 2:
In dieser Analyse wurden lediglich die Ergebnisse betrachtet, die die jeweiligen Modelle auf den maximalen Trainingsdaten erzielt haben. Wenn man sich jedoch den Verlauf der Ergebnisse von 10% der Trainingsdaten bis zu den gezeigten 100% betrachtet, zeigen diese, dass das Overfitting der ZrP- und KoP-Modelle bei 10% am deutlichsten war und mit jeder Vergrößerung des Trainingssets abgenommen hat. Dies entspricht der Erwartung, dass es mit zunehmender Größe des Trainingssets schwieriger wird, den Trainingsfehler auf zufälliges Rauschen in den Daten zu optimieren. Somit wäre eine Vergrößerung der Datenbasis auch nach Analyse 2 eine sinnvolle Optimierung.

Da diese Optimierung nicht möglich ist, wird die erwähnte Strategie des Early-Stoppings bei den ZrP- und KoP-Modellen Anwendung finden:

- ZrP + KoP: Reduzierung der maximalen Anzahl an Trainingsiterationen, die die verbleibenden Modelle laufen sollten. Nach den bisher gelieferten Ergebnissen sollten sie folgende Anzahl nicht überschreiten:
  - ZrP(14): 100 Iterationen
  - ZrP(28) + KoP(14): 500 Iterationen
  - KoP(7): 2500 Iterationen

Beim KaP-Modell hingegen wird überprüft werden, ob sich der VE dieses Modells durch eine Erhöhung der maximalen Anzahl an Trainingsiterationen weiter senken lässt.

Abgeleitete Reduzierung der Modellmenge aus Analyse 2:
Von den in der Modellmenge verbliebenen Modellen werden diejenigen aussortiert, die die größte Tendenz zum Overfitten der Datenbasis zeigen: das ZrP(28)-Modell und das KoP(14)-Modell.

Diese Entscheidung wird durch die Tatsache gestützt, dass in allen 140 Testläufen der ZrP-Modelle das in der Modellmenge verbleibende ZrP(14) den geringsten VE ermittelt hat und KoP(7) den geringsten VE aller 140 Testläufen der KoP-Modelle geliefert hat.

12.2. Optimierung der eingegrenzten Modellmenge

Die in der Modellmenge verbliebenen Modelle sind: **ZrP(14)**, **KaP** und **KoP(7)**. Diese wurden gemäß der Optimierungsmaßnahmen aus den vorangegangenen Analysen verändert und die daraus resultierenden Ergebnisse werden in diesem Abschnitt präsentiert und diskutiert.

Alle folgenden Versuche werden mit 100% der jeweils verfügbaren Trainingsdaten durchgeführt.

12.2.1. Optimierung des Zeitreihenmodells ZrP(14)

- Steigerung der Netzkomplexität durch Hinzunahme weiterer Hidden-Neuronen.
- Ermittlung der idealen Anzahl zu laufender Trainingsiterationen durch Early-Stopping.

Versuchsaufbau:
Die Anzahl der Hidden-Neuronen (HN) war bisher 16 und wurde durch die Empfehlung von Masters (1993) bestimmt. Dabei handelt es sich um die geringste, in der Literatur ermittelte, Empfehlung. Die Erhöhung der Netzkomplexität wird somit erreicht, indem die in Abschnitt 7.1.2 vorgestellten Empfehlungen von Bailey und Thompson (1990) und Heaton (2011)/Moustafa (2011) auf das ZrP(14) angewandt werden. Aus der Empfehlung von Bailey und Thompson (1990) ergeben sich 63 HN und aus der von Heaton (2011)/Moustafa (2011) 168.

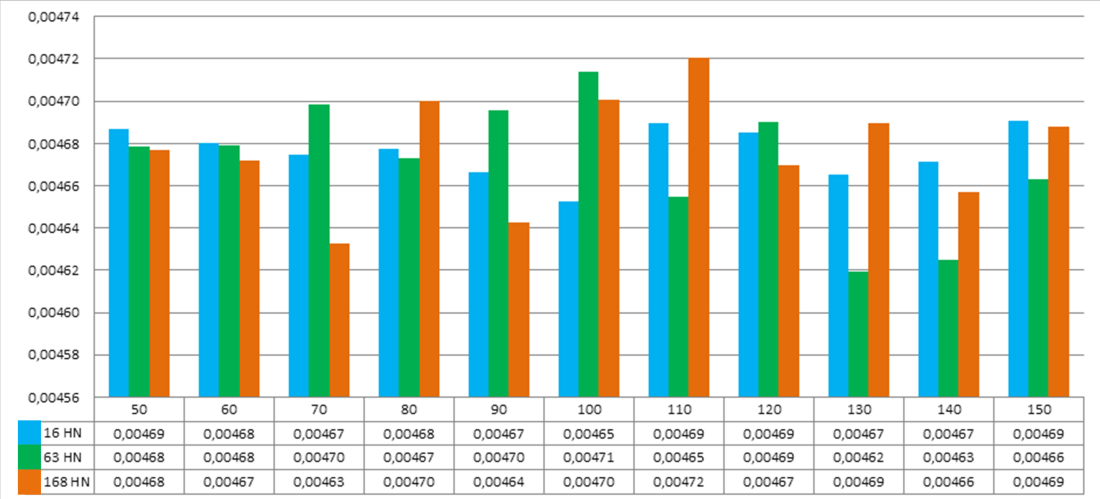
Die bisherigen Ergebnisse zeigten, dass der niedrigste Validierungsfehler des ZrP(14)-Modells nach 100 Trainingsiterationen erzielt wurde. Dieses Ergebnis soll nun überprüft und ein ideales Stopkriterium ermittelt werden.

Der daraus resultierende Versuchsaufbau lässt sich folgender Tabelle entnehmen, wobei die farbig markierten Felder die Kombination mit dem jeweils niedrigsten Validierungsfehler angeben:

Anzahl Iterationen	16 HN (bisher)	63 HN (Optimierung 1)	168 HN (Optimierung 2)
50			
60			
70			
80			
90			
100			
110			
120			
130			
140			
150			

Es ergeben sich also 3 Versuchsreihen, in denen jeweils ein ZrP(14)-Modell mit einer der dargestellten Anzahl Hidden-Neuronen (16, 63 oder 168) mit 11 verschiedenen Stopkriterien (50 - 150 Trainingsiterationen) trainiert und validiert wird. Das Validierungsset ist das gleiche wie bei den vorangegangenen Analysen, so dass die Ergebnisse vergleichbar bleiben. Auch bei diesem Versuchsaufbau wurde jede Versuchsreihe wieder mit 5 verschiedenen zufälligen Startgewichten durchgeführt und das Ergebnis mit dem niedrigsten Validierungsfehler protokolliert. Es wurden also insgesamt 165 Versuche durchgeführt, von denen die 33 besten im folgenden Balkendiagramm 12.5 (Seite 106) gegenüber gestellt werden.

Die Ergebnisse zeigen, dass das ZrP(14)-Modell mit 63 HN den niedrigsten Validierungsfehler von allen Durchläufen produziert hat , mit MSE = 0,00462, und zwar bei 130 Trainingsiterationen. Im Vergleich zum besten Validierungsfehler mit MSE = 0,00465, den das ZrP(14)-Modell mit 16 HN bisher produziert hatte, ist die Verbesserung minimal. Durch die Steigerung der Komplexität konnte das allgemeine Underfitting der Problemstellung der ZrP-Modelle also nicht behoben werden. Der niedrigste erzielte Validierungsfehler ist, im Vergleich zum niedrigsten Validierungsfehler des KoP(7)-Modells, immer noch knapp 20 mal größer. Somit scheidet auch das letzte ZrP-Modell aus der Modellmenge aus.



	50	60	70	80	90	100	110	120	130	140	150
16 HN	0,00469	0,00468	0,00467	0,00468	0,00467	0,00465	0,00469	0,00469	0,00467	0,00467	0,00469
63 HN	0,00468	0,00468	0,00470	0,00467	0,00470	0,00471	0,00465	0,00469	0,00462	0,00463	0,00466
168 HN	0,00468	0,00467	0,00463	0,00470	0,00464	0,00470	0,00472	0,00467	0,00469	0,00466	0,00469

Abbildung 12.5.: ZrP(14)-Optimierung: Niedrigste Validierungsfehler von KNN mit unterschiedlicher Anzahl Hidden-Neuronen (HN) in Abhängigkeit zur Anzahl Trainingsiterationen

12.2.2. Optimierung des kausalen Modells KaP

- Steigerung der Netzkomplexität durch Hinzunahme weiterer Hidden-Neuronen.
- Ermittlung der idealen Anzahl zu laufender Trainingsiterationen durch Erhöhung der Iterationen.

Auch hier wurde die bisherige Anzahl Hidden-Neuronen (HN) aufgrund der Empfehlungen von Bailey und Thompson (1990) und Heaton (2011)/Moustafa (2011) erhöht. Von 15 auf 59 und 158.

Die bisherigen Ergebnisse zeigten, dass der niedrigste Validierungsfehler des KaP-Modells nach 100 Trainingsiterationen erzielt wurde, dass die Kurve des VE aus Abbildung 12.3 (Seite 101) aber, zwischen 2500 und 10000 Trainingsiterationen, eine deutliche Tendenz nach unten aufwies. Es soll nun überprüft werden, wie sich der VE mit weiterer Steigerung der Trainingsiterationen im Vergleich zu den 100er Ergebnissen, verhält.

Der daraus resultierende Versuchsaufbau lässt sich folgender Tabelle entnehmen, wobei auch hier die farbig markierten Felder wieder die Kombination mit dem jeweils niedrigsten Validierungsfehler angeben:

Anzahl Iterationen	15 HN (bisher)	59 HN (Optimierung 1)	158 HN (Optimierung 2)
100			
500			
2500			
5000			
10000			
50000			
100000			

Es ergeben sich also 3 Versuchsreihen, in denen jeweils ein KaP-Modell mit einer der dargestellten Anzahl Hidden-Neuronen (15, 59 oder 158) mit 7 verschiedenen Stopkriterien (100 - 100000 Trainingsiterationen) trainiert und validiert wird. Das Validierungsset ist das gleiche wie bei den vorangegangenen Analysen, so dass die Ergebnisse vergleichbar bleiben. Auch bei diesem Versuchsaufbau wurde jede Versuchsreihe wieder mit 5 verschiedenen zufälligen Startgewichten durchgeführt und das Ergebnis mit dem niedrigsten Validierungsfehler protokolliert. Es wurden also insgesamt 105 Versuche durchgeführt, von denen die 21 besten im folgenden Balkendiagramm 12.6 (Seite 107) gegenüber gestellt werden.

