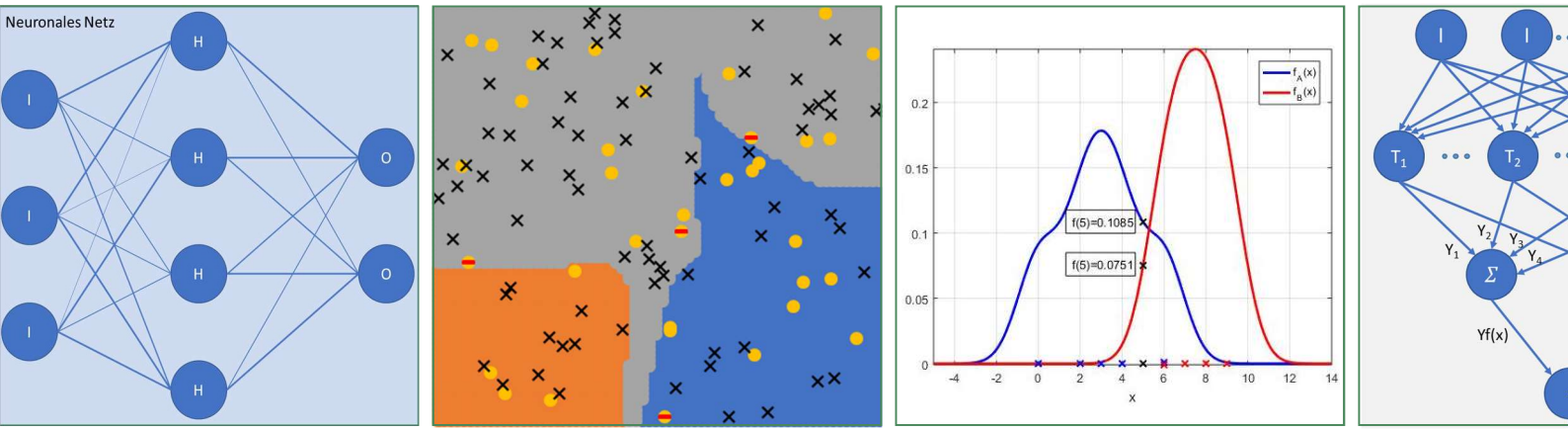
INSTITUT FÜR BAUBETRIEB UND BAUWIRTSCHAFT

MASTERARBEIT



EINSATZMÖGLICHKEITEN VON NEURONALEN NETZEN IM BAUBETRIEB UND IN DER BAUWIRTSCHAFT

Ralph Jakob Stöckl, BSc

Vorgelegt am

Institut für Baubetrieb und Bauwirtschaft

Betreuer

Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Hofstadler

Mitbetreuender Assistent

Dipl.-Ing. Dr.techn. Markus Klaus Kummer

Graz am 16. August 2018

Ralph Jakob Stöckl, BSc

Einsatzmöglichkeiten von neuronalen Netzen im Baubetrieb und in der Bauwirtschaft

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Wirtschaftsingenieurwesen – Bauwesen

eingereicht an der

Technischen Universität Graz

Betreuer

Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Hofstadler

Institut für Baubetrieb und Bauwirtschaft

Dipl.-Ing. Dr.techn. Markus Klaus Kummer

Graz, August 2018

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, ...

date (signature)

Anmerkung

In der vorliegenden Masterarbeit wird auf eine Aufzählung beider Geschlechter oder die Verbindung beider Geschlechter in einem Wort zugunsten einer leichteren Lesbarkeit des Textes verzichtet. Es soll an dieser Stelle jedoch ausdrücklich festgehalten werden, dass allgemeine Personenbezeichnungen für beide Geschlechter gleichermaßen zu verstehen sind.

Danksagung

An dieser Stelle möchte ich allen Personen danken, die mir während meiner Masterarbeit mit Rat und Tat zur Seite standen.

Besonderer Dank gebührt meiner Familie, die mich die gesamte Ausbildungszeit hindurch unterstützte. Ohne dem Wohlwollen und der Beihilfe meiner Eltern, wäre es mir nicht möglich gewesen, so rasch und intensiv zu studieren. Vielen Dank! Weiters möchte ich mich bei meinem Bruder für diverse Ratschläge bei kniffligen Fragestellungen und für das Korrekturlesen bedanken.

Für die engagierte Betreuung von universitärer Seite bedanke ich mich bei Herrn Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Hofstadler und Herrn Dipl.-Ing. Dr.techn. Markus Klaus Kummer.

Graz, August 2018 Ralph Jakob Stöckl

Kurzfassung

Die vorliegende Masterarbeit beleuchtet die theoretischen Grundlagen von ausgewählten neuronalen Netzen, deren Anwendungsmöglichkeiten im Bauwesen und die Software NeuralTools 7.5.

Durch die steigende Rechenleistung der Computer und die in den letzten Jahren weit vorangeschrittene Forschung im Bereich künstlicher Intelligenz, können immer größere Datenmengen in kürzer werdenden Trainingsdauern verarbeitet werden. Dies führt zu einer ständigen Erweiterung der Anwendungsgebiete. Mögliche Bereiche sind beispielsweise das Bankenwesen, die Automationstechnik oder die medizinische Diagnostik. Im Bauwesen wird dieses neue Hilfsmittel, welches seine Stärken in der Kategorisierung, Klassifizierung und den numerischen Prognose hat, nur sehr selten eingesetzt. Damit die versteckten Potenziale nicht unentdeckt bleiben, werden in dieser Arbeit neuronale Netze und deren Anwendungsmöglichkeiten im Baubetrieb und der Bauwirtschaft vorgestellt. Weiters wird auf NeuralTools 7.5, ein Excel-Aufsatz der Palisade Corporation, eingegangen. Dieses ermöglicht neuronale Netze zu trainieren, zu testen und Prognosen zu erstellen.

Nach der Erläuterung der theoretischen Grundlagen sowie der potenziellen Anwendungsmöglichkeiten, wird die Funktionsweise dieser Software vorgestellt. Abschließend folgt ein Anwendungsbeispiel zur Darstellung der Funktionsweise.

Abstract

This master thesis deals with the theoretical background of selected neural networks, their applications in construction engineering and the software NeuralTools 7.5.

Due to the increasing computing power and the great research achievements in the field of artificial intelligence in the recent years, more and more data can be analysed in the Training. Moreover the durations decreases. This leads to more and more fields of application, such as in the banking industry, in automation technology or in the medical sector. Only in construction engineering this new tool, which has its strengths in the categorization, classification and numerical prediction, is used very rarely. In order to show the high potential of artificial intelligence, this work presents neural networks and their application possibilities in construction engineering. It also explains NeuralTools 7.5, an Excel-Tool made by the Palisade Corporation. This program helps to train, test and create predictions with neural networks.

After explaining the theoretical basics and talking about possible fields of application, the functionality of this software is presented. Finally, an application example follows.

Inhaltsverzeichnis

1 Einleitung

In folgendem Kapitel werden die Situationsanalyse, die Zielsetzung, die Vorgehensweise und die Gliederung dieser Arbeite vorgestellt.

1.1 Situationsanalyse

Neuronale Netze erlangten in den letzten Jahren, aufgrund steigender Rechenleistungen von Computern, immer mehr an Bedeutung. Heutzutage sind sie aus dem Alltag nicht mehr wegzudenken. Bei der Entscheidung einer Kreditvergabe wirken diese Netze mit. Sie greifen auf riesige Datenmengen zurück und liefern daraus gute Prognosen über die Kreditwürdigkeit. Ein weiteres wichtiges Anwendungsgebiet stellt die Bilderkennung dar. Sowohl für das Arbeiten mit Robotern, als auch bei selbst fahrenden Autos ist es wesentlich aus Bildern Inhalte und Formen zu erkennen um daraus weiteres Handeln abzuleiten. Neben diesen Einsatzmöglichkeiten existieren noch unzählige weitere, wie beispielsweise Diagnosen im medizinischen Bereich oder die Erforschung des menschlichen Gehirns. Um diese Aufzählung zu verkürzen, kann grundsätzlich gesagt werden, dass sich neuronale Netze hervorragend zur Lösung von Aufgabenstellungen aus dem Bereich numerische Prognosen, Klassifizierungen und Mustererkennungen eignen. Deren Einsatz spart Zeit, wirkt unterstützend bei Entscheidungsfindungen, liefert dadurch wirtschaftliche Vorteile und ermöglicht Schritte in Richtung Automatisierung.

Trotz vieler Einsatzmöglichkeiten und einem immer stärker werdenden Preiskampf haben bis jetzt neuronale Netze in der Bauwirtschaft keine allzu große Verwendung gefunden. Jedoch ist es an der Zeit, die klassischen Methoden zu hinterfragen und neue zu implementieren um weiterhin durch Erkennen versteckter Potenziale die langfristige Bestandssicherung der Unternehmen zu gewährleisten. Es müssen die Produktionsfaktoren, bestmöglich aufeinander abgestimmt und die Entscheidungen rasch getroffen werden um am Markt Aufträge ohne Lohn- und Sozialdumping zu lukrieren. Neben Chancen-Risiko-Analysen oder Optimierungsmodellen eignen sich neuronale Netze hervorragend, um einen Wettbewerbsvorteil zu erlangen.

1.2 Zielsetzung

Die Aufgabenstellung dieser Masterarbeit wird in Muss-, Soll-, Kann- und Nicht-Ziele eingeteilt.

Ersteres beinhalten das Anwenden von NeuralTools 7.5 auf eine konkrete Aufgabenstellung sowie das Erklären dieser Software. Damit dieses Programm verstanden und richtig angewendet werden kann, ist es ebenfalls notwendig, die theoretischen Grundlagen der neuronalen Netze zu erläutern.

Die Soll-Ziele umfassen das Erarbeiten möglicher Anwendungsbereiche im Bauwesen, von den im theoretischen Teil vorgestellten Netztypen. Hierbei wird allerdings nicht auf die genauen Inputparameter eingegangen. Diese gilt es in einer separaten Arbeit durch eine Auswirkungsanalyse zu bestimmen. Die Auflistung der verschiedenen Einsatzmöglichkeiten soll das Potenzial von neuronalen Netzen verdeutlichen und als Anregung dienen, weitere noch unbekannte Verwendungszwecke im Bauwesen zu entdecken.

Das Vergleichen von unterschiedlichen Netztypen gehört zum Kann-Ziel. Abhängig von der Aufgabenstellung und vom vorhandenen Datensatz eignen sich manche Netztypen besser als andere. Die beste Netzkonfiguration ist jedoch im Vorhinein nicht sofort ersichtlich.

Zu den Nicht-Zielen gehört das Programmieren eines neuronalen Netzes, das Erarbeiten eines neuen Netztyps für Aufgabenstellungen aus dem Bauwesen und das Bearbeiten von Fragestellungen aus dem Bereich Warten Verwalten und Zurverfügungstellen von Datenmaterial, welche die Grundlage für das Training eines Netzes darstellen.

Die nachfolgende Abbildung 1-1 zeigt die Prioritäten dieser Masterarbeit in absteigender Reihenfolge, wobei das unterste Feld das Nicht-Ziel darstellt.

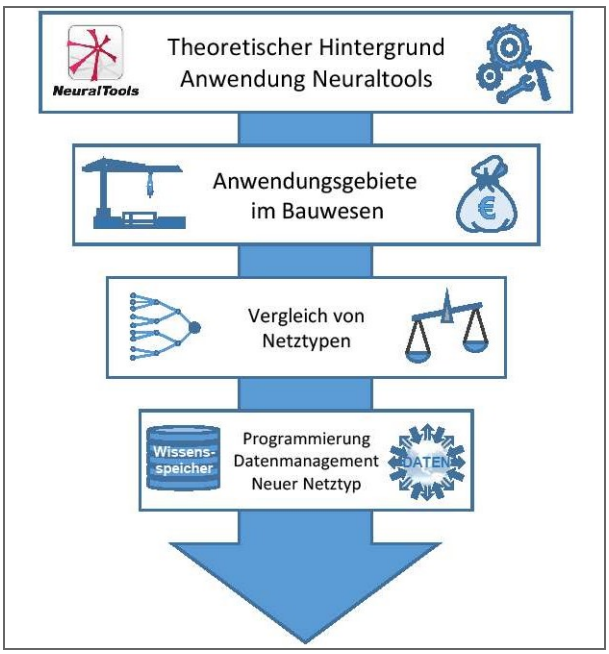


Abb. 1-1 Muss-, Soll-, Kann- und Nicht-Ziele

1.3 Vorgehensweise

Bei der vorliegenden Arbeit wird mit einem Brainstorming bezüglich künstlicher Intelligenz und neuronalen Netzen begonnen. Anschließend werden die Ergebnisse durch eine Mind-Map abgebildet. Nach dieser Orientierungsphase wird die Mind-Map sukzessive mit dem Wissen aus dem Fachliteraturstudium erweitert. Nachdem die einzelnen Äste ausgearbeitet sind, gilt es die Ziele der Masterarbeit zu definieren. Schlussendlich wird die Kapitelgliederung durchgeführt und mit dem Verfassen dieses Werkes begonnen.

Um einen reibungslosen Ablauf beim Erstellen dieser Masterarbeit zu ermöglichen, werden Elemente des Systems Engineering (SE) angewandt. Die Systems Engineering-Philosophie bildet den geistigen Überbau des SE-Denkmodells zur Lösung komplexer Probleme und besteht aus dem Systemdenken und Vorgehensmodell.[1]

Dem Systemdenken liegt der Systemansatz zu Grunde. Dieser besagt, dass mit Hilfe von Systemmodellen komplexe Zusammenhänge als Abstraktion der Wirklichkeit abgebildet werden können. Ein System besteht aus Elementen, die untereinander in Beziehung stehen und wird durch die Systemgrenze von der Umwelt getrennt. Neben einem Hauptsystem können auch mehrere Untersysteme existieren. Ein Bagger könnte beispielsweise als System mit den Elementen Schaufel, Ausleger, Motor, usw. beschrieben werden. Die nachfolgende Abbildung stellt eine schematische Darstellung eines System dar. [2]

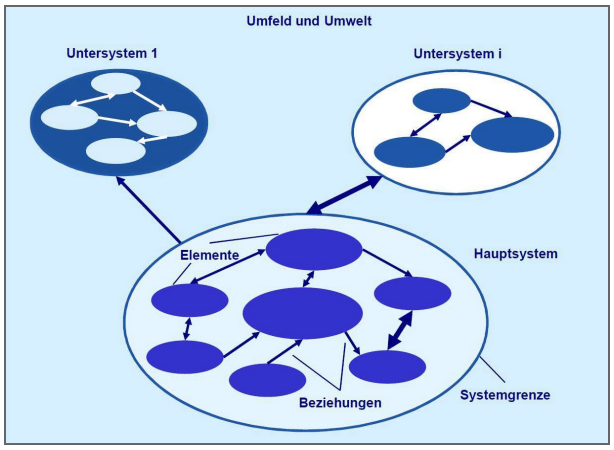


Abb. 1-2 Schematische Darstellung eines Systems[3]

Vorallem bei der Analyse, Strukturierung und Abgrenzung des Problemfelds, aber auch bei der Lösungssuche wird Systems Engineering erfolgreich angewendet.[4]

1 Vgl.: Hofstadler, C.; Schütz, M.: Anwendung des Systems Engineering auf die Arbeitsvorbereitung von Bauprojekten. In: Bautechnik, 11/2012. Seite 2

2 Vgl.: Hofstadler, C.; Schütz, M.: Anwendung des Systems Engineering auf die Arbeitsvorbereitung von Bauprojekten. In: Bautechnik, 11/2012. Seite 2

3 Hofstadler, C.: Baubetrieb Forschungsseminar: Systems Engineering Einführung, Seite 16

4 Vgl.: Hofstadler, C.; Schütz, M.: Anwendung des Systems Engineering auf die Arbeitsvorbereitung von Bauprojekten. In: Bautechnik, 11/2012. Seite 2

Das Vorgehensmodell beruft sich auf folgende vier Grundgedanken:[5]

- Vorgehensprinzip „Vom Groben zum Detail“: Das sukzessive Einengen der Lösungsmöglichkeiten verringert die Gefahr, wichtige Elemente außer Acht zu lasen.
- Prinzip der Variantenbildung: Das erstbeste Modell sollte nicht sogleich zur Lösungsfindung herangezogen werden.
- Prinzip der Phasengliederung als Makro-Logik: Durch die Gliederung in Teiletappen wird ein Überblick über die ständig fortschreitende Konkretisierung gegeben.
- Problemlösungszyklus als Mikro-Logik: Dieser hilft bei einer rascheren Lösungsfindung.