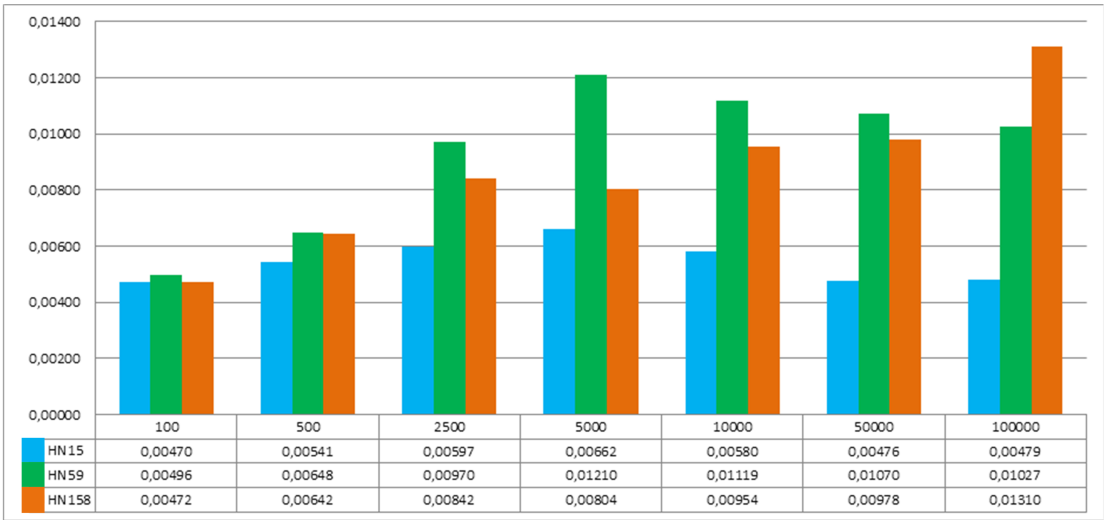


Abbildung 12.6.: KaP-Optimierung: Niedrigste Validierungsfehler von KNN mit unterschiedlicher Anzahl Hidden-Neuronen (HN) in Abhängigkeit zur Anzahl Trainingsiterationen

Die Ergebnisse zeigen, dass die Komplexitätssteigerung über die Erhöhung der Hidden-Neuronen keine Verbesserung des Validierungsfehlers bewirken konnte. Die Ergebnisse des KaP-Modells mit den bisherigen 15 HN liefert durchweg die besten Ergebnisse. Dies bedeutet, dass dem allgemeine Underfitting der Problemstellung des KaP-Modells durch eine Komplexitäts-

steigerung nicht entgegengewirkt werden konnte. Die Ergebnisse des KNN mit 15 HN zeigen jedoch weiterhin eine leicht fallende Tendenz mit steigender Anzahl an Trainingsiterationen. Um festzustellen, ob sich der VE des HN15-Modells weiter senken lässt wurden dieses KNN nochmals mit gesteigerten Trainingsiterationen trainiert und validiert - mit 150000 und 200000 Iterationen. Die Ergebnisse werden in folgender Tabelle präsentiert und zeigen keine weitere Verbesserung. Das KaP-Modell mit 15HN ist also weiterhin sehr robust in Bezug auf das Overfitting von verrauschten Daten, weist jedoch nach wie vor einen Validierungsfehler auf, der knapp 20 mal höher ist, als der des KoP(7)-Modells. Somit scheidet auch das KaP-Modell aus der Modellmenge aus.

	15 HN (bisher)
150000	0.004780
200000	0.004944

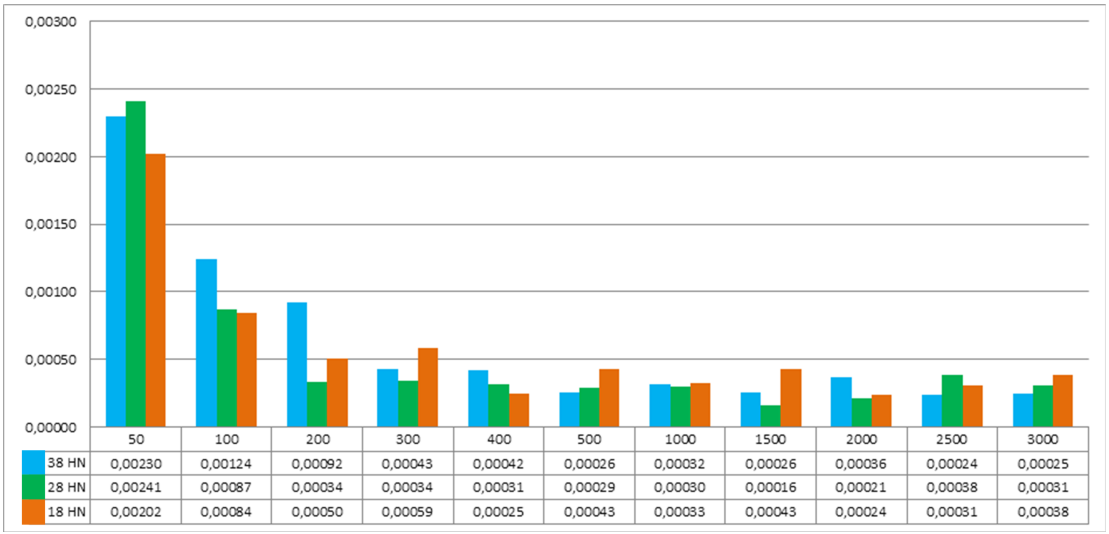
12.2.3. Optimierung des kombinierten Modells KoP(7)

- Reduzierung der Netzkomplexität durch Verringerung der Anzahl Hidden-Neuronen
- Ermittlung der idealen Anzahl zu laufender Trainingsiterationen durch Early-Stopping.

Die bisherige Anzahl der Hidden-Neuronen (HN), von 38, wurde auch im KoP-Fall über die Empfehlung von Masters (1993) errechnet. Bei der Geometrische-Pyramiden-Regel von Masters handelt es sich um die geringste in der Literatur gefundene Empfehlung. Masters weist jedoch ausdrücklich darauf hin, dass eine weitere Reduzierung der HN jederzeit vorgenommen werden kann, wenn es sinnvoll erscheint. Die weitere Reduzierung der Hidden-Neuronen wurde somit ohne Literaturvorgabe vorgenommen - und zwar in zwei Schritten um jeweils 10 Neuronen. Der Versuchsaufbau kann der folgende Tabelle entnommen werden, die jeweils niedrigsten Validierungsfehler sind wieder farblich hervorgehoben:

Anzahl Iterationen	38 HN (bisher)	28 HN (Optimierung 1)	18 HN (Optimierung 2)
50			
100			
200			
300			
400			
500			
1000			
1500			
2000			
2500			
3000			

Es wurden also 3 Versuchsreihen mit 38, 28 und 18 HN durchgeführt. Die Anzahl der zu durchlaufenden Trainingsiterationen bewegt sich zwischen 50 und 3000 Iterationen und soll das bestmögliche Stopkriterium im Rahmen der Early-Stopping Strategie ermitteln. Das Validierungsset ist das gleiche wie bei den vorangegangenen Analysen, so dass die Ergebnisse vergleichbar bleiben. Auch bei diesem Versuchsaufbau wurde jede Versuchsreihe wieder mit 5 verschiedenen zufälligen Startgewichten durchgeführt und das Ergebnis mit dem niedrigsten Validierungsfehler protokolliert. Es wurden also insgesamt 165 Versuche durchgeführt, von denen die 33 besten im folgenden Balkendiagramm 12.7 (Seite 109) gegenüber gestellt werden.



	50	100	200	300	400	500	1000	1500	2000	2500	3000
38 HN	0,00230	0,00124	0,00092	0,00043	0,00042	0,00026	0,00032	0,00026	0,00036	0,00024	0,00025
28 HN	0,00241	0,00087	0,00034	0,00034	0,00031	0,00029	0,00030	0,00016	0,00021	0,00038	0,00031
18 HN	0,00202	0,00084	0,00050	0,00059	0,00025	0,00043	0,00033	0,00043	0,00024	0,00031	0,00038

Abbildung 12.7.: KoP(7)-Optimierung: Niedrigste Validierungsfehler von KNN mit unterschiedlicher Anzahl Hidden-Neuronen (HN) in Abhängigkeit zur Anzahl Trainingsiterationen

Die Ergebnisse zeigen, dass eine Reduzierung der Anzahl HN von 38 auf 28 eine deutliche Verbesserung des Validierungsfehlers bewirkt hat. Das KoP(7)-Modell mit 28HN und 1500 Trainingsiterationen hat einen Validierungsfehler von 0,00016 erreicht, der somit um 36% besser ist als der bisher niedrigste von 0,00025 des KoP(7)-Modells mit 38 HN und 2500 Trainingsiterationen. Eine weitere Reduzierung um 10 HN auf 18 HN brachte jedoch keine weiter Verbesserung.

Das KoP(7)-Modell mit 28 Hidden-Neuronen hat bei einem Stopkriterium von 1500 Trainingsiterationen somit den mit Abstand niedrigsten Validierungsfehler aller im Rahmen dieser Arbeit durchgeführten Versuche erzielt. Dieses Modell wird im Folgenden mit *KoP(7)-28HN-1500IT* bezeichnet.

12.3. Auswertung und Test des besten Modells

Dieser Abschnitt wird den praktischen Teil dieser Arbeit abschließen und zeigen, was die Trainings- und Validierungsergebnisse des KoP(7)-28HN-1500IT konkret für die Bedarfsprognose von Laugenbrötchen bedeuten. Außerdem wird das Modell gegen das Testset getestet und die Testergebnisse präsentiert.

12.3.1. Auswertung Validierungsfehler

Wie bereits in der Einleitung dieses Kapitels erwähnt, ist der bisher präsentierte Validierungsfehler ein Mittelwert aus den Validierungsfehlern der drei Output-Werte. Hier soll nun einerseits gezeigt werden, wie gut das beste Modell die drei Prognosegegenstände Abverkauf (A), diskontierter Abverkauf (DA) und letzte Abverkaufszeit (LAZ) vorhersagen kann und andererseits, was ein Validierungsfehler von 0,00016, berechnet auf den normalisierten Output-Werten, konkret über die Prognosegüte aussagt.