1.4 Gliederung

Die vorliegende Masterarbeit weist fünf Kapitel auf, welche in Abbildung 1-3 gezeigt werden. Nach der Einleitung folgt das zweite Kapitel, welches die theoretischen Grundlagen sowie verschiedene Anwendungsmöglichkeiten im Bauwesen behandelt. Es wird der mathematische Hintergrund von Feed-forward-, Wahrscheinlichkeits-, und verallgemeinerten Regressions-Netzen erläutert. Dieses Verständnis ist wichtig, um anschließend in Kapitel drei verstehen zu können, wie das Programm NeuralTools aufgebaut ist und funktioniert. In Kapitel 4 wird ein Anwendungsbeispiel mit Hilfe der vorgestellten Software analysiert. Schlussendlich folgt eine Zusammenfassung samt Ausblick.

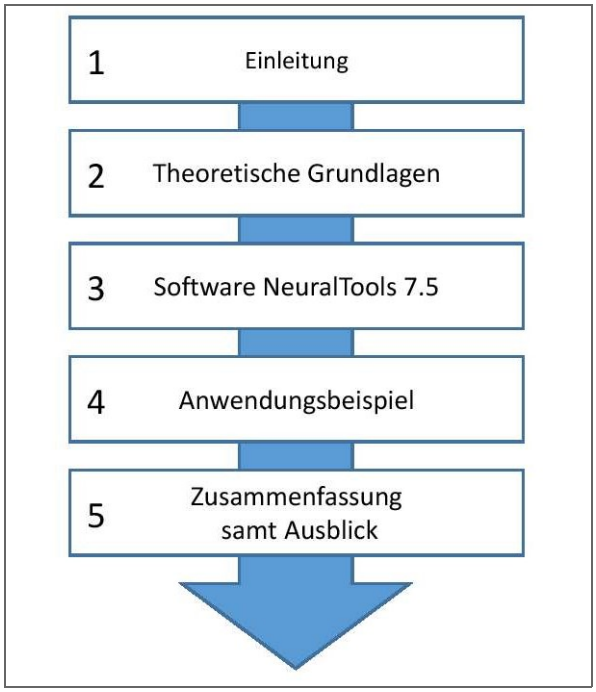


Abb. 1-3 Aufbau Masterarbeit

[5] Vgl.: Hofstadler, C.; Schütz, M.: Anwendung des Systems Engineering auf die Arbeitsvorbereitung von Bauprojekten. In: Bautechnik, 11/2012. Seite 2

2 Theoretische Grundlagen

In diesem Kapitel wird auf die Grundlagen von mehrschichtigen Feedforward-Netzen, Wahrscheinlichkeitsnetzen und Regressionsnetzen eingegangen. Diese drei Netzwerktypen stehen bei der Verwendung von NeuralTools zur Verfügung und werden näher erklärt, damit diese abhängig von deren Anwendungsmöglichkeiten richtig eingesetzt werden können.

2.1 Neuronale Netze

Ein Tag im Zeitalter der digitalen Revolution ohne Künstlicher Intelligenz ist kaum vorstellbar. Autos, die selbstständig fahren können, Bilderkennung, lernfähige Roboter und Spracherkennung erleichtern heutzutage den menschlichen Alltag.

Die künstliche Intelligenz stellt ein Teilgebiet der Informatik dar und wird von Rich/Knight/Nair auf folgende Art definiert:

„Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better“[1]

Mit anderen Worten, Künstliche Intelligenz beschreibt Forschungen, die das Ziel haben, einem Computer beizubringen, Aufgaben zu lösen, in denen der Mensch im Moment besser ist. Das heißt Maschinen müssen die Fähigkeit besitzen zu lernen.

Die folgende Abbildung 2-1 verdeutlicht die vier Teilgebiete der Informatik sowie weitere Einsatzbereiche der angewandten Informatik.

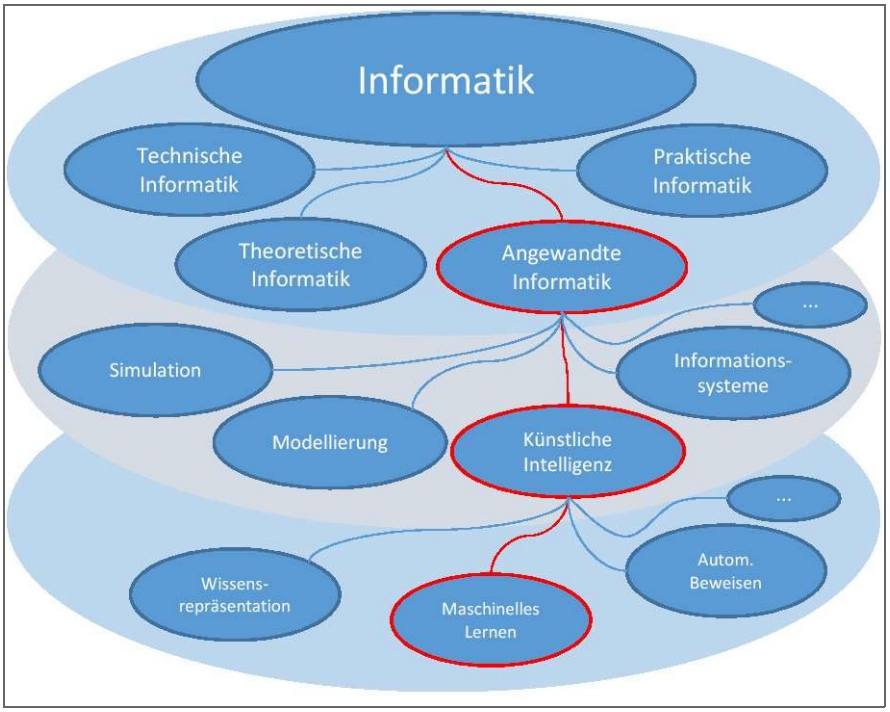


Abb. 2-1 Teilgebiete der Informatik[2]

[1] Rich, E.; Knight, K.; Nair S.B.:Artificial Intelligence, Seite 3

[2] Angewandte Informatik: Einsatz der Informatik in anderen Wissenschaften oder Gesellschaftsbereichen
Praktische Informatik: beschäftigt sich mit der Nutzung in Hinblick auf Betriebssysteme und Programme

Diese Begabung, bestehendes Wissen auf neue Aufgaben anwenden zu können, wird durch maschinelle Lernverfahren, welche ein Teilgebiet der Künstlichen Intelligenz darstellen, trainiert. Das Ziel beim maschinellen Lernen besteht darin, aus den gesammelten klassifizierten Daten eine Funktion zu generieren, mit Hilfe deren anschließend neue Inputparameter ausgewertet werden können.[3]

Grundsätzlich lassen sich die diversen Lernverfahren in überwachtes sowie nicht überwachtes Lernen und in Lernen durch Verstärkung unterteilen. Bei ersterem steht ein Lehrer zur Verfügung, der das Lernen beaufsichtigt, in dem er beispielsweise die Ausgabe zu einem gewünschten Input vorgibt. Hierbei ist das Ziel, eine Funktion zu finden, die auch zwischen den gegebenen Datenpunkten die Problemstellung gut annähert. Weiters besteht die Möglichkeit, ab und zu ein positives oder negatives Feedback zu geben. Dies wird Lernen durch Verstärkung genannt. Ein junges Teilgebiet der Lernverfahren stellt das Semi-Supervised-Learning dar. Hierbei werden sowohl Trainingsdaten mit dazugehörigem Output als auch Trainingsdaten ohne bekanntem Output für das Lernen herangezogen.[4]

Das Lernen mit Lehrer ist heutzutage eine etablierte Ingenieursdisziplin, welche viele erfolgreiche Anwendungen vorweist. Sind Inputdaten mit dazugehörigem Output gegeben, stellt die Mathematik und Informatik eine große Anzahl an Funktionsapproximationsalgorithmen zur Verfügung. Diese versuchen aus dem gegebenen Datenmaterial eine Funktion zu erstellen, mit der anschließend neue Inputs ausgewertet werden können. Sehr bekannte Vertreter dieser Approximationen stellen neuronale Netze dar.[5]

Neuronale Netze versuchen die Funktionen des Gehirns zu simulieren, um dadurch Aufschluss über menschliches Verhalten zu erlangen oder um konkrete Anwendungsprobleme zu lösen. Um das Jahr 1900 wurde erkannt, dass die motorischen und intellektuellen Fähigkeiten zu lernen auf die einzelnen Nervenzellen und deren komplexe Verschaltungen zurückzuführen sind. Im Jahr 1943 wurde erstmals von MCCulloch und Pitts ein mathematisches Modell eines Neurons, welches einen kleinen Baustein des Gehirns darstellt, entwickelt. Diese Erkenntnis legte die Basis für weitere Forschungen im Bereich neuronaler Netze, welche aufgrund der Nachahmung des menschlichen Gehirns auch als Bionik-Zweig der künstlichen Intelligenz verstanden werden können.[6]

Etwa 100 Milliarden Neuronen befinden sich im menschlichen Gehirn und bestehen neben dem Zellkörper aus einem Axon und Dendriten. Eine Nervenzelle erhält über diverse Dentrite, welche Verbindungen zu anderen Neuronen ermöglichen, Spannungsimpulse und speichert diese. Wird ein gewisser Schwellenwert überschritten schickt das Neuron die gesammelte Spannung über das Axon an andere Nervenzellen weiter, in denen der gleiche Prozess abläuft.[7]

Die Struktur des Gehirns ändert sich ständig, wodurch das Zeichnen eines Schaltplanes für das bessere Verständnis ein Ding der Unmöglichkeit wird. Dies stellt jedoch kein großes Problem dar, weil die zentrale Rolle von den Synapsen, welche die Verbindungspunkte zwischen Neuronen darstellen,

3 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 177
4 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 239
5 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 239
6 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 247
7 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 248

gespielt wird. Diese Verbindungsstellen sind nicht perfekt leitend verbunden, sondern weisen einen kleinen Spalt auf, welcher mit einer chemischen Substanz, den Neurotransmittern gefüllt ist. Abhängig von verschiedenen Parametern, wie beispielsweise der Konzentration oder der chemischen Zusammensetzung von Neurotransmittern, weisen diese Spalte unterschiedliche Leitfähigkeiten auf. Durch das Ändern dieser Leitfähigkeit ist es uns möglich zu lernen. Mittlerweile ist bekannt, dass Synapsen gestärkt werden, sprich eine höhere Leitfähigkeit aufweisen, wenn viele Spannungsimpulse übertragen werden müssen. Werden wenig Reize weitergegeben wird diese Gewichtung in die andere Richtung durchgeführt und kann bis zum Absterben einer Verbindung führen.[8]

Bis heute ist noch immer nicht ganz geklärt, wie diese bereits erforschten Prinzipien das intelligente Verhalten ermöglichen. Aus diesem Grund versuchen Forscher der Neuroinformatik zu neuen Erkenntnissen durch Simulieren von mathematischen Modellen zu gelangen.[9] Diese künstlichen neuronalen Netze, welche die zuvor beschriebenen Gesetzmäßigkeiten beinhalten, werden allerdings auch schon aufgrund des rasanten Forschungsfortschrittes bei konkreten Anwendungsproblemen, wie beispielsweise der Bilderkennung oder der Kreditwürdigkeit, erfolgreich eingesetzt.

2.1.1 Mathematisches Modell

Die einzelnen Neuronen bilden die Grundelemente der neuronalen Netze. Aus mathematischer Sicht ist dieses Neuron mit einer Variablen, welche bestimmte Zahlenwerte annehmen kann, gleichzusetzen. Meistens weist es einen positiven reellen Wert auf. Abhängig von speziellen Netztypen können jedoch auch binäre (0 oder 1) oder bipolare (-1 oder +1) Werte angenommen werden.[10]

Neuronen, die auch Units, Einheiten oder Knoten genannt werden, haben die Aufgabe Informationen aus der Umwelt oder von anderen Neuronen aufzunehmen, zu modifizieren und an andere Units oder die Umwelt weiterzugeben. Anhand der zuvor erwähnten Bestimmungen lassen sich diese Einheiten in drei verschiedene Arten unterteilen:[11]

-Input-Units: empfangen Signale/Reize von der Außenwelt

-Hidden-Units: befinden sich zwischen Input-und Output-Units

-Output-Units: geben Signale an die Außenwelt weiter

Das zweite Grundelement eines neuronalen Netzes stellen die einzelnen Gewichtungen, die in einer Gewichtsmatrix $W = (w_{ij})$ gespeichert werden, dar. Anhand dieser kann in der Regel die topologische Struktur des Netzes abgelesen werden, weil die Gewichtsmatrix die Verknüpfungen unter den Neuronen und deren Stärke (Gewichtung) verkörpert.[12]Erst die Gewichtungen zwischen den einzelnen Neuronen ermöglichen die Adaptionsfähigkeit dieses dynamischen Systems. Sie sind vergleichbar mit der Leitfähigkeit einer Synapse, welche unter anderem von den Neurotransmittern

8 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 248ff
9 Vgl.: Ertel, W..: Grundkurs Künstliche Intelligenz, Seite 250
10 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 141
11 Vgl.: http://www.neuronalesnetz.de/. Datum des Zugriffs: 17.05.2018
12 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 141

abhängt. Somit ermöglichen Änderungen in der Gewichtsmatrix die Fähigkeit zu lernen.

Die nachfolgende Abbildung 2-2 zeigt den schematischen Aufbau eines neuronalen Netzes.

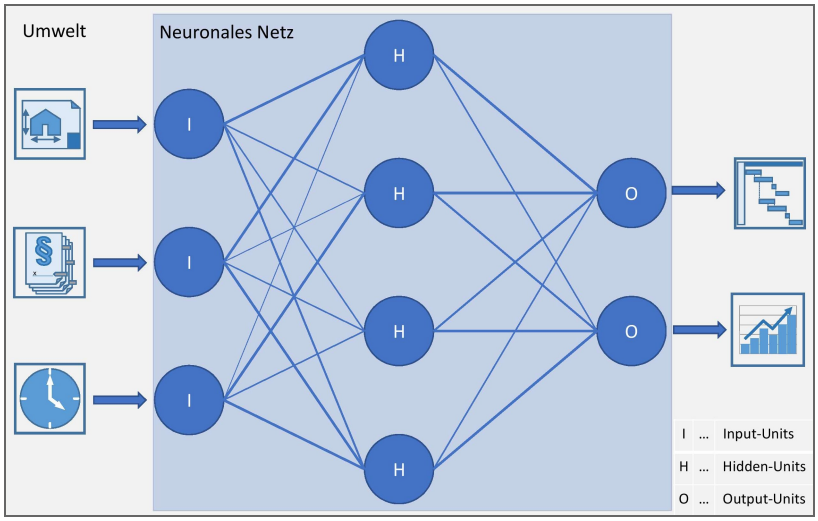


Abb. 2-2 Schematischer Aufbau eines neuronalen Netzes

Die runden Kreise repräsentieren die Neuronen. I steht für Input-Units, welche die vorgegebenen Reize aus der Umwelt aufnehmen. Dies könnten unter anderem Pläne, Bauverträge oder Zeitvorgaben sein. Die H-Neuronen stehen für Hidden-Units, welche das gesamte Modell verkörpern und in verschiedenen Schichten (Layer) angeordnet werden. In diesem Fall gibt es nur einen Layer für Hidden-Units. Die O-Einheiten stellen die Output-Units dar. Diese geben die Ergebnisse, beispielsweise Tätigkeitsanordnungen in Bauzeitplänen oder Prognosen, an die Umwelt aus. Die Verknüpfung zwischen den Neuronen wird durch Linien abgebildet. Verschiedene Strichstärken verdeutlichen die unterschiedlichen Gewichtungen.

Grundsätzlich kann jedes Neuron mit jedem anderen in Verbindung stehen. Jedoch führen oftmals derartige freie Strukturen wegen Rückkopplungen zu komplexen Dynamiken und konvergieren nur selten. Aus diesem Grund werden die einzelnen Units in einem neuronalen Netz, wie bereits in Abbildung 2-2 angedeutet, in verschiedenen Klassen zusammengefasst, um eine Schichtenstruktur zu erhalten.[13]

Weiters können neuronale Netze nach deren gerichteten Verbindungen eingeteilt werden. Verläuft der Informationsfluss nur von Eingabeschicht in Richtung Ausgabe, so wird dies Feedforward-Struktur genannt. Eine Verbindung ist rekurrent, wenn sie der zuvor erwähnten Vorzugsrichtung entgegengesetzt ist.[14] Die folgende Abbildung 2-3 zeigt einige Beispiele.

[13] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 143
[14] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 143f

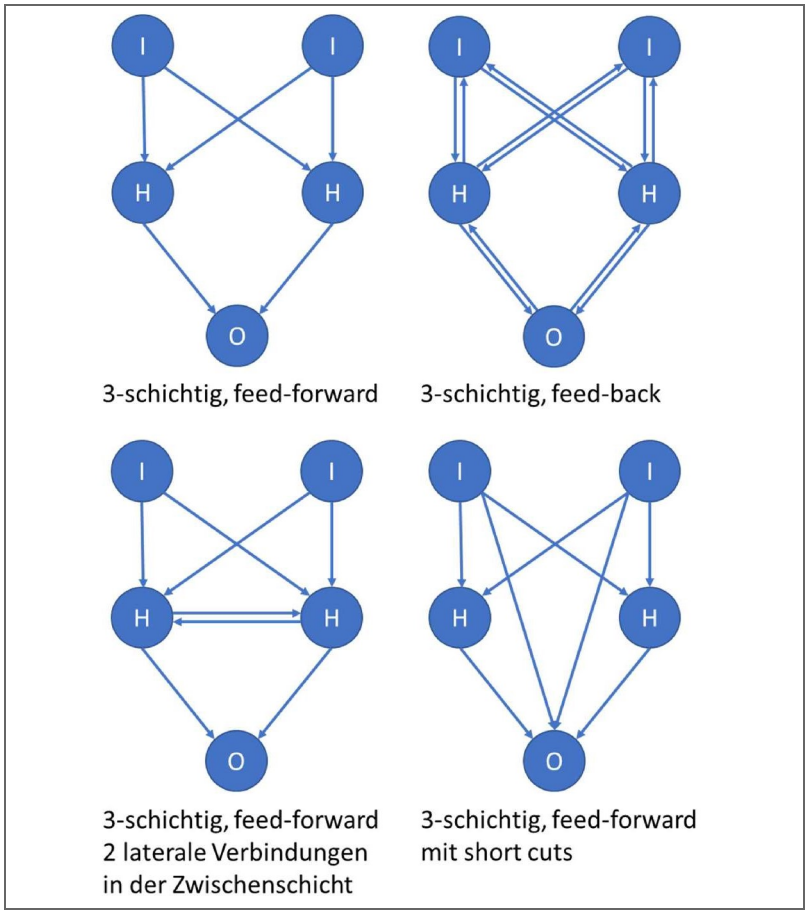


Abb. 2-3 Schematische Darstellung von Netzstrukturen[15]

Alleine bei der Betrachtung von vier Neuronen, können bereits 218 verschiedene Strukturen mit gerichteten Verbindungen nachgebaut werden. Bei sechs Neuronen steigt die Anzahl bereits auf 1,5 Millionen. Dieser explosionsartige Anstieg ist ein Grund dafür, weshalb es empfohlen wird sich auf wenige spezielle Strukturen zu beschränken, von denen man weiß, dass sie hervorragend übliche Probleme lösen oder simulieren können. Ein weiterer Grund für Strukturbeschränkungen ist, dass das Repertoire an Lernalgorithmen, welche theoretisch hinlänglich erfassbar sind um fundierte Aussagen über das Konvergenzverhalten zu machen, sehr bescheiden ist. Das dritte Argument, nicht jede beliebige Struktur zu nehmen, ist die Performanz der Programme. Dies beruht auf der Tatsache, dass Algorithmen, die allgemeine Matrizen bearbeiten, mit steigender Anzahl an Neuronen schnell langsam und uneffektiv werden.[16]

In der Praxis werden deshalb hauptsächlich Schichtstrukturen in Verbindung mit reinen Feedforward-Architekturen verwendet. Netze mit rekurrenten Verbindungen würden zu weitaus komplexeren Dynamiken führen, die sich wiederum negativ auf die Konvergenz bzw. Stabilität auswirken können. Die geeignetste Vorgehensweise, um eine Aufgabenstellung mit neuronalen Netzen zu lösen ist, aus der Literatur ein im selben Fachbereich eingesetztes Netz zu suchen und dieses anschließend anzugleichen. Somit ist sicherge-

[15] In Anlehnung an: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 144f
[16] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 144f

stellt, dass die Netzstruktur und die Lernalgorithmen zusammenpassen, was sich wiederum positiv auf Eignung und Effizienz des Netzes auswirkt.[17]

2.1.2 Funktionen

Damit ein Neuron den gegebenen Input in einen Output transformieren kann, sind in der Regel drei hintereinanderliegende Funktionen auszuführen. Diese drei werden Propagierungs-, Aktivierungs- und Outputfunktion genannt. Um kein zu kompliziertes System zu erhalten, werden meistens alle Neuronen, oder zumindest schichtgleiche Units mit den selben Funktionen versehen.[18] In den folgenden Absätzen werden die gebräuchlichsten Funktionen kurz erklärt, weil diese große Auswirkungen auf die Netzeigenschaften haben.

Propagierungsfunktionen

Das Aufsummieren der Inputs stellt die einfachste Form von Propagierungsfunktionen dar. Hierbei wird der Nettoinput (net_j) aus der Summe von den Outputs (o_i) der sendenden Neuronen, multipliziert mit den dazugehörigen Gewichtungen (w_{ij}) berechnet. Diese lineare Funktion wird in Gleichung (2-1) gezeigt. Wird zusätzlich ein Schwellenwert Θ_j berücksichtigt, wird der Nettoinput nach Gleichung (2-2) berechnet.[19]

$$net_j = \sum_{i=1}^{n} w_{ij} o_i \tag{2-1}$$

$$net_j = \begin{pmatrix} x \text{ für } \sum_{i=1}^{n} w_{ij} o_i > \Theta_j \\ 0 \text{ sonst} \end{pmatrix} \tag{2-2}$$

Aktivierungsfunktionen

Die Aktivierungsfunktion $F_j(net_j)$ beeinflusst den Wert, welchen das Neuron mit dem Nettoinput (net_j) an andere Units weitergibt. Hierbei ist zu beachten, dass die Wahl dieser Funktion die Eigenschaften des Netzes sehr stark beeinflusst, weil dadurch Nicht-Linearitäten eingebaut werden können. Das Ziel ist es die Aktivierungswerte a_j der einzelnen Neuronen innerhalb sinnvoller Intervalle zu halten, damit ein Netz konvergieren und schnell lernen kann. Zu große Werte, welche durch ständiges Aufsummieren entstehen, würden zu sinnlosen Rechenoperationen aufgrund infinitesimaler kleiner Veränderungen führen. Aus diesem Grund sollte die Auswahl nach dem zu simulierenden Problem gerichtet werden. Folgende Aktivierungsfunktionen sind sehr gebräuchlich:[20]

eine Schwellenwertfunktion

$$a_j = \begin{pmatrix} x \text{ für } net_j > \Theta_j \\ y \text{ für } net_j \leq \Theta_j \end{pmatrix} \tag{2-3}$$

In dem Neuron j stellen x und y mögliche Aktivierungszustände dar. Weiters kann der Schwellenwert Θ_j eigens für jedes Neuron selbst, für Klassen oder für das gesamte Netzt frei gewählt werden. Diese Funktionen, bei denen ein

[17] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 145ff
[18] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 139
[19] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 149
[20] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 150

gewisser Schwellenwert überwunden werden muss, werden allerdings auch häufig bei Outputfunktionen eingesetzt.[21]

Signumfunktion sgn(net_j)

Diese Funktion, auch Vorzeichenfunktion genannt, wird bei bi-polar kodierten Netzen eingesetzt, weil sie Aktivierungszustände a_j = +1 oder -1 liefert. Die folgende Abbildung und die kommende Gleichung (2-4) zeigen die Signumfunktion, welche wiederum häufig auch als Outputfunktion Verwendung findet.[22]

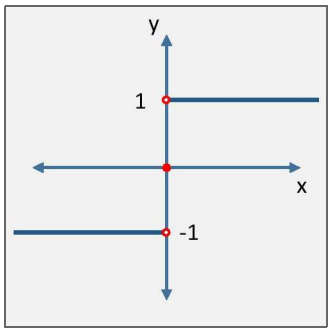


Abb. 2-4 Signumfunktion

$$a_j = \begin{pmatrix} 1 \text{ für } x > 0 \\ 0 \text{ für } x = 0 \\ -1 \text{ für } x < 0 \end{pmatrix} \tag{2-4}$$

lineare Aktivierungsfunktion

$$a_j = \sum_{i=1}^{n} w_{ij} o_i = net_j \tag{2-5}$$

Die lineare Aktivierungsfunktion wird in Gleichung (2-5) dargestellt. Auf Grund der Tatsache, dass diese beliebig wachsen kann, weil die Aktivierungswerte nicht begrenzt sind, werden häufig eine untere und obere Schranke, Θ_u und Θ_o, hinzugefügt. Die veränderte Funktion wird in Gleichung (2-6) beschrieben und zusätzlich in folgender Grafik 2-5 bildlich verdeutlicht.[23]

$$a_j = \begin{pmatrix} net_j \text{ für } \Theta_u < net_j < \Theta_o \\ x \text{ für } net_j \leq \Theta_u \\ y \text{ für } net_j \geq \Theta_o \end{pmatrix} \tag{2-6}$$

21 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 150
22 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 150
23 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 150f

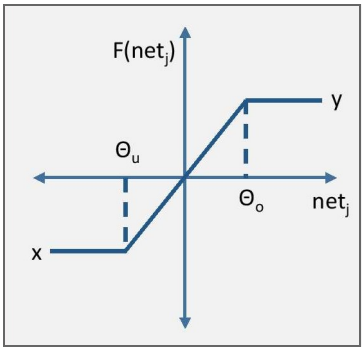


Abb. 2-5 Lineare Aktivierungsfunktion mit oberer und unterer Schranke

Sigmoide Funktion

Oftmals verwendete Aktivierungsfunktionen stellen sigmoide Funktionen, welche auch Schwanenhals- oder S-Funktionen genannt werden, dar. Die Namensgebung ist auf einen S-förmigen Verlauf zurückzuführen. Meistens wird mit sigmoiden Funktionen die logistische Funktion assoziiert. Diese wird in Gleichung (2-7) gezeigt und in Abbildung 2-6 schematisch dargestellt.[24]

$$a_j = F_j(net_j) = \frac{1}{1 + e^{-\frac{net_j}{\delta}}} \tag{2-7}$$

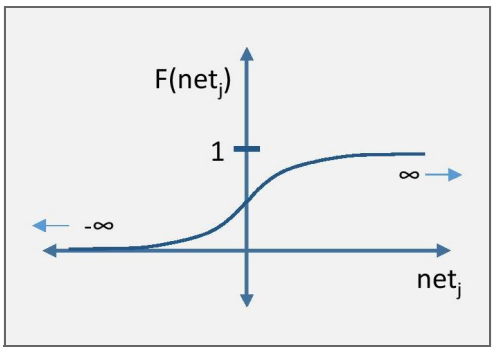


Abb. 2-6 Schematische Darstellung einer sigmoiden Funktion

Tangens-Hyperbolicus-Funktion

Eine weitere sigmoide Funktion, welche als Aktivierungsfunktion eingesetzt wird, ist der Tangens-Hyperbolicus. Die Aktivierungswerte werden nach Gleichung (2-8) bestimmt. Der Graph dieser Funktion wird in Abbildung 2-7 wiedergegeben.[25]

$$a_j = F_j(net_j) = \tanh(net_j) \tag{2-8}$$

[24] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 151
[25] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 151