Folgende Tabelle splittet den Validierungsfehler des KoP(7)-28HN-1500IT-Modells in seine drei Bestandteile auf: MSE auf den Abverkaufszahlen (A), den diskontierten Abverkaufszahlen (DA) und der letzten Abverkaufszeit (LAZ). In der zweiten Spalte stehen die Validierungsfehler, die auf den Werten berechnet wurden, die das KNN am Ende der Validierungsphase ausgibt - also den normalisierten Werten. Der in der dritten Spalte präsentierte de-normalisierte Validierungsfehler entsteht, indem die Output-Werte mit der Formel 11.1 (Seite 84) de-normalisiert werden, bevor der Validierungsfehler berechnet wird. Und die Werte aus der letzten Spalte ergeben sich, indem die Quadratwurzel der de-normalisierten Validierungsfehler berechnet wird, um die der MSE-Berechnung in 7.4 (Seite 41) zugrundeliegende Quadrierung wieder auszugleichen.

Prognose-gegenstand	Validierungs-fehler (**norm**)	Validierungs-fehler (**de-norm**)	Mittlere Abweichung
A	0,00011	2,55	1,60
DA	0,00000	0,00	0,00
LAZ	0,00038	294,05	17,15
Mittelwert	0,00016	98,86706	-

Konkret bedeutet dies, dass die Abverkaufszahlen, die das KoP(7)-28HN-1500IT prognostiziert, im Mittel um 1,6 Laugenbrötchen vom Idealwert abweichen. Bei der Prognose der letzten Abverkaufszeit verfehlt das KNN den Idealwert im Schnitt um 17 Minuten. Für den diskontierten Abverkauf wurde sowohl von diesem als auch von allen anderen Modellen ein Validierungsfehler von 0 ermittelt. Dies resultiert daraus, dass in der gesamten Datenbasis leider kein Daten-

satz mit diskontiertem Abverkauf vorhanden war[7]. Alle vorhandenen Verkaufszahlen beziehen sich auf Verkäufe zum regulären Preis, so dass der zu prognostizierende DA stets 0 war und dadurch von allen KNN-Modellen zuverlässig prognostiziert werden konnte. Die Eignung des KoP(7)-28HN-1500IT-Modells zur Prognose von DA kann somit durch diese Arbeit nicht beantwortet werden.

Die folgenden zwei Abschnitte werden einen detaillierten Überblick über die Prognoseergebnisse für A und LAZ auf dem Validierungsset liefern.

Prognostizierte Abverkaufszahlen (A) auf Validierungsset

Im Folgenden wird die Prognosequalität des KoP(7)-28HN-1500IT-Modells hinsichtlich der Abverkaufszahlen begutachtet. Dazu wurden zuerst alle Prognosen und Zielwerte der 168 Datenpaare des Validierungssets gemäß der in Abschnitt 11.3.1 vorgestellten Formel de-normalisiert. Die Differenz aus (*Prognose* - *Zielwert*) ergibt die tatsächliche Abweichung zwischen der prognostizierten Abverkaufszahl von Laugenbrötchen und der tatsächlich verkauften Menge dieses Artikels für den jeweiligen Prognosezeitpunkt. Abbildung 12.8 (Seite 112) zeigt die Verteilung dieser Abweichung. Auf der x-Achse ist die Höhe der Abweichung aufgetragen und auf der y-Achse die Anzahl der Prognosen, die diese Abweichung aufweisen. Die maximale Abweichung zwischen Prognose und Zielwert bewegt sich zwischen -4 und +2. Es wurde also maximal ein Abverkauf von 2 Laugenbrötchen zu viel bzw. 4 zu wenig vorhergesagt, was als ein sehr gutes Ergebnis zu werten ist. Vor allem, weil eine genaue Betrachtung der Abweichungsverteilung ergibt, dass 80% aller Prognosen (134 von 168 Prognosen) eine maximale Abweichung von lediglich 2 Brötchen aufweisen[8], was mit der mittleren Abweichung von 1,6 Brötchen, die über den MSE berechnet wurde, übereinstimmt.

Im Vergleich dazu zeigt Abbildung 12.9 (Seite 112) die Fehlerverteilung der Prognoseergebnisse des besten ZrP- und KaP-Modells (ZrP(14)-63HN-130IT und KaP-15HN-100IT). Hier bewegt sich die Fehlerdifferenz zwischen +19 und -19 beim ZrP-Modell und zwischen +27 und -19 beim KaP-Modell.

In Ermangelung eines Vergleichs mit der Prognose, die das Prognoseverfahren beim Kunden ermittelt hätte, sollen die Ergebnisse des KoP(7)-28HN-1500IT-Modells zumindest mit einem rudimentären Benchmark verglichen werden, um auszuschließen, dass ein ähnlich gutes Ergebnis mit einer naiven Mittelwertbildung erzielt werden könnte. Der Benchmarkwert ergibt sich aus dem Mittelwert über die Abverkaufszahlen der jeweils letzten 7 bzw. 14 Tage, vom Ermittlungszeitpunkt aus gerechnet. Abbildung 12.10 (Seite 113) stellt die Fehlerverteilung der 3

[7] Die Versuchsreihen wurden aber trotzdem darauf ausgelegt alle drei Prognosegegenstände zu prognostizieren, weil die Möglichkeit bestand, dass im Laufe der Bearbeitungszeit dieser Arbeit weitere Daten zur Verfügung gestellt werden, die dann möglicherweise Datensätze mit diskontiertem Abverkauf enthalten hätten. Außerdem sollen mögliche Folgearbeiten oder firmeninterne Weiterführungen auf den hier implementierten Versuchsreihen aufsetzen können.

[8] Im Einzelnen: 39 Prognosen mit einer Abweichung von -2; 55 Prognosen mit einer Abweichung von -1; 40 Prognosen mit einer Abweichung von 0

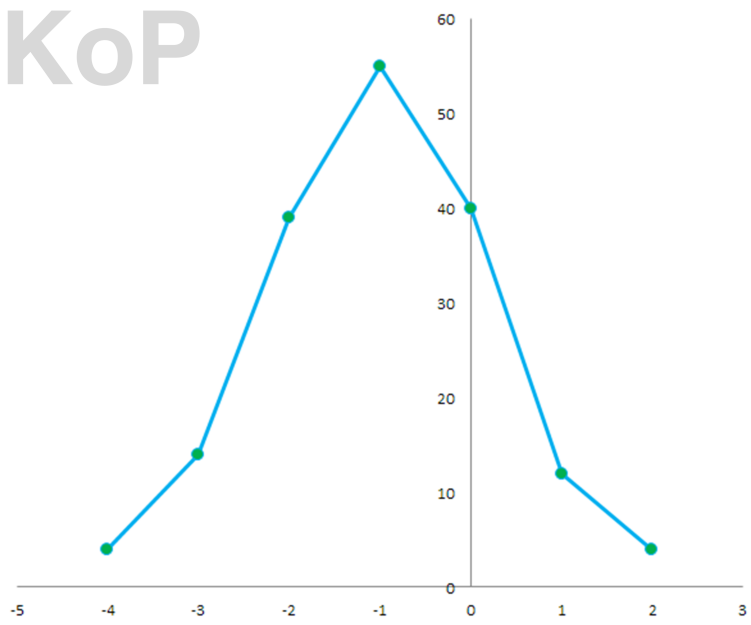


Abbildung 12.8.: KoP(7)-28HN-1500IT: Verteilung der Abweichung zwischen ermittelter Prognose und Zielwert auf Abverkaufszahlen (A) des Validierungssets

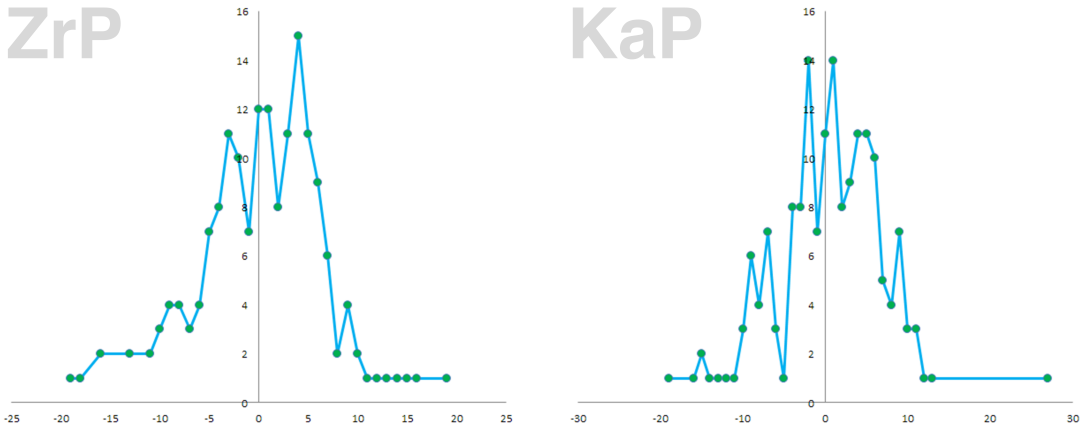


Abbildung 12.9.: ZrP(14)-63HN-130IT und KaP-15HN-100IT: Verteilung der Abweichung zwischen ermittelter Prognose und Zielwert auf Abverkaufszahlen (A) des Validierungssets

Werte gegenüber. Beide Mittelwerte schneiden deutlich schlechter ab als die Prognoseergebnisse des KoP(7)-28HN-1500IT-Modells und werden selbst von den Ergebnissen des ZrP- und KaP-Modells klar überboten.