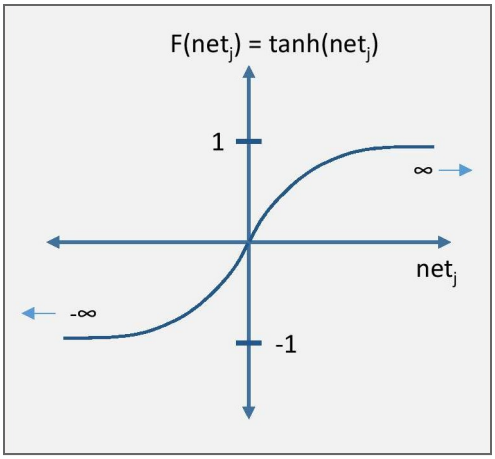


Abb. 2-7 Schematische Darstellung des Tangens-Hyperbolicus

Abhängig von der Aufgabenstellung haben sich in der Praxis verschiedene Funktionen und zugehörige Parameter bewährt. Aus diesem Grund ist es zu empfehlen beim Entwurf eines Netzes zunächst die Literatur nach ähnlichen Problemen zu durchsuchen und mit den dortig vorgeschlagenen Funktionen zu beginnen. Weiters empfiehlt es sich, möglichst einfache Funktionen einzubauen. Diese sollten jedoch mit der Kodierung des Netzes abgestimmt sein. Die Schwellenwertfunktion findet meistens bei binär kodierten und die Signumfunktion bei bi-polar kodierten Netzen Verwendung.[26]

Die Beliebtheit der logistischen Funktion, sowie des Tangens-Hyperbolicus ist darauf zurückzuführen, dass deren Ableitungen, welche bei den Lernregeln einen wichtigen Platz einnehmen, durch die selbige Funktion ausgedrückt werden kann. Dieser Vorteil wird in Gleichung (2-9) gezeigt.[27]

$$\begin{aligned} &\text{logistische Funktion:} && f'(x) = f(x)(1-f(x)) \\ &\text{Tangens-Hyperbolicus:} && f'(x) = 1-(f(x))^2 \end{aligned} \tag{2-9}$$

Outputfunktionen

Vorzugsweise werden die Identitätsfunktion, welche in Gleichung (2-10) abgebildet ist, und die Schwellenwertfunktion, die in Gleichung (2-3) dargestellt wird, als Outputfunktion o_j in Betracht gezogen. Ist eine bi-polare Ausgabe erwünscht, ist eine Signumfunktion zu verwenden.

$$o_j = a_j = F(net_j) \tag{2-10}$$

Von Schmidt/ Klüver/ Klüver werden diese drei Funktionen in jeweils einem Satz sehr treffend zusammengefasst:

[26] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 152f

[27] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 152

„Die Propagierungsfunktion fasst durch eine gewichtete Summe die Eingaben aus allen auf ein betreffendes Neuron j wirkenden anderen Neuronen zu der Netzeingabe net_j zusammen.

Die Aktivierungsfunktion erzeugt aus net_j und ggf. schon vorhandener Aktivierung einen neuen Aktivierungswert a_j des Neurons j.

Die Outputfunktion (Ausgabefunktion) erzeugt aus der Aktivierung a_j die Ausgabe o_j, die an andere Neuronen bzw. den Outputvektor weitergegeben wird.“[28]

2.1.3 Die Fähigkeit zu lernen

Mathematisch gesehen ermöglicht die Adaptionsfähigkeit dynamischer Systeme die Fähigkeit neuronaler Netze zu lernen. Hierbei wird durch eine steuernde Instanz systematisch oder auch stochastisch das Netz verändert. Um eine Problemstellung lösen zu können, muss die Instanz zunächst den Status des Netzes evaluieren. Im Zuge dessen können sowohl externe Größen, wie beispielsweise ein Lernziel oder ein zu erkennendes Muster, als auch interne Größen, welche zum Beispiel Zustandsfunktionen darstellen, analysiert werden. Im ersten Fall spricht man von überwachtem Lernen, bei dem die Annäherung des Netzoutputes an das Lernziel zurückgegeben wird, im zweiten Fall von nicht überwachtem Lernen, wobei hier auch vom selbstorganisierten Lernen gesprochen wird.[29]

Die Anpassung eines Netzes an die Lösung einer Aufgabenstellung, kurz lernen, kann auf folgende Arten erfolgen:[30]

1) neue Neuronen können hinzugefügt oder bestehende gelöscht werden

2) neue Verbindungen können hinzugefügt oder bestehende gelöscht werden

3) die Gewichtswerte (Verbindungsstärken) können verändert werden

4) die Transformationsfunktionen können modifiziert werden

5) kontinuierliche Inputs spezieller Neuronen können abgeändert werden

Unter den ersten drei Möglichkeiten wird die Veränderung der Topologie eines Netzes verstanden, welche über die Anpassung der Gewichtsmatrix realisiert wird. Die letzten zwei Modifikationen werden eher als Veränderung des Ablaufmodus interpretiert.[31]

Um den Stand eines Lernprozesses bewerten zu können, wird eine neue Funktion eingeführt. Diese Fehlerfunktion E beschreibt in überwacht lernenden neuronalen Netzen die Abweichung oder Distanz des Outputvektors Y und des Zielvektors T, der auch Target-Vektor genannt wird. Hierbei wird die Euklidische Distanz, die in Gleichung (2-11) gezeigt wird, am häufigsten für n-dimensionale Vektoren mit binären, ganzzahligen oder reellen Komponenten eingesetzt.[32]

[28] Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 153
[29] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 140
[30] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 140
[31] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 140
[32] Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 160

$$d = \sqrt{\sum_{i=1}^{n} (t_i - y_i)^2} \tag{2-11}$$

Oftmals wird jedoch das Quadrat derselben benutzt, wobei der Faktor 1/2 aus Gründen der Vereinfachung hinzugefügt wird. Dies ist möglich, weil es beim Berechnen des Minimums der Distanz (Extremwertaufgabe) grundsätzlich keine Rolle spielt, ob die Euklidische Distanz oder deren Quadrate, welche in Gleichung (2-12) abgebildet ist, eingesetzt werden.

$$d = \frac{1}{2} \sum_{i=1}^{n} (t_i - y_i)^2 \tag{2-12}$$

Der Lernprozess läuft solange, bis die Fehlerfunktion einen genügend kleinen Wert angenommen hat. Dieses Abbruchkriterium beinhaltet somit den Fehler des gesamten Outputvektors Y bezogen auf den gesamten Target-Vektor T. An dieser Stelle gilt es jedoch anzumerken, dass der Wert der Fehlerfunktion nicht mit den Fehlern der einzelnen Komponente y_i des Outputvektors bezüglich der dazugehörigen Komponente t_i des Zielvektors verwechselt werden darf. Diese Fehler der Form $\delta_i = t_i - y_i$ stellen das Maß der einzelnen Abweichungen dar und helfen bei der Steuerung der mit den Komponenten verknüpften Gewichtswerten w_{ki}.[33]

Der Fehler eines Netzes hängt naturgemäß von allen Parametern ab, die während des Lernens angepasst werden können. Aus diesem Grund wird die Fehlerfunktion, welche durch Distanzen oder in einer modifizierten Form Anwendung findet, durch die veränderbaren Gewichtswerte dargestellt. Diese mehrdimensionale Fehlerfunktion bildet somit eine Fläche im n+1-dimensionalen Raum und wird in der Grundform in Gleichung (2-13) beschrieben.[34]

$$E(W) = E(w_{ij}) \qquad i = 1, 2, 3, ..., n \tag{2-13}$$

In der Trainingsphase des überwachten Lernens wird mit Hilfe bereits bekannter Muster versucht das Minimum dieser Fehlerfläche zu finden. Bei einer Funktion mit einer Variablen muss die erste Ableitung zu Null gesetzt werden um einen Extremwert zu finden, bei einer mehrdimensionalen Funktion wird bei der Berechnung des Minimums der Gradient, welcher die Richtung des steilsten Anstieges repräsentiert, mit Null gleichgesetzt.[35]

Ein allgemeiner Algorithmus, welcher sehr häufig eingesetzt wird, um von einem beliebigen Punkt aus das Minimum der Fehlerfläche zu erreichen, ist das Gradienten-Abstiegs-Verfahren. Hierbei bewegt man sich von einem Standpunkt , welcher den derzeitigen Systemzustand verkörpert, in Richtung des steilsten Abstiegs, sprich in Richtung des größten negativen Gradienten, um den kleinsten Fehlerwert zu finden.[36]

33 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 160f
34 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 161
35 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 161f
36 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 162

Beim Lernen in mehrschichtigen neuronalen Netzen müssen die Änderungen der Gewichte zwischen Input- und Zwischenschicht sowie zwischen zwei Hidden-Layern von den Abweichungen, die sich aus den Aktivierungswerten der Output-Schicht und den Zielwerten ergeben, abhängen. Der am weitesten verbreitete Lernalgorithmus, der hierbei zum Einsatz kommt, wird Backpropagation genannt. Die Namensgebung dieser Lernregel ist darauf zurückzuführen, dass der Fehler, der in der Outputschicht entsteht, gegen die Rechenrichtung des Netzes rückwärts in die Verbindungsgewichte eingearbeitet wird. Anwendung findet diese Lernregel bei mehrschichtigen Feedforward-Netzen mit monotonen, differenzierbaren Aktivierungsfunktionen.

2.1.4 Anwendungsmöglichkeiten

Neuronale Netze stellen eine moderne Lösungsmöglichkeit für technische Probleme dar. Wenn klassische Modelle oder statistische Methoden an ihre Grenzen kommen, können neuronale Modelle immer noch passende Ergebnisse liefern. Jegliche Art von Problemstellung, bei der genügend Datenmaterial, beispielsweise aus einem Experiment oder analytischen Berechnungen, vorhanden ist, eignet sich für die Modellierung durch neuronale Netze.[37] Speziell bei Aufgabenstellungen, die aus dem Bereich der Mustererkennung, Klassifizierung oder Prognosen kommen, sind Modelle von überwacht lernenden neuronalen Netzen sehr gut einsetzbar.

Sowohl im konstruktiven Ingenieurbau als auch im Baubetrieb und der Bauwirtschaft werden neuronale Netze schon seit mehreren Jahren eingesetzt. Bereits im Jahre 1989 verwendeten Adeli und Jeh ein derartiges Model um Stahlträger zu modellieren. Unter anderem wurden solche Netze auch in der Bruchmechanik zur Parameteridentifikation, im Stahlbau für das Verhalten von Verbindungen, im Hochbau für Windlastenberechnungen oder im Betonbau zur Bewertung bestehender Betonbrücken eingesetzt. Weiters fanden neuronale Netze bereits Anwendung bei der Vorhersage von Knicklasten für Stützen, bei der Optimierung von kalt verformten Stahlträgern, bei der Schadensdetektion oder bei Sicherheitsfragen bezüglich der Einsturzgefahr nach Erdbeben. Hierbei muss jedoch angemerkt werden, dass gerade bei Fragen bezüglich der Sicherheit und Tragfähigkeit genaue Ergebnisse gebraucht werden. Neuronale Netze können allerdings nur Annäherungen liefern. [38]

Neben den zuvor exemplarisch aufgezählten Einsatzmöglichkeiten wird die Verwendung von neuronalen Netzen auch in Bereichen wie zum Beispiel der Finiten-Elemente-Methode, der Modellierung von Materialien, im Verkehrswesen, in der Geotechnik oder im Bauwirtschaftsingenieurswesen untersucht. [39]

In der Bauwirtschaft werden häufig neuronale Netze bei Fragestellungen aus dem Spezialgebiet der Terminplanung, der Kostenschätzung, der Produktivität, der Projektdauer oder der Aufwandswerte bzw. der Leistungswerte eingesetzt.[40]

[37] Vgl.: Lazarevska M. et al.: Application of artificial neural networks in civil engineering (2014), Seite 1353
[38] Adeli, H.: Neural Networks in Civil Engineering: 1989-200;(2001), Seite 126ff
[39] Adeli, H.: Neural Networks in Civil Engineering: 1989-200;(2001), Seite 126ff

Zu folgenden Aufgabenstellungen wurden beispielsweise bereits universitäre Überlegungen angestellt: nicht lineare Optimierung, Erstellung eines Bauzeitplans unter Minimierung der Konstruktionskosten, Vorhersage von Rechtsstreitigkeiten im Bauwesen, Schalungsauswahl für Wand und Stütze, Auswahl der horizontalen Schalung für Decken und Dächer, Messung der Effizienz im Baumanagement sowie die optimale Geräteauswahl für Erdbewegungen.[41]

Weitere Forschungsthemen, welche in Fachzeitschriften publiziert wurden, beschäftigen sich mit: Kostenschätzungen von Schulgebäuden und Wohnprojekten sowie von Stahlbeton-Tragstrukturen, Gemeinkosten, der Schätzung von Angebotspreisen aufgrund umweltrelevanter Faktoren, Leistungswerte von Planierraupen, Aufwandswerte für Schalen, Bewehren, Betonieren, Aufwandswerte für das Aufkleben von Fliesen an Wänden sowie das Aufstellen von Ziegelwänden, der Risikoabschätzung eines Verzugs des Auftragnehmers, Kostenabweichungen, Abweichungen der Kosten aufgrund politischem Einfluss, Sicherheit auf der Baustelle, Hebezeiten bei Kränen, Auswahl des besten Angebots sowie Brückenbauzeiten.[42]

Die zuvor aufgezählten Einsatzmöglichkeiten sollen die vielseitige Verwendung von neuronalen Netzen verdeutlichen. Sobald genügend Daten vorhanden sind, kann ein Problem modelliert und anschließend analysiert werden. Gerade bei Abschätzungen, welche nicht zu viel Zeit in Anspruch nehmen dürfen, eignen sie sich besonders gut. Dies gilt beispielsweise für Kostenschätzungen oder bei Annahmen für Aufwands- und Leistungswerte, die anschließend in der Kalkulation eingesetzt werden. Weiters könnten auch Aufwandswerterhöhungen mit Hilfe neuronaler Netze analysiert werden.

Einen noch nicht angesprochenen Verwendungszweck stellt die Bilderkennung dar. Es ist möglich auf Fotos oder in Aufnahmen bestimmte Formen zu erkennen. Dies kann beim automatischen Überwachen des Baustellenfortschrittes eingesetzt werden. Mit Hilfe eines trainierten Netzes ist es möglich Fotos oder Videos, welche vom Kran herab gemacht werden, auszuwerten. Die Fähigkeit zwischen einer Wand und deren dazugehörigen Schalung unterscheiden zu können ist beispielsweise ein sehr wichtiger Schritt, um eine völlig automatisierte Fortschrittskontrolle einführen zu können.

Weiters kann auch durch Bilderkennung die Sicherheit auf Baustellen erhöht werden. Das Ignorieren der Helmpflicht oder das Ablegen von Warnwesten würde durch neuronale Netze sofort bemerkt und gemeldet werden. Eine weitere Möglichkeit stellt das Melden von gefährlichen Arbeiten ohne passender Schutzausrüstung dar. Beispielsweise könnte gewarnt werden, wenn Arbeiter ohne Schutzbrille Trennarbeiten durchführen.

Des weiteren ist es vorstellbar, Bilderkennung bei Beurteilungen von Schäden oder zur Bewertung von Sichtbeton einzusetzen. Inwieweit die menschlichen Unterschiede beim Fotografieren und die verschiedenen Lichteinflüsse von neuronalen Netzen herausgefiltert werden können, ist in einer separaten Forschungsarbeit zu klären. Im Rahmen dieser Arbeit wird lediglich aufgezeigt, bei welchen Tätigkeiten Bilderkennung hilfreich eingesetzt werden kann.

40 Vgl.: Kulkarni, P.; Londhe, S.;, Deo, M.: Artificial Neural Networks for Construction Management: A Review (2017) Seite 72

41 Adeli, H.: Neural Networks in Civil Engineering: 1989-200;(2001), Seite 131f

42 Vgl.: Kulkarni, P.; Londhe, S.;, Deo, M.: Artificial Neural Networks for Construction Management: A Review (2017) Seite 72f

2.2 Wahrscheinlichkeitsnetze

Äußerst zeitintensive Rechenverfahren für das Trainieren von neuronalen Netzen, sowie die Gefahr, bei der Anwendung diverser heuristischer Verfahren, in einem lokalen Minimum stecken zu bleiben, veranlassten Dr. Donald F. Specht in den 1960er-Jahren, eine Klassifikationsmöglichkeit zu entwickeln, die auf statistischen Grundlagen beruht.[43]

Im Jahre 1967 wird das Paper „Generation of Polynomial Discriminant Functions for Pattern Recognition“ veröffentlicht. Hiermit legt Dr. Specht die Grundbausteine für Wahrscheinlichkeitsnetze, welche auch Probability Neural Networks (PNN) genannt werden. Mit Hilfe von Wahrscheinlichkeitsdichtefunktionen sowie dem Bayes Theorem ist es möglich Mustererkennungen oder Klassifizierungen durchzuführen und deren Eintrittswahrscheinlichkeit festzulegen.[44]

> „In the PNN algorithm, a Parzen window is used for nonparametric approximation of the parent probability distribution function (PDF) of each class population and Bayes' rule is then employed to allocate the class with highest posterior probability to new input data, and the probability of misclassification is minimized.“[45]

Mit anderen Worten, unter der Anwendung des Bayes Theorems kann von der Wahrscheinlichkeitsverteilung innerhalb der einzelnen Klassen auf die A-posteriori-Wahrscheinlichkeit der einzelnen Klassen geschlossen werden. Somit bilden die Approximation der Wahrscheinlichkeitsdichte und der Satz von Bayes die Grundlage für das Verständnis von PNN und werden in den nächsten Abschnitten erläutert.

2.2.1 Bayes Theorem

Der Bayes-Klassifikator wählt unter Minimierung des erwarteten Fehlers jene Klasse aus bei der die Eintrittswahrscheinlichkeit am größten ist.[46] Diese wird über den Satz von Bayes, welcher sowohl bei Wahrscheinlichkeiten, als auch bei bedingte Dichten angewendet werden kann, bestimmt.[47] Bei bedingten Dichten ist darauf zu achten, ob deren Variablen einer stetigen oder diskreten Verteilung unterliegen. Viele Klassifikationsaufgaben weisen beide Verteilungen auf. Der Inputvektor ist oftmals stetig verteilt, die einzelnen Klassen hingegen diskret.

Die nachfolgende Gleichung (2-14) zeigt das Bayes Theorem unter Berücksichtigung der Wahrscheinlichkeiten.

$$P(B_n|A) = \frac{P(B_n \cap A)}{P(A)} = \frac{P(B_n)P(A|B_n)}{P(A)} \tag{2-14}$$

43 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 109
44 Vgl.: Specht, D.F..: Generation of Polynomial Discriminant Functions for Pattern Recognition (1967), Seite 308
45 Zeinali, Y.; Story, B. A..: Competitive probabilistic neural network (2017), Seite 1
46 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 109
47 Vgl.: Winzenborg, I.: Bayes'sche Schätztheorie und ihre Anwendung auf neuronale Daten zur Reizrekonstruktion, Diplomarbeit, Seite 7ff

In dieser Gleichung stellt $P(A|B_n)$ die Wahrscheinlichkeit von A unter der Bedingung, dass B eintritt dar. $P(B_n|A)$ wird A-posteriori-Wahrscheinlichkeit genannt und gibt die Wahrscheinlichkeit wieder, mit der beispielsweise ein zu kategorisierendes Muster, welches den Inputvektor x (A) besitzt, in die Klasse B_n fällt. Für $P(B_n)$ ist die Anfangswahrscheinlichkeit jener Klasse einzusetzen. P(A) stellt die totale Wahrscheinlichkeit für das Eintreten von A dar[48] und kann als Skalierungsfaktor verstanden werden, damit die Summe der verschiedenen A-posteriori-Wahrscheinlichkeiten zu 1 wird.[49]

Zusammenfassend folgt eine englische Erläuterung, weil im Forschungsbereich PNN fast ausschließlich englische Fachliteratur vorliegt.

The Bayes formula shows that by observing the value x the prior probability $P(w_j)$ can be converted to the posterior probability $P(w_j|x)$. Thus the probability of being w_j can be easily calculated with equation (2-15) and (2-16) if the value x has been measured.[50]

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)} \tag{2-15}$$

$$p(x) = \sum_{j=1}^{n} p(x|w_j)P(w_j) \tag{2-16}$$

Mit Hilfe des nachfolgenden Beispiels wird nochmals der Satz von Bayes anhand verschiedener Wahrscheinlichkeiten erklärt:

In zwei Obstkisten (Klassen) befinden sich Äpfel und Birnen. In der Kiste (K_1) liegen 3 Äpfel (A) und 7 Birnen (B), in der Kiste (K_2) 6 Äpfel und 4 Birnen. Ein beliebiges Stück Obst wird aus einer Kiste entnommen. Es ist ein Apfel. Wie groß ist die bedingte Wahrscheinlichkeit, dass der Apfel aus K_1 oder K_2 kommt. $P(K_1|A)$ und $P(K_2|A)$ sind somit gefragt. Aufgrund der Gleichheit der Obstkisten betragen die Anfangswahrscheinlichkeiten für das Auswählen einer Kiste jeweils 50 %. Dadurch ergibt sich, dass $P(K_1)$ und $P(K_2)$ den Wert 0,5 annehmen. Die nachfolgenden Gleichungen (2-17), (2-18), (2-19), (2-20) und (2-21) zeigen den Rechenweg.

$$P(A|K_1) = \frac{3}{7+3} = 0,3 \tag{2-17}$$

$$P(A|K_2) = \frac{6}{6+4} = 0,6 \tag{2-18}$$

48 Vgl.: Winzenborg, I.: Bayes'sche Schätztheorie und ihre Anwendung auf neuronale Daten zur Reizrekonstruktion, Diplomarbeit, Seite 7ff

49 Vgl.: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm, Datum des Zugriffs: 07.05.2018

50 Vgl.: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm, Datum des Zugriffs: 07.05.2018

$$P(A) = P(A|K_1)P(K_1) + P(A|K_2)P(K_2) = 0,3 \times 0,5 + 0,6 \times 0,5 = 0,45 \quad (2\text{-}19)$$

$$P(K_1|A) = \frac{P(A|K_1)P(K_1)}{P(A)} = \frac{(0,3 \times 0,5)}{0,45} = 0,3333 \quad (2\text{-}20)$$

$$P(K_2|A) = \frac{P(A|K_2)P(K_1)}{P(A)} = \frac{(0,6 \times 0,5)}{0,45} = 0,6666 \quad (2\text{-}21)$$

Muss eine Entscheidung getroffen werden, wird K2 gewählt, weil ein Apfel mit einer Wahrscheinlichkeit von 66,66 % aus K2 und nur mit 33,33 % aus K1 stammt.

Die selbe Überlegung liegt Wahrscheinlichkeitsnetzen zugrunde. Zunächst stellt sich die Frage, mit welcher Wahrscheinlichkeit ein Input (Apfel) in eine gewisse Klasse fällt. Anschließend wird jene Klasse mit Hilfe der Maximumsuche ausgewählt, die am ehesten zutrifft. Der einzige Unterschied zum obigen Beispiel besteht darin, dass mit Dichtefunktionen gearbeitet wird und nicht mit Wahrscheinlichkeiten.

Für die reine Entscheidungswahl kann der Nenner aus Gleichung (2-15) beiseitegelassen werden, weil dieser bei jeder Klasse den selben Wert annimmt. Somit hat er keinen Einfluss auf die Auswahl des Maximums. Jedoch muss hierbei beachtet werden, dass dadurch das Ergebnis keine Wahrscheinlichkeit mehr für das Eintreten einer Klasse repräsentiert.

Unter Berücksichtigung dieses Punktes und der zusätzlichen Beachtung der Fehlerfunktion wird beispielsweise die Entscheidung für die Auswahl einer aus zwei unterschiedlichen Klassen nach folgender Gleichung (2-22)[51] getroffen.

$$\begin{aligned} d(x) &= w_A \rightarrow \text{if } h_A l_A f_A(x) > h_B l_B f_B(x) \\ d(x) &= w_B \rightarrow \text{if } h_A l_A f_A(x) < h_B l_B f_B(x) \end{aligned} \quad (2\text{-}22)$$

Hierbei stellen w_A und w_B die unterschiedlichen Klassen dar. h_A und h_B beschreiben die Anfangswahrscheinlichkeiten, mit der eine Klassifizierung in die jeweilige Klasse fällt. Die Summe aus h_A und h_B beträgt 1. l_A und l_B repräsentieren die Fehlerfunktion. l_A beschreibt den Fehler, bei dem die Entscheidung für w_A getroffen wird, obwohl das Muster in die Klasse w_B gehört. $f_A(x)$ und $f_B(x)$ beschreiben die jeweiligen Dichtefunktionen innerhalb der Klassen. Die endgültige Klassifizierung fällt immer mit dem größten Wert zusammen. Durch Gleichsetzen beider Ungleichungsseiten kann die Grenze zwischen den Klassen berechnet werden.[52]

[51] Specht, D. F.: Probabilistic Neural Networks (1990), Seite 110

[52] Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 110

2.2.2 Wahrscheinlichkeitsdichtefunktion

Die Wahrscheinlichkeitsverteilung, welche die Eintrittswahrscheinlichkeit eines bestimmten Ereignisses beschreibt, kann sowohl durch die Verteilungsfunktion selbst als auch durch deren Dichte beschrieben werden.[53]

Die Zufallsvariable, welche durch ein zufälliges Ereignis einen davon abhängigen Wert verkörpert, kann einer diskreten oder stetigen Verteilung zugrunde liegen. Die Dichte einer diskreten Zufallsvariable kann beispielsweise in einem Histogramm oder einer Tabelle dargestellt werden, denn sie weist jedem Wert die passende Wahrscheinlichkeit zu. Im Gegensatz dazu, ist das Erkennen der Wahrscheinlichkeit durch Ablesen des Funktionswertes bei einer Dichte mit stetiger Verteilung unmöglich. Die Wahrscheinlichkeit wird durch das Integral über die Verteilung bestimmt. Für beide Dichtefunktionen gilt allerdings, dass die gesamte Fläche unter der Kurve den Wert eins annimmt.[54]

Die nachfolgende Abbildung 2-8 stellt die Dichtefunktion sowie die sich daraus ergebende Summenkurve abhängig von einer diskreten und einer stetigen Verteilung dar.

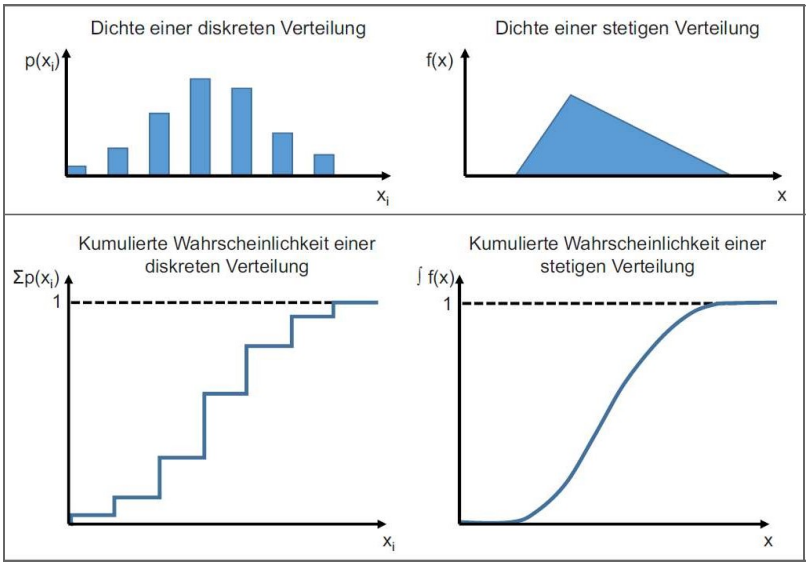


Abb. 2-8 Dichtefunktion mit zugehöriger Wahrscheinlichkeitsverteilung[55]

Neben der Einteilung in stetige und diskrete Verteilungen kann die Dichtefunktion ebenfalls in offene/geschlossene, unimodale/multimodale oder schiefe/symmetrische Verteilungen gegliedert werden.[56] Die Kenntnis vom Verlauf der Verteilungsfunktion innerhalb einer Klasse spielt bei PNN eine entscheidende Rolle. Denn diese bilden die Grundlage für die Anwendung des Bayes Theorems um die Klassifizierung durchzuführen. Die folgende Abbildung 2-9 zeigt die schematische Darstellung einer einseitig (links) und beidseitig offenen (rechts) Verteilung, sowie eine rechtschiefe und linksschiefe Verteilungsfunktion.

53 Vgl.: Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 52
54 Vgl.: Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 53
55 Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 53
56 Vgl.: Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 54ff

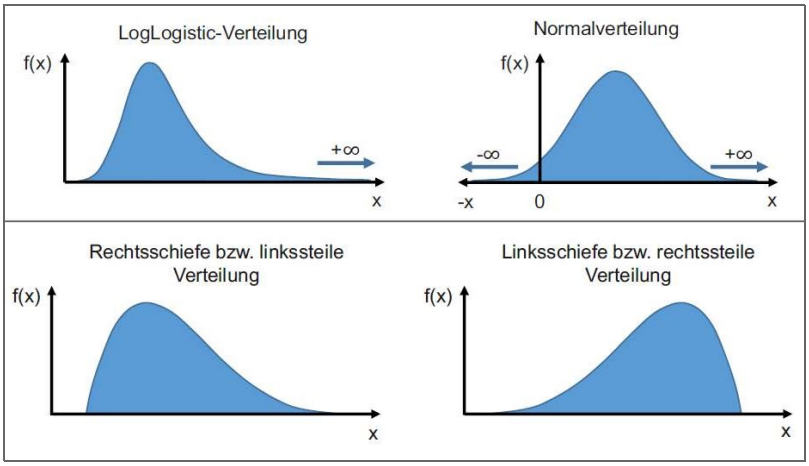


Abb. 2-9 Qualitativer Überblick über Verteilungsfunktionen[57]

Die kommende Abbildung 2-10 verdeutlicht den Unterschied zwischen unimodalen und multimodalen Verteilungen.