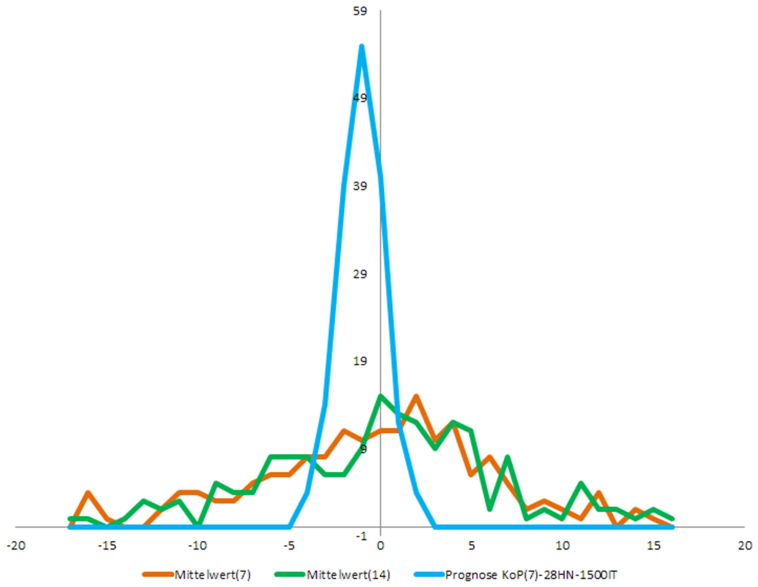


Abbildung 12.10.: Verteilung der Abweichung zwischen ermittelter Prognose, dem Mittelwert über 7 und 14 Tage und dem Zielwert auf Abverkaufszahlen (A) des Validierungssets

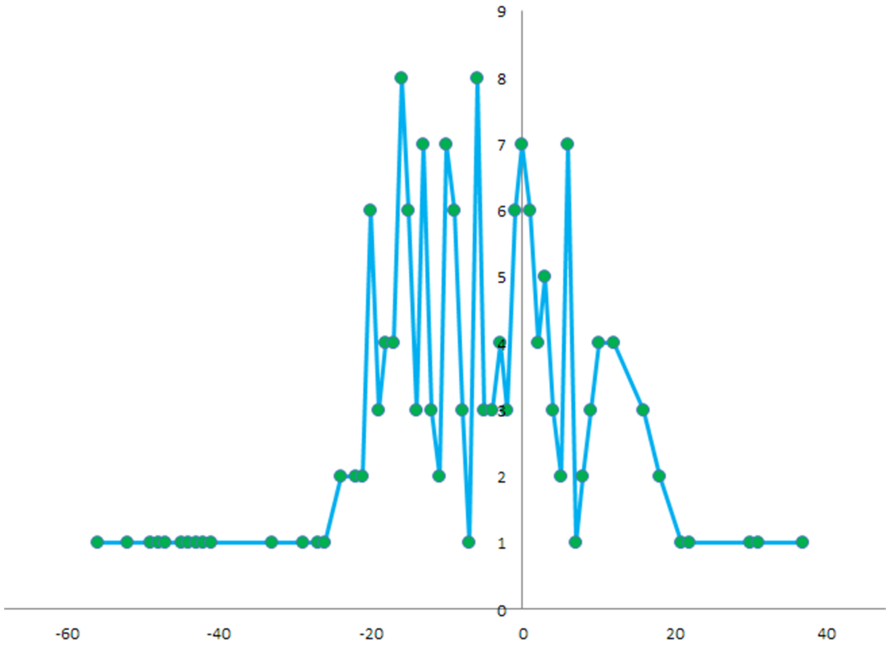


Abbildung 12.11.: KoP(7)-28HN-1500IT: Verteilung der Abweichung zwischen ermittelter Prognose und Zielwert von LAZ auf Validierungsset

Prognostizierte letzte Abverkaufszeiten (LAZ) auf Validierungsset

Wie bereits vorweg genommen, beträgt der normalisierte Validierungsfehler des KoP(7)-28HN-1500IT-Modells auf den LAZ-Werten des Validierungssets 0,00038 und der de-normalisierte 294,05. Gemäß MSE-Umrechnung bedeutet dies eine mittlere Abweichung von 17,15 Minuten vom Idealwert. Die Abweichungsverteilung in Abbildung 12.11 (Seite 113) zwischen prognostizierter LAZ und tatsächlicher LAZ bestätigt dies. 76% (128 von 168 Prognosen) aller Prognosen auf dem Validierungsset weichen maximal +/- 17 Minuten vom Idealwert ab, was als ein sehr gutes Ergebnis gewertet werden kann.

Warum die Abweichung bei 65% der Prognosen (110 von 168 Prognosen) negativ ist, also eine LAZ vorhergesagt wurde, die früher ist als die tatsächliche, ließ sich anhand der vorhandenen Daten nicht ermitteln, sollte an dieser Stelle aber nicht unerwähnt bleiben.

12.3.2. Auswertung Testfehler

Die guten Validierungsergebnisse des KoP(7)-28HN-1500IT-Modells müssen nun noch durch ein Testset überprüft werden. Dadurch soll, wie in Abschnitt 8.5 beschrieben, überprüft werden, ob die erlernte Abbildungsfunktion versehentlich auf das Validierungsset optimiert wurde und nicht gut genug generalisiert, um auch auf gänzlich unbekannten Daten zufriedenstellende Vorhersagen treffen zu können. Das Testset besteht aus 48 Datenpaaren und enthält, wie in Abbildung 8.10 (Seite 61) angedeutet, die chronologisch neuesten Daten. Das Testset enthält also zwei Datenpaare jeder der 24 Filialen. Um zu vermeiden, dass im Testset von jeder Filiale jeweils die letzten 2 Tage enthalten sind und somit lediglich die Prognosequalität von Samstag, dem 23.06.2012 und Sonntag, dem 24.06.2012 getestet wird, wurden aus den Daten der letzten 7 Tage jeder Filiale zwei Tage zufällig ermittelt.

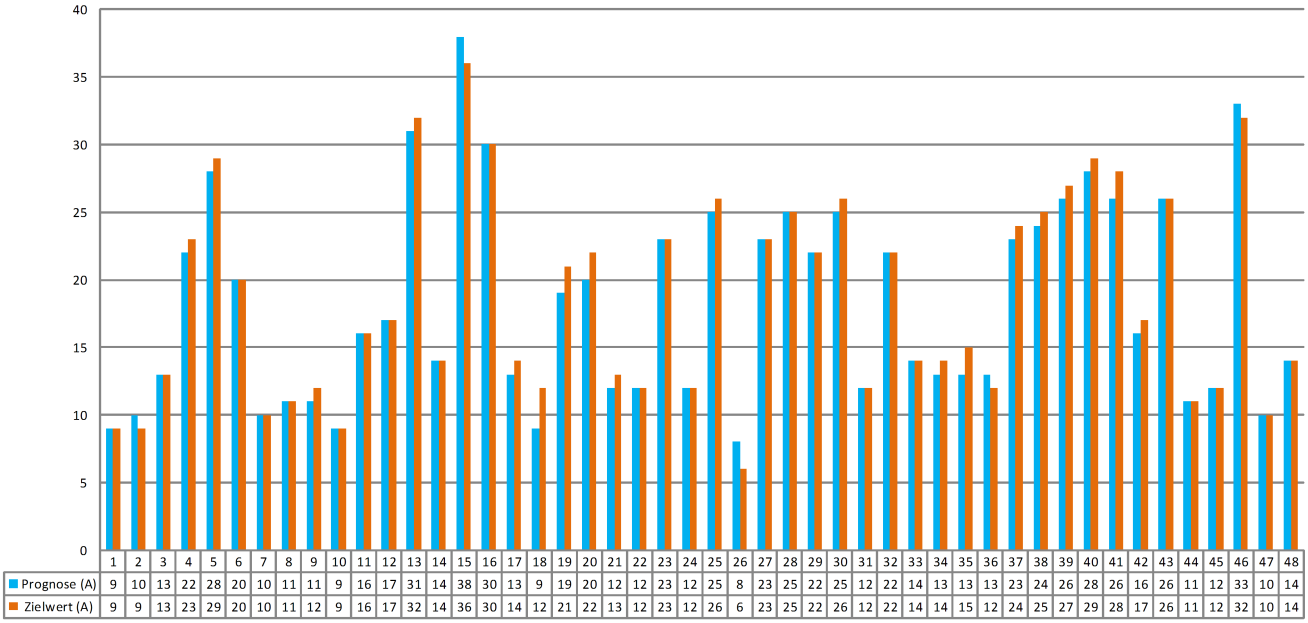
Die Abbildungsfehler auf dem Testset werden in der folgenden Tabelle dargestellt und in den kommenden zwei Abschnitten diskutiert:

Prognose-gegenstand	Test-fehler (**norm**)	Test-fehler (**de-norm**)	Mittlere Abweichung
A	0,00005	1,04	1,02
DA	0,00000	0,00	0,00
LAZ	0,00069	522,52	22,86
Mittelwert	0,00025	174,52	-

Prognostizierte Abverkaufszahlen (A) auf Testset

Die über den MSE berechnete mittlere Abweichung zwischen der Prognose und dem Zielwert von Abverkaufszahlen ist auf dem Testset mit 1,02 Laugenbrötchen noch geringer als auf dem

Validierungsset - dort betrug die Abweichung im Mittel 1,6. In Abbildung 12.12 (Seite 115) wird die konkrete Anzahl prognostizierter und idealer Werte erstmals[9] einander gegenüber gestellt und Abbildung 12.13 (Seite 115) zeigt die Verteilung der Abweichung.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Prognose (A)	9	10	13	22	28	20	10	11	11	9	16	17	31	14	38	30	13	9	19	20	12	12	23	12	25	8	23	25	22	25	12	22	14	13	13	13	23	24	26	28	26	16	26	11	12	33	10	14
Zielwert (A)	9	9	13	23	29	20	10	11	12	9	16	17	32	14	36	30	14	12	21	22	13	12	23	12	26	6	23	25	22	26	12	22	14	14	15	12	24	25	27	29	28	17	26	11	12	32	10	14

Abbildung 12.12.: Gegenüberstellung prognostizierter und idealer Abverkaufszahlen (A) auf Testset

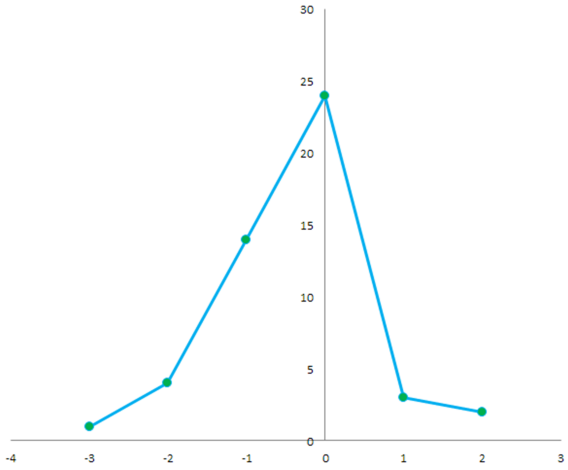


Abbildung 12.13.: Verteilung der Abweichung prognostizierter und idealer Abverkaufszahlen (A) auf Testset

[9] Diese Art der Gegenüberstellung war auf den Validierungsdaten nicht möglich, da ein Balkendiagramm oder selbst ein Liniendiagramm mit 168 Datenpunkten zu unübersichtlich ist.