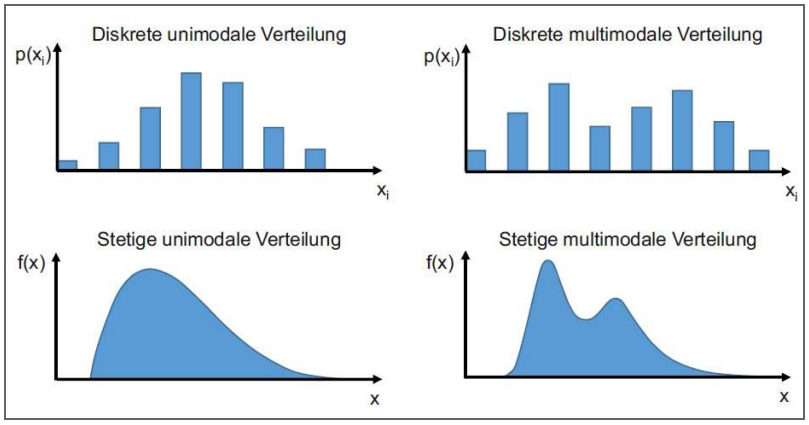


Abb. 2-10 Qualitive Darstellung unimodaler und multimodaler Verteilungen[58]

2.2.3 Kerndichteschätzer

Nach einem kurzen Überblick über die wichtigsten Grundlagen von Dichtefunktionen wird in diesem Abschnitt auf die Approximation der Dichtefunktion einer Stichprobe mit Hilfe des Kerndichteschätzers eingegangen.

Unter dem Kern wird eine messbare Funktion verstanden, welche den gleichen Definitionen, wie die der Dichtefunktion, unterliegt. Somit ist dieser Funktionswert immer positiv und das Integral über dem Definitionsbereich ergibt eins, sprich bei zwei Variablen weist das Volumen unter dem Kern den Wert eins auf.[59]

57 Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 55ff
58 Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 56
59 Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

Der Kerndichteschätzer $\hat{f}_n(x)$, welcher eine Approximation der Dichtefunktion darstellt, wird in folgender Gleichung (2-23) dargestellt. K beschreibt die Kernfunktion und h die Bandbreite. Diese beiden Parameter können frei gewählt werden.[60]

$$\hat{f}_n(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2-23}$$

Es ist ersichtlich, dass der Kerndichteschätzer $\hat{f}_n(x)$ die Summe aus den einzelnen Beiträgen der bekannten Punkte aus der Stichprobe darstellt. Der einzelne Beitrag wird durch den Term (2-24) charakterisiert. Das Integral über dessen Definitionsbereich beträgt 1/n. Somit hat das Integral über die approximierte Dichtefunktion, welche die Summe der n Beiträge ist, den Wert eins.[61]

$$\frac{1}{nh}K\left(\frac{x - x_i}{h}\right) \tag{2-24}$$

Für die Wahl des Kernes stehen verschiedene Möglichkeiten zur Verfügung. Beispiele sind hierfür der Rechteckskern, der Gauß-Kern, der Epanechnikov-Kern oder der Bisquare-Kern.[62]

Aufgrund der Tatsache, dass die anfänglichen Forschungsarbeiten in Bezug auf Wahrscheinlichkeitsnetze einen Gauß-Kern aufwiesen, wird dieser näher beleuchtet.

Der Gauß-Kern beschreibt die Dichte der Normalverteilung und wird in Gleichung (2-25) gezeigt. Somit stellt der Kerndichteschätzer mit diesem Kern das arithmetische Mittel der einzelnen Dichten der Standardnormalverteilung dar.[63]

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$
$$\frac{1}{h}K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}h} e^{-\frac{(x - x_i)^2}{2h^2}} \tag{2-25}$$

Parzen (1962) zeigte, dass der quadratische Fehler zwischen diesem Kerndichteschätzer und der tatsächlichen Verteilungsfunktion mit zunehmender Anzahl an bekannten Punkten abnimmt, bis er schließlich bei n gegen unendlich zu Null wird. Weiters gilt es anzumerken, dass trotz Verwendung eines Gauß-Kernes jede beliebige Dichtefunktion nachgebildet werden kann.[64] Diese Nachbildung wird allerdings immer eine Approximation bleiben, weil die Stichproben begrenzt sind. Jedoch kann der Fehler mit zunehmender Datenmenge minimiert werden. Der Fehlerwert Null wird nicht erreicht.

60 Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

61 Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

62 Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

63 Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

64 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 110f

Der allgemeine mehrdimensionale Kerndichtenschätzer, dem ein Gauß Kern zugrunde liegt, wird in Gleichung (2-26) gezeigt. Die Variable i stellt die Nummer eines Elements aus der Stichprobe dar, m repräsentiert die gesamte Anzahl an Elementen einer Stichprobe, X_{Ai} beschreibt das ite Element aus Klasse A, Sigma ist mit einem Glättungsfaktor (Bandbreite) gleichzusetzen und p vertritt die Dimension von X_{Ai}.[65]

$$f_A(X) = \frac{1}{(2\pi)^{p/2}\sigma^p m} \sum_{i=1}^{m} e^{-\frac{(X-X_{Ai})^t(X-X_{Ai})}{2\sigma^2}} \tag{2-26}$$

Die Ermittlung des Glättungsfaktors stellt kein triviales Problem dar. Er kann jedoch experimentell recht einfach ermittelt werden, weil die Funktion der Fehlerrate abhängig von Sigma sehr flach verläuft. Somit hat eine kleine Änderung von Sigma keine allzu große Auswirkung auf die Fehlerrate.[66]

Das folgende Beispiel zeigt wie mit Hilfe des Kerndichteschätzers eine Kategorisierung eines eindimensionalen Inputvektors durchgeführt werden kann. Folgende Werte sind den Klassen A und B zugeordnet:

	x_1	x_2	x_3	x_4	x_5
Klasse A	0	2	3	4	6
Klasse B	6	7	8	9	

Tab. 2-1 Zuordnung von x in die Klasse A und B

In Welche Klasse fällt x = 5 unter der Bedingung Sigma = 1?

Um diese Frage zu beantworten wird zunächst die Dichtefunktion der einzelnen Klassen mittels Formel (2-26) approximiert um anschließend durch Einsetzen des gefragten Wertes das Maximum bestimmen zu können.

$$f_A(x) = \frac{1}{(2\pi)^{1/2} \times 1^1 \times 5} \sum_{i=1}^{5} e^{-\frac{(x-x_{Ai})^t(x-x_{Ai})}{2\times 1^2}} = \tag{2-27}$$

$$= \frac{1}{5\times\sqrt{2\pi}} \times \left[e^{-\frac{(x-0)^2}{2}} + e^{-\frac{(x-2)^2}{2}} + e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x-4)^2}{2}} + e^{-\frac{(x-6)^2}{2}} \right]$$

$$f_B(x) = \frac{1}{(2\pi)^{1/2} \times 1^1 \times 4} \sum_{i=1}^{4} e^{-\frac{(x-x_{Ai})^t(x-x_{Ai})}{2\times 1^2}} = \tag{2-28}$$

$$= \frac{1}{4\times\sqrt{2\pi}} \times \left[e^{-\frac{(x-6)^2}{2}} + e^{-\frac{(x-7)^2}{2}} + e^{-\frac{(x-8)^2}{2}} + e^{-\frac{(x-9)^2}{2}} \right]$$

65 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 110f

66 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 113

$$f_A(5) = 0,11$$
$$f_B(5) = 0,075 \qquad (2\text{-}29)$$

Die Annäherung der Dichtefunktion wird in Gleichung (2-27) für Klasse A und in Gleichung (2-28) für Klasse B gezeigt. Anschließend wird der zu untersuchende Input x = 5 in beide Verteilungsfunktionen eingesetzt. Das daraus berechnete Ergebnis wird in Gleichung (2-29) dargestellt.

Abbildung 2-11 zeigt den Verlauf der Dichtefunktionen, die mittels Kerndichteschätzers berechnet worden sind. Die blaue Funktion repräsentiert die Verteilung der Klasse A, die rote die der Klasse B. An der Stelle x = 5 weist die Dichte aus Klasse A einen höheren Wert als Klasse B auf. Somit ist Klasse A zu wählen.

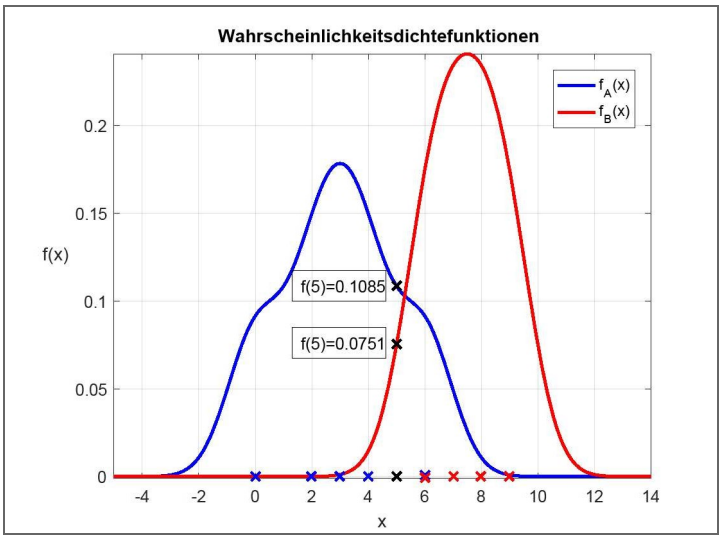


Abb. 2-11 Verteilungsfunktionen innerhalb der Klassen A und B

2.2.4 Implementierung

Im folgenden Abschnitt wird gezeigt, wie die zuvor erklärten Grundlagen in einem neuronalen Netz umgesetzt werden.

Grundsätzlich besteht ein Wahrscheinlichkeitsnetz aus mehreren Subnetzten, wobei jeder Teil einen Kerndichteschätzer für eine mögliche Klasse darstellt.[67] Um das Wahrscheinlichkeitsnetz einfach und bestmöglich erklären zu können, wird der Aufbau anhand eines Netzes, welches zwei Entscheidungsklassen besitzt, verdeutlicht. Abbildung 2-12 zeigt ein solches.

[67] Vgl.: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

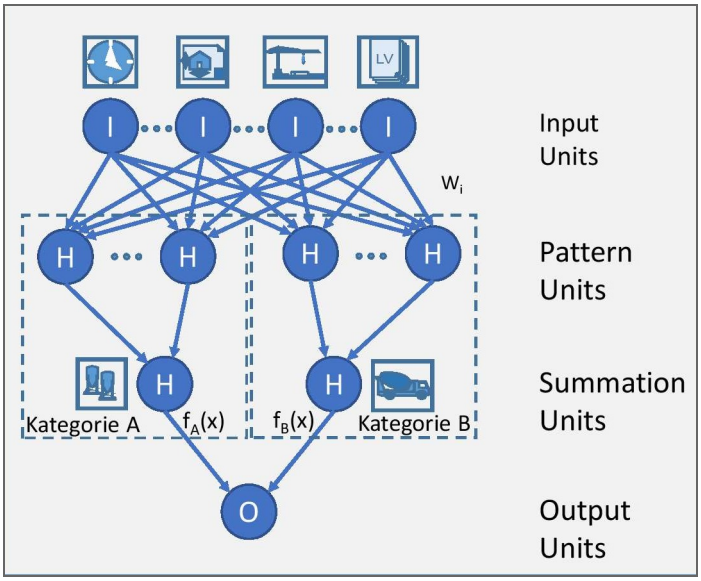


Abb. 2-12 Aufbau eines Wahrscheinlichkeitsnetzes[68]

PNN weisen einen vier schichtigen Aufbau auf. Sie bestehen aus einem Input-Layer, zwei Hidden-Layer und einem Output-Layer. Die englische Bezeichnung für die Strukturierung der Neuronen lautet: Input Units, Pattern Units, Summation Units und Output Units.[69]

Die erste Schicht hat die Aufgabe, jeden Inputparameter aus dem Inputvektor X, welcher bis zu n Dimensionen besitzen kann, in einem eigenen Neuron aufzunehmen. Weiters wird jedes dieser Inputneuronen mit den Neuronen in der zweiten Schicht verbunden. Dadurch wird der gesamte Input an die einzelnen Neuronen der 2.Schicht weitergegeben.[70] Mit Hilfe folgender Abbildung 2-13 wird der Aufbau der Neuronenstruktur in diesen zwei Layern und deren mathematischen Zusammenhänge, welche anschließend erklärt werden, verdeutlicht.

68 in Anlehnung an Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112
69 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f
70 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f

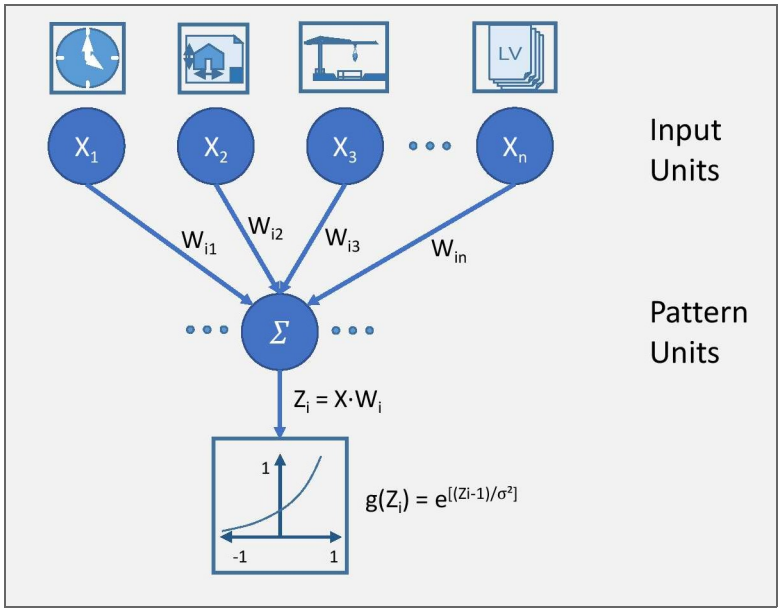


Abb. 2-13 Aufbau der Neuronen in den ersten zwei Layern[71]

Die Verbindungen zwischen Schicht 1 und einem Neuron aus Schicht 2 werden als Vektor W_i dargestellt. Somit ergibt sich der Input Z_i der zweiten Schichte aus der Multiplikation von $X*W_i$. Die darauffolgende nichtlineare Aktivierungsfunktion der Pattern Units verarbeitet den Input weiter. Hierbei wird meistens eine Exponentialfunktion eingesetzt, welche die Form gemäß Term (2-30) aufweist.[72]

$$e^{\frac{(Z_i - 1)^2}{\sigma^2}} \tag{2-30}$$

Es sei hier angemerkt, dass auch andere Funktionen als Aktivierungsfunktion verwendet werden können. Unter der Voraussetzung, dass sowohl X als auch W_i Einheitsvektoren sind, lässt sich die Exponentialfunktion, wie in Term (2-31) gezeigt, folgendermaßen umschreiben.[73]

$$e^{-\frac{(W_i - X)^t (W_i - X)}{2\sigma^2}} \tag{2-31}$$

Dies Funktion ist bereits bekannt. Sie repräsentiert den Kerndichteschätzer dem ein Gauß-Kern zugrunde liegt. Diese Umformung soll zeigen, dass die vorherige Multiplikation und die Anwendung der Aktivierungsfunktion einen an der Stelle X ausgewerteten Gauß-Kern, welcher W_i als Zentrum hat, darstellt.[74]

Im Gegensatz zur zweiten Schicht, in der für jeden Trainingsfall ein eigenes Neuron platziert wird, existiert im dritten Layer für jede zu wählende Klasse ein Neuron. Dieses summiert die klassenangehörigen Outputs, welche den

71 in Anlehnung an: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112
72 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f
73 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f
74 Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f

ausgewerteten Gauß-Kernen entsprechen, von dem vorherigen Layer auf. Somit entsteht in den Summation Units die an der Stelle X ausgewertete approximierte Dichtefunktion der einzelnen Klassen.[75]

In der vierten Schichte wird schlussendlich mit Hilfe des Bayes Theorems entschieden, welche Klasse auszuwählen ist. In diesem Beispiel, bei dem die Klassen A und B zur Auswahl stehen, wird die Summe aus Klasse B mit dem Faktor C, welcher negativ ist, multipliziert. Dieser besteht aus dem Verhältnis der Anfangswahrscheinlichkeiten multipliziert mit dem Verhältnis der Fehlzuweisungen dividiert durch das Verhältnis der Trainingsfälle. Gleichung (2-32) stellt diesen dar.[76]

$$C = -\frac{h_B l_B}{h_A l_A} \cdot \frac{n_A}{n_B} \tag{2-32}$$

Anschließend werden beide Werte addiert. Stellt die Summe einen positiven Wert dar, ist Kategorie A auszuwählen. Wird das Ergebnis der Addition negativ, fällt die Entscheidung auf Klasse B.[77] Dies wird in Abbildung 2-14 verdeutlicht.

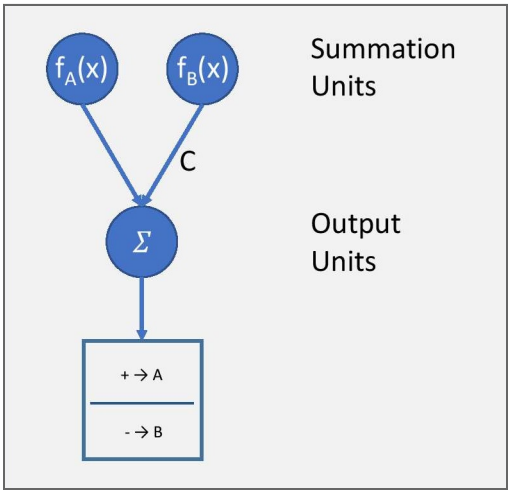


Abb. 2-14 Darstellung Output Unit[78]

Das Training eines Wahrscheinlichkeitsnetzes ist mit sehr geringem Zeitaufwand verbunden. Hierbei muss lediglich für jeden Trainingsfall i ein Neuron in der 2. Schicht erstellt und die zugehörigen Gewichte W_i mit dem Inputvektor dieses Trainingsbeispiels gleichgesetzt werden.[79]

[75] Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 111f
[76] Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112
[77] Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112f
[78] in Anlehnung an: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112
[79] Vgl.: Specht, D. F.: Probabilistic Neural Networks (1990), Seite 112f

2.2.5 Einsatzmöglichkeiten

Wahrscheinlichkeitsnetze eigenen sich sehr gut für Kategorisierungsaufgaben und für Mustererkennungsfragestellungen. Eine der Stärken liegt unbestritten in der kurzen Zeit, die für das Trainieren des Netzes gebraucht wird. Jedes Trainingsbeispiel wird durch ein Neuron in dem ersten Hidden-Layer dargestellt. Somit werden keine zeitintensiven Rechenvorgänge benötigt um dem Netz etwas zu erlernen oder um neue Erkenntnisse miteinzubeziehen. Jedoch wurde dieser Vorteil in den letzten Jahren aufgrund der rasanten Weiterentwicklung der Rechenkapazitäten sehr stark minimiert. Es können immer komplizierter und größer werdende Netze gelöst werden. Abbildung 2-15 verdeutlicht den steilen Anstieg der Rechengeschwindigkeit der letzten Jahre. Diese wird in FLOPS (Floating Point Operations Per Second) gemessen und stellt ein Maß für die Leistungsfähigkeit von Computern dar. Mit diesem Anstieg einhergehend nimmt auch die Bedeutung von neuronalen Netzen zu. Es können immer mehr Einflüsse berücksichtig und größere Datenmengen für Trainingszwecke analysiert werden.

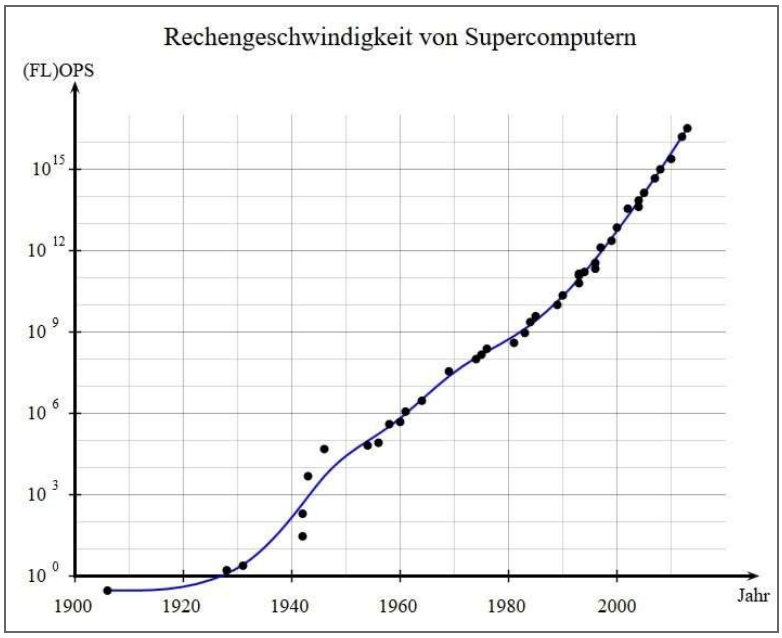


Abb. 2-15 Entwicklung der Rechengeschwindigkeit[80]

[80] https://image.jimcdn.com/app/cms/image/transf/none/path/s493dce01392ad78f/image/ieac8ba1c54de1863/version/1449164092/image.jpg, Datum des Zugriffs: 13.05.2018

Für Aufgabenstellungen aus dem Bereich Klassifizierung und Mustererkennung sind PNN besonders gut geeignet. Folgende Anwendungsmöglichkeiten wurden unter anderem bereits untersucht: Klassifizierung von Schiffen mit Hilfe von Sonarsignalen (Chen,Lee & Lin, 2000), Vorhersage von Leukämie und embryonalen Tumoren (Huang & Liao, 2004), die Vorhersage von Betondruckfestigkeiten (Kim, Lee & Chang, 2005), die Vorausbestimmung von Erdbebenstärken (Adeli & Panakkat, 2009)[81] oder im Bereich der Bodenmechanik die Vorhersage von Schubfestigkeiten (Kiran, Lal, Tripathy, 2016).

Im Bereich der Bauwirtschaft können Wahrscheinlichkeitsnetze bei Fragestellungen eingesetzt werden, bei denen die beste Lösungsmöglichkeit aus einer bestimmten Anzahl an möglichen Lösungen auszuwählen ist. Hierbei muss erwähnt werden, dass diese Netze nicht eingesetzt werden können, um neue Probleme zu lösen. Sie dienen lediglich als unterstützendes Hilfsmittel für das Treffen von Entscheidungen, indem das erlernte Wissen aus bereits abgeschlossenen Projekten auf die aktuelle Aufgabenstellung angewendet wird.

Um Wissen erfolgreich zu erlernen ist ein umfangreiches Trainigsdatenset notwendig. Abhängig von der Aufgabenstellung sind mindestens 100, 250 bis hin zu mehreren tausend Trainingsdaten erforderlich um ein akzeptables Ergebnis zu erzielen. Erst beim Erreichen einer unendlichen Anzahl von Trainingsdaten ist es möglich, den quadratischen Fehler bei der Approximation der Dichtefunktionen innerhalb einer Klasse auf Null zu reduzieren (Parzen, 1962).

Aufgrund der Einzigartigkeit eines Bauwerks und unterschiedlichster Baustellenbedingungen stellt die Datenakquisition ein nicht triviales Unterfangen dar. Ist es möglich auf ausreichend Daten zurückzugreifen, können Wahrscheinlichkeitsnetze erfolgreich eingesetzt werden. Im folgenden Absatz werden potenzielle Anwendungsmöglichkeiten vorgestellt, ohne näher auf die erforderlichen Inputparameter einzugehen. Diese müssen in einer separaten Untersuchung erhoben und nach deren Signifikanz ausgewählt werden.

Große Potenziale für die Anwendung von PNN liegen in der Verfahrensauswahl. Beispielsweise können diese Netze als Entscheidungshilfe für die Auswahl von Schalungssystemen herangezogen werden. Konventionelle Schalung, Rahmenschalung oder Deckentische werden zuerst als Klassen definiert um anschließend einen Inputvektor zuordnen zu können. Weiters ist es möglich, zu unterscheiden, ob eine Kletterschalung oder Gleitschalung zu bevorzugen wäre oder welche Schalungshaut und welches Trennmittel als geeignet erscheint. Nach der Wahl des Schalungssystems stellt sich bei größeren Baustellen die Frage, ob Transportbeton bestellt oder eine mobile Mischanlage errichtet und auf welche Art der Beton eingebaut werden soll. Diese Fragestellungen können ebenfalls mit Hilfe von Wahrscheinlichkeitsnetzen beantwortet werden.

Die Strategieauswahl eines Unternehmens kann ebenso von PNNs unterstützt werden. Sowohl bei der Auswahl zukünftiger Standorte im Zuge einer Weiterentwicklung eines Unternehmens, als auch bei der Entscheidung sich für einen potenziellen Auftrag zu bewerben, kann dieses Netz als Ratgeber fungieren und aufgrund vergangener Handlungen eine Empfehlung abgeben.

[81] Vgl.: Adeli, H.; Panakkat,A.:A probabilistic neural network for earthquake magnitude prediction (2009), Seite 1018f

Gerade in der Kalkulationsphase müssen für die Preisermittlung Entscheidungen über den Einsatz von Bauverfahren getroffen werden, obwohl diese noch gar nicht feststehen. Hierbei wäre es möglich mit Hilfe von Wahrscheinlichkeitsnetzen, die durch erfolgreiche Projekte trainiert wurden, das passendste Verfahren auszuwählen. Die Baugrubensicherung stellt hierfür ein sehr gutes Beispiel dar. Abhängig von einem bestimmten Inputvektor könnte ein trainiertes Netz vorschlagen, ob Spundwände, DSV-Körper, Schlitzwände, Schmalwände, Böschungen, Trägerbohlwände oder Bohrpfähle verwendet werden sollen. Sollte dem Kalkulanten ein Bauteilaufbau noch nicht bekannt sein, ist es auch hier möglich, den wahrscheinlichsten aus vergangenen Projekten, passen zu den neuen Projektkennwerten, festzulegen. Schon alleine anhand dieser Beispiele wird ersichtlich, dass Wahrscheinlichkeitsnetze viele Anwendungsmöglichkeiten in der Kalkulation aufweisen.

Nicht nur Auftragnehmer, sondern auch Planer und Auftraggeber können sich diesem Hilfsmittel bedienen. Beispielsweise könnten mit diesen Netzen Sanierungsmaßnahmen abgeschätzt werden. Weiters ist es auch vorstellbar, ein PNN zu trainieren, welches vorschlägt, ob eine Asphalt oder Betonfahrbahn gebaut werden soll.

Zusammenfassend kann gesagt werden, dass Wahrscheinlichkeitsnetze als hilfreiches Werkzeug in der Entscheidungsfindung eingesetzt werden können. Jedoch stellen sie kein Wundermittel für Problemlösungen dar. Die endgültige Auswahl wird immer der Verantwortliche selbst treffen müssen. Das trainierte Netz gibt nur Wahrscheinlichkeiten wieder, die etwas über die Zuordnungen des Inputs in die jeweiligen Klassen aussagen. Die Anwendung ist nicht auf spezielle Bereiche begrenzt. Es muss jedoch eine ausreichende Datengrundlage für die zu analysierende Fragestellung vorhanden sein, ansonsten kann kein zielführendes Ergebnis bereitgestellt werden.