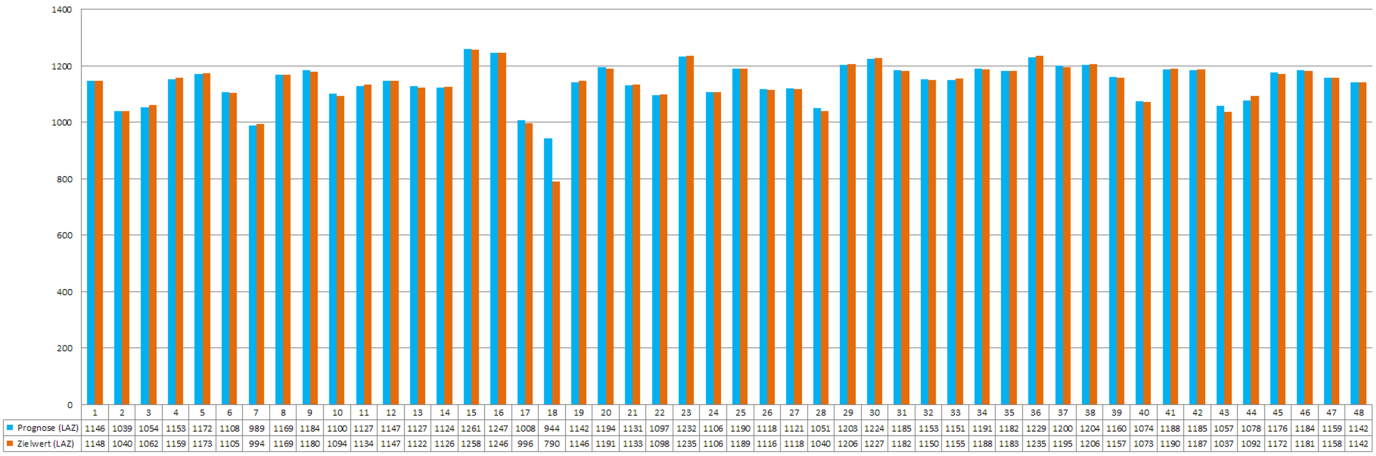
Die erzielten Prognoseergebnisse für den Abverkauf (A) auf dem Testset lassen somit folgende Schlussfolgerung zu: Obwohl die Ergebnisse auf dem Validierungsset Grundlage für die Optimierung des KoP(7)-Modells waren, overfittet das KoP(7)-28HN-1500IT die Validierungsdaten bzgl. A nicht und ist in der Lage die guten Validierungsergebnisse auf dem Testset zu bestätigen.

Prognostizierte letzte Abverkaufszeiten (LAZ) auf Testset

Die über den MSE berechnete mittlere Abweichung zwischen der Prognose und dem Zielwert der letzten Abverkaufszeit (LAZ) auf dem Testset beträgt 22,86 Minuten und ist somit höher als auf dem Validierungsset - dort betrug die Abweichung im Mittel 17,15. Wenn man sich nun die konkrete Gegenüberstellung prognostizierter und idealer Werte in Abbildung 12.14 (Seite 116) und die Abweichungsverteilung in 12.15 (Seite 117) ansieht, wird man feststellen, dass das höhere Mittel aus einer einzigen sehr hohen Abweichung resultiert. In Testpaar 18 hat das KoP(7)-28HN-1500IT eine letzte Abverkaufszeit von 944 (15:44h) prognostiziert, wobei der Zielwert 790 (13:10h) war. Die Abweichung betrug also 154 Minuten und hat somit die Ermittlung der mittleren Abweichung über den MSE stark beeinflusst.

Ein möglicher Grund für diese vergleichsweise hohe Abweichung ist, dass von allen 2736 zur Verfügung stehenden letzten Abverkaufszeiten lediglich 28 kleiner als 800 (13:20h) waren und sich davon wiederum nur 20 im Trainingsset des KoP(7)-28HN-1500IT befanden. Dem KNN standen somit nicht genügend Datensätze mit einer so frühen letzten Abverkaufszeit zur Verfügung, als dass es eine sinnvolle Abbildungsfunktion für diese Fälle hätte ableiten können.

Die Prognoseergebnisse des KoP(7)-28HN-1500IT-Modells auf dem Testset bestätigt somit einerseits die guten Ergebnisse auf dem Validierungsset, zeigt aber andererseits, dass ein möglichst vollständiges Trainingsset Voraussetzung für gute Prognosen im Allgemeinen ist.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Prognose (LAZ)	1146	1039	1054	1153	1172	1108	989	1169	1184	1100	1127	1147	1127	1124	1261	1247	1008	944	1142	1194	1131	1097	1232	1106	1190	1118	1121	1051	1203	1224	1185	1153	1151	1191	1182	1229	1200	1204	1160	1074	1188	1185	1057	1078	1176	1184	1159	1142
Zielwert (LAZ)	1148	1040	1062	1159	1173	1105	994	1169	1180	1094	1134	1147	1122	1126	1258	1246	996	790	1146	1191	1133	1098	1235	1106	1189	1116	1118	1040	1206	1227	1182	1150	1155	1188	1183	1235	1195	1206	1157	1073	1190	1187	1037	1092	1172	1181	1158	1142

Abbildung 12.14.: Gegenüberstellung prognostizierter und idealer LAZ auf Testset

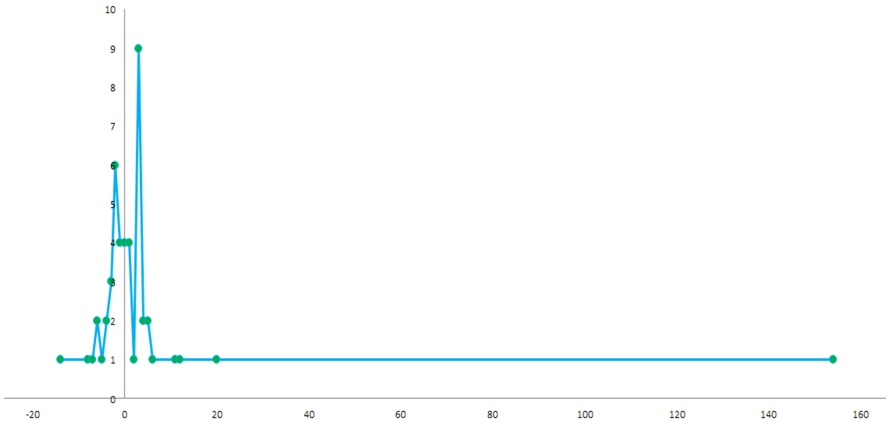


Abbildung 12.15.: KoP(7)-28HN-1500IT: Verteilung der Abweichung zwischen ermittelter Prognose und Zielwert von LAZ auf Testset

Die erzielten Prognoseergebnisse für die letzte Abverkaufszeit (LAZ) auf dem Testset lassen somit folgende Schlussfolgerungen zu:

- Obwohl die Ergebnisse auf dem Validierungsset Grundlage für die Optimierung des KoP(7)-Modells waren, overfittet das KoP(7)-28HN-1500IT die Validierungsdaten bzgl. LAZ nicht und ist in der Lage die guten Ergebnisse des Validierungsset auf dem Testset zu bestätigen.
- Die verwendete Datenbasis war nicht ausreichend repräsentativ für eine genauere Prognose früher LAZ.

12.4. Zusammenfassung der Versuchsergebnisse

Wie dem aufmerksamen Leser nicht entgangen sein dürfte, lag die Größe, sowohl des Validierungssets als auch des Testsets, weit unter der in Kapitel 8 angegebenen prozentualen Verteilung von 20% aller Datenpaare für das Validierungsset und 10% für das Testset. Bei 2544 verfügbaren Datenpaaren für das KoP(7)-Modell ensprechen 168 Validierungspaare lediglich 7% und 48 Testpaare gar nur 2%.

Hintergrund für die Wahl dieser kleinen Validierungs- und Testsets ist die Tatsache, dass sich hinter den 2544 Datenpaaren lediglich 114 Paare für jede der 24 Filialen verbergen. Von diesen 114 müssen nochmal 8 gestrichen werden, weil sie das 7-tägige Zeitfenster in die Vergangenheit nicht erfüllen konnten und der letzte Tag jeder Filiale keinen Zielwert besitzt. Somit bleiben den KoP(7)-Modellen 106 Datenpaare pro Filiale. Wenn davon nun 21 Datenpaare (20%) für das Validierungsset und 11 (10%) für das Testset beiseite gelegt würden, dann müssten die KoP(7)-Modelle die gesuchte Abbildungsfunktion anhand von 74 Datenpaaren pro Filiale erlernen und das wurde als zu wenig erachtet. Vor allem weil die Validierungs- und Testset für

alle getesteten Modelle gleich sein mussten, um die Vergleichbarkeit der Modelle zu gewährleisten. Bei den KoP(14) und KoP(28)-Modellen (und natürlich auch ZrP(14) und ZrP(28)) hätten dann respektive nur noch 69 und 59 Trainingspaare pro Filiale zur Verfügung gestanden und das wären definitiv zu wenige gewesen (siehe zur Erinnerung Tabelle 12.2 (Seite 99)).

Der gewählte Kompromiss bestand nun darin, aus den 114 Tagen jeder Filiale lediglich 7 Tage für ein Validierungsset und 2 Tage für ein Testset auszuwählen und diese zur Überprüfung aller verwendeten Modelle einzusetzen. Die Konsequenz dieser Entscheidung ist, dass die geringen Abweichungen zwischen den prognostizierten und tatsächlichen Werten des KoP(7)-28HN-1500IT-Modells zwar positiv zu bewerten sind, aber keinen abschließenden Beweis für die Prognosegüte des Modells erbringen können. Die Validierungs- und Testdaten sind in dieser Zusammenstellung weder umfangreich noch repräsentativ genug, um die Prognosegenauigkeit zuverlässig überprüfen zu können. Dafür müssten in einer Folgearbeit oder firmenintern Testreihen mit einer größeren Datenbasis durchgeführt werden.

Unangetastet von der Relativierung der geringen Validierungs- und Testfehler bleibt jedoch die Feststellung, dass die KoP-Modelle gegenüber den ZrP- und KaP-Modellen die bessere Prognosegüte aufweisen konnten. Die ermittelten Abbildungsfunktionen aller getesteten Modelle wurden auf demselben Validierungs- und Testset getestet, so dass die deutlich geringeren Abweichungen der KoP-Modelle definitiv eine Bevorzugung dieser Modelle rechtfertigt. Bei einer deutlichen Vergrößerung der Datenbasis wäre jedoch eine weitere Analyse zur Ermittlung der besten Anzahl an Hidden-Neuronen und der Höhe der optimalen Anzahl an Trainingsiterationen ratsam.

Den Modellen der Zeitreihen- und kausalen Prognose soll mit dieser Arbeit keine inhärente Untauglichkeit zur Bedarfsprognose von Backwaren attestiert werden, aber es wurde gezeigt, dass sie unter den gegebenen Umständen keine zufriedenstellende Prognose ermitteln konnten. Dazu beigetragen hat eventuell eine für Zeitreihenprognosen möglicherweise zu kleine Datenbasis und eine Auswahl an Einflussvariablen, die ohne Domänenexperten getroffen wurde und für eine rein kausale Prognose möglicherweise noch erweitert/optimiert werden müsste. Auch diese Fragen müssten in einer Folgearbeit oder firmenintern beantwortet werden.

Teil V.

Schlussbetrachtungen

13. Fazit

Ein Hauptziel dieser Bachelorarbeit war zu untersuchen, ob sich KNN eignen den Bedarf an Backwaren in Filialen eines Einzelhandelsdiscounters für den Folgetag zuverlässig vorherzusagen. Konkret bedeutet dies zu verstehen, wie die Problemstellung und das KNN modelliert werden müssen, damit genau die Informationen zur Verfügung gestellt werden anhand derer das KNN in der Lage ist auf den Prognosegegenstand zu schließen. Dazu gehörte die Einarbeitung in die Konzepte der Bedarfsprognose, in die allgemeinen Grundlagen der Modellierung von Bedarfsprognosen im Handel und in die Konzepte und Funktionsweisen KNN. Die theoretischen Grundlagen dafür wurden im Teil II dieser Arbeit gelegt. Zusammen mit den praktischen Grundlagen, die in Teil III eingeführt wurden, konnte dieses Wissen dann in Teil IV auf die vorliegende Problemstellung angewandt werden.

In den Kapiteln 5 (Abschnitt 5.2), 8 (Abschnitt 8.1) und 11 (Abschnitt 11.2.2) wurden 3 mögliche Modellierungsformen für Bedarfsprognosen erarbeitet, die in Kapitel 12 konkret auf die Bedarfsprognose von Laugenbrötchen in 24 Filialen im Raum Prag (Tschechien) angewandt und miteinander verglichen wurden: Zeitreihenprognosemodelle, kausale Prognosemodelle und kombinierte Prognosemodelle.

Die Ergebnisse der Versuchsreihen aus Kapitel 12 zeigen, dass sowohl reine Zeitreihenmodellierungen als auch reine kausale Modellierungen deutlich schlechtere Prognoseergebnisse liefern als die kombinierte Modellierungsform. Bei der kombinierten Prognose werden dem KNN sowohl historische Realisierungen *des* Prognosegegenstandes als auch Einflussvariablen, die ausreichende Informationen *über* den Prognosegegenstand enthalten, zur Verfügung gestellt. Das Modell, das die besten Prognoseergebnisse lieferte, hatte sowohl die letzten 7 Realisierungen des Prognosegegenstands zur Verfügung als auch die letzten 7 Realisierungen der Einflussvariablen zusammen mit ihrer Ausprägung zum Prognosezeitpunkt (vgl. Abbildung 11.4, Seite 83). Kombinierte Modelle mit einem Zeitfenster von mehr als 7 Tagen in die Vergangenheit (getestet mit 14 und 28 Tagen) konnten keine Verbesserung der Prognose bewirken.

Die Prognosegüte der getesteten Modelle wurde einerseits anhand der Abweichung zwischen der gelieferten Prognose und dem vorgegebenen Zielwert bewertet und andererseits mit einem Mittelwert-Benchmark verglichen. Die ursprüngliche Zielsetzung die Prognosen der KNN gegen die Prognoseergebnisse des bisher eingesetzten Prognoseverfahrens der produktiven Softwarelösung zu validieren, konnte nicht realisiert werden. Die Analyse der produktiven Softwarelösung (zusammengefasst in Kapitel 10) ergab, dass die Ergebnisse beider Lösungen zum jetzigen Zeitpunkt nicht vergleichbar sind. Die Vergleichbarkeit hätte nur durch eine Refaktorierung der bestehenden Softwarelösung hergestellt werden können, die den zeitlichen Rahmen

dieser Arbeit gesprengt und das eigentliche Ziel verfehlt hätte. Die durchgeführten Analysen der produktiven Softwarelösung resultierten in der Definition einer Teillösung (vgl. Abschnitt 10.3.4), die darauf ausgelegt ist die Höhe des Abverkaufs, die Höhe des diskontierten Abverkaufs und die letzte Abverkaufszeit von Laugenbrötchen für den Folgetag zu prognostizieren.

Auf den vorhandenen Trainings-, Validierungs- und Testdaten ermittelte das beste Modell, eine maximale Abweichung bei der Abverkaufszahl von +/- 3 Brötchen und bei der letzten Abverkaufszeit von +20 bis -69 Minuten. Die Höhe dieser Abweichungen ist sowohl bei den Abverkaufszahlen als auch bei der letzten Abverkaufszeit als ausgezeichnetes Ergebnis zu werten und setzt sich deutlich von den Ergebnissen der anderen Modelle, sowie des Mittelwert-Benchmarks, ab. Das Vorhersagen des diskontierten Abverkaufs konnte nicht sinnvoll erlernt werden, da in der Datenbasis keine Datensätze mit diskontiertem Abverkauf vorhanden waren.

Zusätzlich zur Ermittlung der besten Modellierungsform haben die Versuchsreihen in Kapitel 12 Aufschluss über die ideale Größe des historischen Zeitfensters, sowie über das beste Stopkriterium für das Training des gewählten KNN geliefert. Diese Ergebnisse stehen jedoch in direktem Zusammenhang mit der Größe der verwendeten Datenbasis und müssten auf einer größeren und neueren Datenbasis überprüft werden. Der letzte Abschnitt dieser Arbeit wird einen Ausblick auf die Versuche und Untersuchungen geben, die noch geleistet werden müssten, um die hier getroffenen Aussage bzgl. der Modellierung bestätigen zu können.

Zuvor aber noch eine paar Worte zur Handhabbarkeit der hier implementierten Prognose mit KNN im Umfeld einer klassischen Java Softwareentwicklung: Es steht außer Frage, dass die hier vorgestellte Implementierung einer Bedarfsprognose mit KNN deutlich komplexer ist, als das bisher implementierte Prognoseverfahren der Mittelwertbildung. Ein Einsatz von KNN als Prognoseverfahren sollte also nur in Erwägung gezogen werden, wenn sicher gestellt ist, dass die guten Prognoseergebnisse dieser Arbeit auch auf einer realistischeren Datenbasis Bestand haben und deutlich bessere Prognosen liefert als das bestehende System. Einen Großteil der Komplexität macht jedoch das Automatisieren der Beschaffung und Vorverarbeitung der Daten aus und dabei handelt es sich um eine gut handhabbare programmierhandwerkliche Herausforderung. Im Aufwand deutlich schwieriger einzuschätzen ist die Zeit die benötigt wird, um ausreichend Erfahrung mit der Interpretation und Auswertung der Ergebnisse zu sammeln. Und daran sollte auf keinen Fall gespart werden, denn der Erfolg des produktiven KNN hängt zu 100% von der korrekten Interpretation der Ergebnisse während der Entwicklungsphase ab.

14. Ausblick

Diese Arbeit hat viele Aspekt der Bedarfsprognostizierung von Backwaren mit KNN vorgestellt, erarbeitet und durchleuchtet. Gleichzeitig ist eine Liste mit Aufgaben und Fragen entstanden, die in einer Folgearbeit oder firmenintern bearbeitet und beantwortet werden müssten, um die hier gezeigten Ergebnisse bestätigen und verbessern zu können.

An erster Stelle steht das Trainieren und Validieren von KNN mit einer deutlich größeren Datenbasis als im Rahmen dieser Arbeit zur Verfügung stand. Wobei sich die Größe in erster Linie nicht auf die Breite der Daten (mehr Filialen), sondern auf die Länge des historischen Zeitraums bezieht. Es wäre erstrebenswert Trainingsdaten zu haben, die ein komplettes Kalenderjahr abdecken und Validierungs- und Testdaten die außerhalb dieses Jahres liegen.

Sollten die hier gelieferten Prognoseergebnisse nach Erweiterung der Datenbasis bestätigt werden können, sollte die in Abschnitt 10.3 skizzierte Refaktorierung der bisherigen Lösung durchgeführt werden, um feststellen zu können wie deutlich die Prognose durch KNN, den bisherigen Prognoseergebnissen überlegen ist. Wenn dieser Vergleich die Implementierung der Prognose durch KNN rechtfertigt, gibt es mehr zu tun, als in diesem Ausblick sinnvoll beschrieben werden kann. Die folgende Liste liefert jedoch erste Anhaltspunkte:

- Erweiterung der Prognose auf alle zu prognostizierenden Backwaren, indem entweder für jeden *Artikel* ein eigenes KNN trainiert und validiert wird oder der in Abschnitt 10.1 Alternativansatz umgesetzt wird. Für jede *Filiale* ein KNN entwerfen, das den Bedarf aller dort verkauften Backwaren prognostizieren kann.
- Ermittlung weiterer oder besserer Einflussfaktoren auf den Prognosegegenstand durch die Einschätzung von Domänenexperten.
- Durchführung einer Sensibilitätsanalyse nach (Kruse u. a., 2012, S. 77ff), durch die bestimmt wird, welchen Einfluss die gewählten Input-Variablen auf die Prognosequalität des KNN haben.

Teil VI.

Anhang

Glossar

Abverkauf (A) Die verkaufte Anzahl eines Artikels. Im Handel sind stündliche, tägliche, wöchentliche, monatliche und jährliche Abverkaufszahlen üblich. Ein möglicher Synonym ist: Absatz.

Backmengenempfehlung Die Grundlage der Backmengenempfehlung ist die Bedarfsprognose. Dieser prognostizierte Bedarf wird dann aber u.a. noch auf verschiedenen Zeitfenster verteilt, es werden Backblechbelegungen errechnet und Überhänge auf Zusatzblechen zusammengefasst. Am Ende dieses Prozesses, steht ein versandfertiges PDF mit allem nötigen Informationen für die einzelnen Filialleiter. Die Backmengenempfehlung ist nicht Gegenstand dieser Arbeit.

Bedarfsprognose Der Bedarf ist die am Markt tatsächlich auftretende Nachfrage nach Gütern. Die Bedarfsprognose ist somit die Vorhersage der insgesamt und maximal benötigten Menge eine Produktes am Markt zu einem zukünftigen Zeitpunkt.

Datenpaar Ein Datenpaar besteht genauso wie ein Trainingspaar aus einem Input-Vektor $\vec{x}$ und einem Output- bzw. Ziel-Vektor $\vec{y}$. Der Begriff Datenpaar lässt die Verwendung des Paares, im Vergleich zum Trainingspaar, jedoch offen. Ein Datenpaar kann sowohl im Training als auch während der Validierungs- und Testphase eines KNN eingesetzt werden.

diskontierter Abverkauf (DA) Die verkaufte Anzahl eines Artikels zu einem reduzierten (diskontierten) Preis. Im Einzelhandel ist es durchaus üblich verderbliche Waren ab einem bestimmten Zeitpunkt im Preis zu reduzieren, mit der Erwartung den Abverkauf dadurch zu steigern und zu beschleunigen.

Ermittlungszeitpunkt Der Tag an dem die Bedarfsprognose ermittelt wird. Bezeichnet immer einen Tag in der Vergangenheit oder den gegenwärtigen Tag. Meist als Tag *t* bezeichnet.

erwartete letzte Abverkaufszeit (eLAZ) Der Zeitpunkt (auf die Minute genau) für den der Abverkauf der letzten Einheit eines Artikels erwartet wird - bezogen auf einen Verkaufstag. Dieser Zeitpunkt wird für jeden Artikel im Vorwege bestimmt und dem Algorithmus zur Bedarfsannäherung zugrunde gelegt.

Gewicht Die Stärke der Verbindung zwischen zwei Neuronen wird durch ein Gewicht ausgedrückt. Je größer der Absolutbetrag des Gewichtes ist, desto größer ist der Einfluss eines

Neurons auf ein anderes Neuron. Das Wissen eines neuronalen Netzes ist in seinen Gewichten gespeichert. Ein möglicher Synonym ist: Verbindungsgewicht.

Hidden-Schicht Eine Menge von Neuronen, die sich zwischen Input- und Output-Schicht befinden und eine interne Repräsentation der Außenwelt beinhalten. In der Hidden-Schicht findet der Großteil der Rechen- und Lernleistung eines MLP statt.

Input-Schicht Eine Menge von Neuronen, die Eingaben empfangen und diese an die tieferen Schichten weiterleiten. Die Input-Schicht bildet somit die Eingabe-Schnittstelle zwischen Außenwelt und KNN.

Künstliches Neuronales Netz (KNN) Der Begriff *Neuronales Netz* beziehen sich auf das Neuronennetz des menschlichen Gehirns. Dieses dient als Analogie und Inspiration für die in Computern simulierten Künstlichen Neuronalen Netze. Der Begriff *Künstliches Neuronales Netz* dient also als Abgrenzung zum biologischen Vorbild.

letzte Abverkaufszeit (LAZ) Der Zeitpunkt (auf die Minute genau) an dem die letzte Einheit eines Artikels verkauft wurde - bezogen auf einen Verkaufstag.

Multilayer-Perceptron (MLP) Ein MLP ist in der Regel ein Vorwärtsgerichtetes (Feedforward) Neuronales Netz mit mindestens einer Hidden-Schicht.

Output-Schicht Eine Menge von Neuronen, die das Rechen/Lernergebnis der Hidden-Schicht(en) an die Außenwelt weitergeben. Die Output-Schicht bildet somit die Ausgabe-Schnittstelle zwischen KNN und Außenwelt.

Prognosezeitpunkt Der Tag für den die Bedarfsprognose ermittelt wird. Bezeichnet immer einen Tag in der Zukunft. Meist als Tag *t+1* bezeichnet.

Trainingsfehler (TE) Der Abbildungsfehler auf den Trainingsdaten.

Trainingspaar Jedes Trainingspaar besteht aus einem Input-Vektor $\vec{x}$ und einem Output- bzw. Ziel-Vektor $\vec{y}$. Alle Trainingspaare zusammengenommen bilden das Trainingsset. Während der Trainingsphase eines KNN, werden alle Trainingspaare mehrmals durchlaufen und das KNN versucht seine Verbindungsgewichte so anzupassen das bei gegebenem Input-Vektor der jeweilige Ziel-Vektor ausgegeben wird.

Validierungsfehler (VE) Der Abbildungsfehler auf den Validerungsdaten.

Literaturverzeichnis

[Alex 1998] Alex, Björn: *Künstliche neuronale Netze in Management-Informationssystemen: Grundlagen und Einsatzmöglichkeiten*. Wiesbaden : Gabler, 1998. -- ISBN 9783409135788

[Anderson 1995] Anderson, James A.: *An introduction to neural networks*. Cambridge and Mass : MIT Press, 1995. -- ISBN 978-0262011440

[Armstrong 2001] Armstrong, J. S.: Combining Forecasts. In: Armstrong, Jon S. (Hrsg.): *Principles of forecasting*. Boston and MA : Kluwer Academic, 2001, S. 1--19. -- ISBN 978-0792379300

[Auer 2013] Auer, Benjamin R. D.: Zeitreihe (Version 9). In: Gabler Verlag (Hrsg.): *Gabler Wirtschaftslexikon*. Wiesbaden : Springer-Verlag, 2013 (Gabler Wirtschaftslexikon). -- URL http://wirtschaftslexikon.gabler.de/Archiv/57589/zeitreihe-v9.html

[Bailey und Thompson 1990] Bailey, David L. ; Thompson, Donna: Developing neural-network applications. In: Zeichik, Alan L. (Hrsg.): *AI Expert* Bd. 9. San Francisco and CA and USA, 1990, S. 34--41

[Braun u. a. 1997] Braun, Heinrich ; Feulner, Johannes ; Malaka, Rainer: *Praktikum neuronale Netze*. 1. Berlin [u.a.] : Springer, 1997. -- ISBN 3540600302

[Callan 2003] Callan, Robert: *Neuronale Netze: Im Klartext*. München and and Harlow : Pearson Studium, 2003. -- ISBN 978-3-8273-7071-6

[Crone und Kourentzes 2009] Crone, S F. ; Kourentzes, N.: Forecasting Seasonal Time Series with Multilayer Perceptrons: An empirical Evaluation of Input Vector Specifications for deterministic Seasonality. In: *Proceedings of the 2009 International Conference on Data Mining, DMIN'09* (2009), S. 232--238

[Crone und Kourentzes 2010] Crone, S F. ; Kourentzes, N.: Naive Support Vector Regression and Multilayer Perceptron Benchmarks for the 2010 Neural Network Grand Competition (NNGC) on Time Series Prediction. In: *Proceedings of the International Joint Conference on Neural Networks, IJCNN'10* (2010)

[Crone 2010] Crone, Sven F.: *Neuronale Netze zur Prognose und Disposition im Handel*. 1. Wiesbaden : Gabler, 2010. -- ISBN 978-3834911742

[Crone u. a. 2011] Crone, Sven F. ; Hibon, Michèle ; Nikolopoulos, Konstantinos: Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. In: *International Journal of Forecasting* 27 (2011), Nr. 3, S. 635--660

[Duch und Jankowski 02.03.2001] DUCH, Włodzisław ; JANKOWSKI, Norbert: *Transfer functions: hidden possibilities for better neural networks*. Torun and Poland, Nicholas Copernicus University, Dissertation, 02.03.2001

[Eisenbach 2005] EISENBACH, Dominik: *Künstliche Neuronale Netze zur Prognose von Zeitreihen*, Westfälische Wilhelms-Universität Münster, Dissertation, 2005

[Fildes u. a. 2008] FILDES, R ; NIKOLOPOULOS, K ; SYNTETOS, A A. ; CRONE, S F.: Forecasting and operational research: A review. In: *Journal of the Operational Research Society* (2008), Nr. 59, S. 1150--1172

[Haar 2011] HAAR, Benjamin: *Analyse und Prognose von Trainingswirkungen: Multivariante Zeitreihenanalyse mit Künstlichen Neuronalen Netzen*. Stuttgart, Universität Stuttgart, Dissertation, 2011

[Hansmann 1979] HANSMANN, K. W.: Heuristische Prognoseverfahren. In: *WISU - Das Wirtschaftsstudium* (1979), Nr. Vol. 8, S. 229--233

[Hansmann 1983] HANSMANN, Karl-Werner: *Kurzlehrbuch Prognoseverfahren: Mit Aufgaben und Lösungen*. Wiesbaden : Gabler, 1983. -- ISBN 3-409-13444-1

[Heaton 2008] HEATON, Jeff: *Introduction to neural networks with Java*. 2. St. Louis : Heaton Research, 2008. -- ISBN 1604390085

[Heaton 2011] HEATON, Jeff: *Programming neural networks with Encog3 in Java*. 2. Heaton Research, Inc., 2011. -- ISBN 978-1-60439-021-6

[Hornik u. a. 1989] HORNIK, Kurt ; STINCHCOMBE, Maxwell ; WHITE, Halbert: Multilayer feedforward networks are universal approximators. In: *Neural Networks* 2 (1989), Nr. 5, S. 359--366

[Horváth 2003] HORVÁTH, Csilla: *Dynamic analysis of marketing systems*. Groningen, University of Groningen, Dissertation, 2003. -- URL http://irs.ub.rug.nl/ppn/252140400

[Hyndman und Athanasopoulos 2012] HYNDMAN, Rob J. ; ATHANASOPOULOS, George: *Forecasting: principles and practice*. 2012. -- URL http://otexts.com/fpp/. -- Zugriffsdatum: 24.03.2013

[Igel und Hüsken 2003] IGEL, Christian ; HÜSKEN, Michael: Empirical evaluation of the improved Rprop learning algorithms. In: *Neurocomputing* 50 (2003), S. 105--123

[Jacobsen 2013] JACOBSEN, Nils ; WISSENMEDIA IN DER INMEDIAONE GMBH (Hrsg.): *Ein Überblick über die Marktsegmente | wissen.de*. 2013. -- URL http://www.wissen.de/ein-ueberblick-ueber-die-marktsegmente. -- Zugriffsdatum: 12.03.2013

[Jain 2005] JAIN, C. L.: Benchmarking Forecasting Practices in Corporate America. In: *Journal of Business Forecasting* Winter 2005/2006 (2005), Nr. 24/4

[Janetzke und Lewandowski 2012] JANETZKE, Philipp ; LEWANDOWSKI, Achim: Der Beitrag der Künstlichen Intelligenz zur betrieblichen Prognose. In: MERTENS, Peter (Hrsg.) ; RÄSSLER,

Susanne (Hrsg.): *Prognoserechnung*. Heidelberg : Physica-Verlag HD, 2012, S. 341--382. -- ISBN 978-3-7908-2797-2

[Kaastra und Boyd 1996] KAASTRA, Iebeling ; BOYD, Milton: Designing a neural network for forecasting financial and economic time series. In: *Neurocomputing* 10 (1996), Nr. 3, S. 215--236

[Kirchgeorg und Piekenbrock 2013] KIRCHGEORG, Manfred Prof. D. ; PIEKENBROCK, Dirk Prof. D.: Bedarf (Version: 6). In: GABLER VERLAG (Hrsg.): *Gabler Wirtschaftslexikon*. Wiesbaden : Springer-Verlag, 2013 (Gabler Wirtschaftslexikon). -- URL http://wirtschaftslexikon.gabler.de/Archiv/194/bedarf-v6.html

[Klimasauskas 1996] KLIMASAUSKAS, Casimir C.: Applying Neural Networks. In: TRIPPI, Robert R. (Hrsg.) ; TURBAN, Efraim (Hrsg.): *Neural networks in finance and investing*. Chicago : Irwin Professional Pub., 1996, S. 45--69. -- ISBN 1557389195

[Kohonen 1989] KOHONEN, Teuvo: *Self-organization and associative memory*. 3. Berlin and New York : Springer-Verlag, 1989. -- ISBN 0-387-51387-6

[Koskela u. a. 1996] KOSKELA, Timo ; LEHTOKANGAS, Mikko ; SAARINEN, Jukka ; KASKI, Kimmo: Time series prediction with multilayer perceptron, FIR and Elman neural networks. In: *WCNN'96, San Diego, California, U.S.A*. Mahwah and N.J : L. Erlbaum, 1996, S. 491--496. -- ISBN 0-8058-2608-4

[Kruse u. a. 2012] KRUSE, Rudolf (Hrsg.) ; BORGELT, Christian (Hrsg.) ; KLAWONN, Frank (Hrsg.) ; MÖWES, Christian (Hrsg.) ; RUSS, Georg (Hrsg.) ; STEINBRECHER, Matthias (Hrsg.): *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Wiesbaden : Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2012. -- ISBN 978-3834812759

[Levine 2000] LEVINE, Daniel S.: *Introduction to neural and cognitive modeling*. 2. Mahwah and N.J : Lawrence Erlbaum Associates Publishers, 2000. -- ISBN 978-0805820058

[Lippe 2006] LIPPE, Wolfram-Manfred Prof. D.: *Soft-Computing: Mit Neuronalen Netzen, Fuzzy-Logic und evolutionären Algorithmen ; mit 27 Tabellen*. Berlin and Heidelberg and New York : Springer, 2006. -- ISBN 978-3-540-20972-0

[Makridakis und Hibon 2000] MAKRIDAKIS, Spyros G. ; HIBON, Michèle: The M3-Competition: results, conclusions and implications. In: *International Journal of Forecasting* (2000), Nr. 16, S. 415--476

[Makridakis u. a. 1998] MAKRIDAKIS, Spyros G. ; WHEELWRIGHT, Steven C. ; HYNDMAN, Rob J.: *Forecasting: Methods and applications*. 3. New York : John Wiley & Sons, 1998. -- ISBN 978-0471532330

[Masters 1993] MASTERS, Timothy: *Practical neural network recipes in C++*. Boston : Academic Press, 1993. -- ISBN 0124790402

[Medesker u. a. 1996] MEDESKER, Larry ; TURBAN, Efraim ; TRIPPI, Robert R.: Neural Network Fundamentals for Financial Analysts. In: TRIPPI, Robert R. (Hrsg.) ; TURBAN, Efraim (Hrsg.): *Neural networks in finance and investing*. Chicago : Irwin Professional Pub., 1996, S. 3--24. -- ISBN 1557389195

[Mertens und Rässler 2012] MERTENS, Peter ; RÄSSLER, Susanne: Prognoserechnung - Einführung und Überblick. In: MERTENS, Peter (Hrsg.) ; RÄSSLER, Susanne (Hrsg.): *Prognoserechnung*. Heidelberg : Physica-Verlag HD, 2012, S. 3--10. -- ISBN 978-3-7908-2797-2

[Moustafa 2011] MOUSTAFA, Akram A.: Performance Evaluation of Artificial Neural: Networks for Spatial Data Analysis. In: *Contemporary Engineering Sciences* Bd. 4. 2011, S. 149--163

[Ng 2012a] NG, Andrew: Advice for applying machine learning: Deciding what to Deciding what to try next. In: COURSERA.ORG (Hrsg.): *Coursera-Kurs: Machine Learning* Bd. 10. 2012

[Ng 2012b] NG, Andrew ; COURSERA.ORG (Hrsg.): *Coursera-Kurs: Machine Learning*. 2012. -- URL https://class.coursera.org/ml/lecture/preview

[Ng 2012c] NG, Andrew: Machine lerning system design: Prioritizing what to work on. In: COURSERA.ORG (Hrsg.): *Coursera-Kurs: Machine Learning* Bd. 11. 2012

[Palit und Popović 2005] PALIT, Ajoy K. ; POPOVIĆ, Dobrivoje: *Computational intelligence in time series forecasting: Theory and engineering applications*. London : Springer, 2005. -- ISBN 978-1852339487

[Pankratz 1991] PANKRATZ, Alan: *Forecasting with dynamic regression models*. New York : Wiley, 1991 (Wiley series in probability and mathematical statistics. Applied probability and statistics). -- ISBN 978-0471615286

[Puma-Villanueva u. a. 2006] PUMA-VILLANUEVA, W.J ; SANTOS, E.P dos ; ZUBEN, F.J v.: Data partition and variable selection for time series prediction using wrappers. In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, IEEE, 2006, S. 4740--4747. -- ISBN 0-7803-9490-9

[Rey 2013] REY, Günter D.: *Neuronale Netze: Eine Einführung*. 2013. -- URL http://www.neuronalesnetz.de/

[Riedmiller und Braun 1993] RIEDMILLER, M. ; BRAUN, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: *IEEE International Conference on Neural Networks*, IEEE, 1993, S. 586--591. -- ISBN 0-7803-0999-5

[Rojas 1993] ROJAS, Raúl: *Theorie der neuronalen Netze: Eine systematische Einführung*. Berlin and New York : Springer-Verlag, 1993. -- ISBN 978-3540563532

[Rosenblatt 1958] ROSENBLATT, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review* 65 (1958), Nr. 6, S. 386--408

[Rummelhart u. a. 1986] RUMMELHART, David E. ; HINTON, Geoffrey E. ; WILLIAMS, Ronald J.:

Learning Internal Representations by Error Propagation. In: *Nature* 323 (1986), Nr. 6088, S. 533--536

[Sarle 2002] SARLE, Warren S.: *Neural Network FAQ*. 2002. -- URL ftp://ftp.sas.com/pub/neural/FAQ.html

[Singh 2008] SINGH, Raj k.: *Forecasting Demand vs Sales*. 2008. -- URL http://oracledemantra.blogspot.de/2008/07/forecasting-demand-vs-sales.html

[Staub 2010] STAUB, Bastian: *Entscheidungsorientierte Marktsegmentbewertung mit dem Realoptionsansatz*. Frankfurt and M. [u.a.] : Lang, 2010. -- ISBN 978-3631602751

[Stepnicka u. a. 2013] STEPNICKA, Martin ; CORTEZ, Paulo ; PERALTA DONATE, Juan ; STEPNICKOVA, Lenka: Forecasting seasonal time series with computational intelligence: On recent methods and the potential of their combinations. In: *Expert Systems with Applications* 40 (2013), Nr. 6, S. 1981--1992

[taheretaheri 2010] TAHERETAHERI: *Benchmarking and Comparing Encog, Neuroph and JOONE Neural Networks*. 2010. -- URL http://www.codeproject.com/Articles/85487/Benchmarking-and-Comparing-Encog-Neuroph-and-JOONE. -- Zugriffsdatum: 25.10.2012

[Thiesing 1996] THIESING, Frank M.: Vorhersagen mit Neuronalen Netzen. In: *UNIX/MAIL* 96 (1996), Nr. 5. -- URL http://www.informatik.uni-osnabrueck.de/um/96/96.5/thiesing/thiesing.html

[Trippi und Turban 1996] TRIPPI, Robert R. (Hrsg.) ; TURBAN, Efraim (Hrsg.): *Neural networks in finance and investing: Using artificial intelligence to improve real-world performance*. Rev. ed. Chicago : Irwin Professional Pub., 1996. -- ISBN 1557389195

[Tzafestas u. a. 1996] TZAFESTAS, S.G ; DALIANIS, P.J ; ANTHOPOULOS, G.: On the overtraining phenomenon of backpropagation neural networks. In: *Mathematics and Computers in Simulation* 40 (1996), Nr. 5-6, S. 507--521

[Vemuri und Rogers 1994] VEMURI, V. R. ; ROGERS, Robert D.: *Artificial neural networks: Forecasting time series*. Los Alamitos and Calif : IEEE Computer Society Press, 1994. -- ISBN 978-0818651205

[Vonko 2009] VONKO, Dima ; INVESTOPEDIA ULC (Hrsg.): *Neural Networks: Forecasting Profits*. 2009. -- URL http://www.investopedia.com/articles/trading/06/neuralnetworks.asp

[Zell 1997] ZELL, Andreas: *Simulation neuronaler Netze*. 2. München [u.a.] : Oldenbourg, 1997. -- ISBN 978-3486243505

[Zhang u. a. 1998] ZHANG, Guoqiang ; PATUWO, Eddy B. ; HU, Michael Y.: Forecasting with artificial neural networks: The state of the art. In: *International Journal of Forecasting* 14 (1998), Nr. 1, S. 35--62

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____