2.3 General Regression Neural Network

In diesem Abschnitt wird auf das von Specht im Jahre 1991 vorgestellte neuronale Regressionsnetz, welches im Englischen die Bezeichnung General Regression Neural Network (GRNN) aufweist, eingegangen. Bevor jedoch die mathematischen Hintergründe und die Implementierung vorgestellt werden, wird zunächst der Begriff Regression definiert:

„Die Regression gibt einen Zusammenhang zwischen zwei oder mehr Variablen an. Bei der Regressionsanalyse wird vorausgesetzt, dass es einen gerichteten Zusammenhang gibt, das heißt, es existieren eine abhängige Variable und mindestens eine unabhängige Variable. Welche Variablen abhängig und welche unabhängig sind, muss aufgrund inhaltlich logischer Überlegungen identifiziert werden können. Mit Hilfe der Regressionsanalyse kann eine Regressionsfunktion errechnet werden, welche die Abhängigkeit der beiden Variablen beschreibt.“[82]

Das Besondere beim Erstellen von Regressionsananlysen durch GRNN ist, dass keine weitere Kenntnis über die Form der zu approximierenden Funktion von Nöten ist. Die linearen und nicht-linearen Abhängigkeiten der unterschiedlichen Inputvariablen müssen zu Beginn nicht bekannt sein, weil diese durch Dichtefunktionen repräsentiert werden. Somit wird ein möglicher Fehler, der durch das anfängliche Voraussetzten eines Funktionsverlaufs entsteht, beispielsweise eines linearen Verlaufs, eliminiert.[83]

2.3.1 Mathematischer Hintergrund

Für das Aufstellen der Regressionsfunktion wird die gemeinsame Dichtefunktion f(x,y) benötigt, wobei x den allgemeinen Inputvektor und y das dazugehörige skalare Output darstellt. Ist ein bestimmter Inputvektor X aus einer Stichprobe und eine bereits bekannten gemeinsame Wahrscheinlichkeitsdichtefunktion von Input und Output vorhanden, kann durch Gleichung (2-33) auf den bedingten Erwartungswert von y unter gegebenem X, E[y|X], geschlossen werden.[84]

$$E[y|X] = \frac{\int_{-\infty}^{\infty} y f(X, y) dy}{\int_{-\infty}^{\infty} f(X, y) dy} \tag{2-33}$$

Unter Erwartungswert wird folgendes verstanden:

„Der Erwartungswert E(X) kann als jene Zahl interpretiert werden, die die Zufallsvariable im Mittel annimmt. Wird der Mittelwert aus den Ergebnissen eines Versuchs gebildet, konvergiert der Mittelwert gegen den Erwartungswert. Der Erwartungswert stellt eine Größe dar, mit der bei einer großen Anzahl an Versuchen zu rechnen ist.“[85]

[82] https://de.statista.com/statistik/lexikon/definition/112/regression/, Datum des Zugriffs: 24.05.2018
[83] Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 568
[84] Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569
[85] Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Seite 58

Gleich wie bei Wahrscheinlichkeitsnetzen wird auch in diesem Fall der Kerndichteschätzer angewandt, um die unbekannte gemeinsame Dichtefunktion zu approximieren. Der mehrdimensionale Fall für die Annäherung der Verteilungsfunktion wird in der folgenden Gleichung (2-34) dargestellt. X_i und Y_i stellen hierbei die bekannten Datenpaare der Regressionsanalyse dar. n verkörpert die Anzahl der bekannten Input-Output-Paare und p repräsentiert die Dimension des Inputvektors X.[86]

$$\hat{f}(X,Y) = \frac{1}{(2\pi)^{(p+1)/2}\sigma^{(p+1)}} \frac{1}{n}\sum_{i=1}^{n} e^{-\frac{(X-X_i)^t(X-X_i)}{2\sigma^2}} e^{-\frac{(Y-Y_i)^2}{2\sigma^2}} \tag{2-34}$$

Somit bildet die Annäherung der Verteilungsfunktion die Summe der Wahrscheinlichkeiten der einzelnen Stichprobenzugehörigkeit X_i und Y_i ab. Durch Einsetzen von Gleichung (2-34) in Gleichung (2-33) und anschließendem Umformen ergibt sich der erwartete gemittelte Y-Wert bei gegebenem X. Dieser wird in Gleichung (2-35) dargestellt.[87]

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} e^{-\frac{(X-X_i)^t(X-X_i)}{2\sigma^2}} \int_{-\infty}^{\infty} y e^{-\frac{(Y-Y_i)^2}{2\sigma^2}} dy}{\sum_{i=1}^{n} e^{-\frac{(X-X_i)^t(X-X_i)}{2\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{(Y-Y_i)^2}{2\sigma^2}} dy} \tag{2-35}$$

Im nächsten Schritt wird die Skalarfunktion D_i^2, welche in Gleichung (2-36) gezeigt wird, definiert. Diese beschreibt die quadratische Distanz zwischen Input X und Trainingsdatensatz X_i.[88]

$$D_i^2 = (X-X_i)^t(X-X_i) \tag{2-36}$$

Nach Einsetzen dieser Skalarfunktion und lösen der beiden Integrale, vereinfacht sich der bedingte Erwartungswert auf folgende Funktion, die in Gleichung (2-37) dargestellt wird. [89]

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y_i e^{-\frac{D_i^2}{2\sigma^2}}}{\sum_{i=1}^{n} e^{-\frac{D_i^2}{2\sigma^2}}} \tag{2-37}$$

Weiters wiesen Parzen (1962) und Cacoullos (1966) nach, dass der Kerndichteschätzer aus Gleichung (2-34), welcher Gleichung (2-37) zu Grunde liegt, konsistent ist. Das heißt, dass der Fehler mit zunehmender Probenanzahl kleiner wird, bis er schließlich bei n gegen unendlich zu Null konvergiert. [90]

86 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569
87 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569
88 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569
89 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569

Bei genauerer Betrachtung der Regressionsfunktion (2-37) wird ersichtlich, dass der zu berechnende Y-Wert aus den gewichteten Outputdaten der Stichprobe Y_i besteht. Die Gewichtung selbst ist abhängig von einer Exponentialfunktion, in welche die euklidische Distanz von X eingesetzt wird. Abhängig von der Distanz D_i^2 und vom gewählten Glättungsfaktor σ fällt die Gewichtung f(x) unterschiedlich stark aus. Dies wird in Abbildung 2-16 verdeutlicht. Anhand des roten Funktionsverlaufs ist zu erkennen, dass bei größeren Glättungsparametern die Exponentialfunktion gedehnt wird. Beim Aufsummieren der einzelnen Komponenten in Gleichung (2-37) wird somit die Regressionsfunktion mit steigendem Glättungsfaktor immer abgerundeter, bis sie schließlich zu einer mehrdimensionalen Normalverteilung konvergiert. Wird Sigma immer kleiner führt dies zu einer immer unebener werdenden Form. Wenn der Glättungsfaktor gegen Null läuft, wird jenes Y_i zum gesuchten Y bei dem die Distanz zwischen X und X_i am kleinsten ist, sprich die am nahegelegenste Stichprobe darstellt.[91]

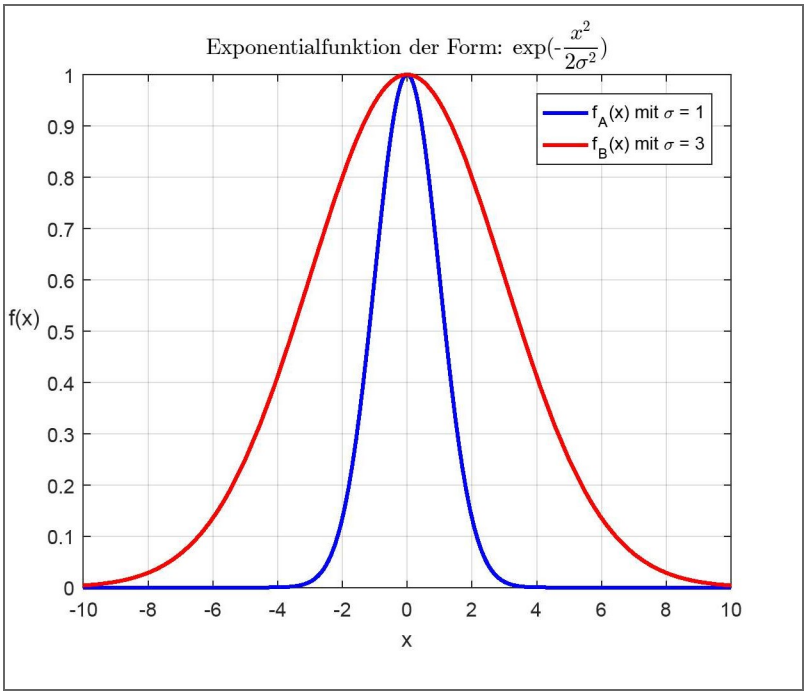


Abb. 2-16 Schematische Darstellung Exponentialfunktion

Um die Regressionsfunktion in den vorhandenen Datensatz bestmöglich einzupassen, ist es notwendig, den optimalen Glättungsfaktor, sprich die geeignetste Bandbreite für den Kerndichteschätzer, zu finden. Dies wird oftmals mit dem Holdout-Verfahren gemacht. Hierbei wird für ausgewählte Sigmas ein Datensatz aus der Stichprobe herausgenommen und anhand der übrigen das Netz trainiert, um anschließend den Fehler zwischen der Stichprobe und dem aus dem Netz berechneten Wert in einer Tabelle zu speichern. Nachdem jeder einzelne Datensatz einmal herausgenommen worden ist, wird anschließend die mittlere quadratische Abweichung bestimmt. Schlussendlich wird jener Glättungsfaktor mit dem geringsten Fehler ausgewählt. Oftmals weist der Graph, welcher die mittlere quadra-

[90] Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569
[91] Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 569

tische Abweichung über dem dazugehörigen Sigma beschreibt, eine sehr flache Steigung um das Fehler-Minimum auf. Somit ist die Auswahl eines passenden Glättungsparameters nicht all zu schwer.[92]

Weiters verdeutlicht Abbildung 2-16, dass jede Distanz zwischen X und X_i berücksichtigt wird. Lediglich weisen sie eine unterschiedliche Wichtigkeit auf. Je weiter X und X_i auseinanderliegen, desto geringer wird der Einfluss des dazugehörigen Y_i-Wertes. Wird die Distanz zu Null, sprich X und X_i sind ident, nimmt die Gewichtung den Wert 1 an.

Einen weiteren wichtigen Schritt für das trainieren eines neuronalen Regressionsnetzes stellt das Skalieren der Inputvariablen, welches vor Trainingsbeginn zu erfolgen hat, dar. Die gewählte Bandbreite des Kerndichteschätzers ist für jede Dimension dieselbe. Aus diesem Grund müssen die Inputvariablen ähnliche Wertebereiche oder Varianzen aufweisen. Es sei jedoch auch darauf hingewiesen, dass ein exaktes Anpassen des Inputs nicht notwendig ist. Die Skalierungsvariablen müssen nicht nach jedem Hinzufügen eines neuen Trainingexamples angepasst werden. Lediglich beim Spezialfall, dass die Anzahl der Stichproben gegen unendlich und Sigma gegen Null geht, ist keine Skalierung notwendig.[93]

2.3.2 Implementierung

Bei der Implementierung eines neuronalen Regressionsnetzes wird auf einen vierschichtigen Aufbau zurückgegriffen. Dieser besteht aus einer Eingabe-, einer Muster-, einer Summierungs- und einer Ausgabeschicht. Diese Struktur und die nachfolgende Erklärung wird in Abbildung 2-17 verdeutlicht. [94]

Im Inputlayer befindet sich für jede Variable des Inputvektors ein einzelnes Neuron, welches die skalierten Eingaben aufnimmt und an die zweite Schicht weitergibt. Somit dienen diese lediglich der fachgerechten Verteilung des Inputs an die Neuronen der zweiten Schichte, auch Pattern Units genannt, welche die übergebenen Daten weiterverarbeiten.[95]

Hierbei werden zunächst die Entfernungen des neuen Inputvektors zu den jeweilig bekannten Trainingsvektoren ermittelt. Die einzelne Differenzberechnung, $X-X_i$, wird in den Pattern Units, die jeweils ein Musterexemplar aus dem Trainingsdatensatz verkörpern, durchgeführt. Anschließend werden entweder die quadratischen oder die absoluten Abweichungen aufsummiert. Im nächsten Schritt wird auf dieses Ergebnis, welches die Distanz zwischen X und X_i repräsentiert, die nichtlineare Aktivierungsfunktion des Neurons angewandt. An dieser Stelle sei darauf hingewiesen, dass eine Exponentialfunktion zwar am häufigsten als Aktivierungsfunktion verwendet wird, aber nicht die einzige mögliche Funktion darstellt. Vergleichend mit dem zuvor vorgestellten mathematischen Hintergrund verkörpert somit der Output eines Neurons der zweiten Schicht einen Summanden aus dem Nenner der Gleichung (2-37).[96]

In der dritten Schicht, auch Summation-Layer genannt, werden daraufhin

92 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 570
93 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 570
94 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572f
95 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572f
96 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572f

Zähler und Nenner der Regressionsfunktion (2-37) nachgebaut. Bei einem neuronalen Netz, welches einen Output generiert, benötigt der Summation-Layer zwei Neuronen, welche jeweils mit allen Einheiten der darüberliegenden Schichte verbunden sind. Eines der beiden summiert jeden Output der Pattern Units auf, um den Nenner zu erhalten. Das andere addiert die Produkte aus Output o des Neurons i mit dem dazugehörigen Y_i des Trainingsdatensatzes zusammen. Die Multiplikation, $Y_i*o_i(x)$, welche den Einfluss der jeweiligen Y_i beschreibt, erfolgt über die Gewichtsmatrix. Die Verbindungen von den Pattern Units zu diesem Neuron weisen genau die dazugehörigen Y_i-Werte auf.[97]

Schlussendlich wird im Output-Layer noch die Division durchgeführt, um das gewünschte Y(X) zu erhalten. Sollten mehrere Y-Werte gefragt sein gilt es lediglich einen neuen Zähler für Gleichung (2-37) und ein weiteres Ergebnis zu berechnen. Hierfür wird einerseits in der Summation-Layer ein Neuron hinzugefügt, dessen Verbindungen zur zweiten Schichte die $Y_{i,2}$-Werte aufweisen. Andererseits muss ein weiteres Outputneuron, um die Division berechnen zu können, ergänzt werden.[98]

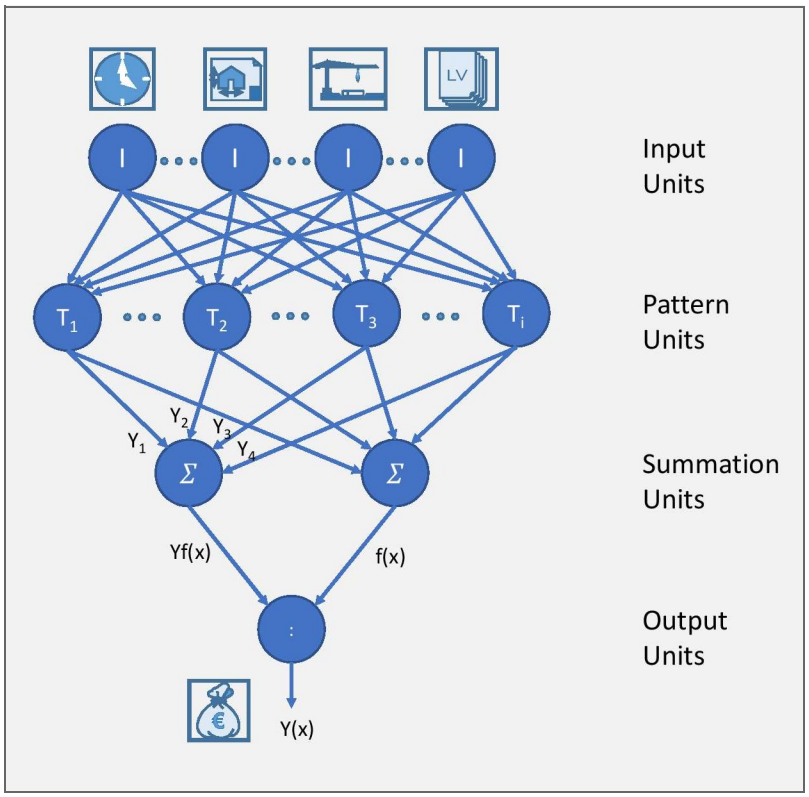


Abb. 2-17 Schematische Darstellung neuronales Regressionsnetz

97 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572f
98 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 573

2.3.3 Beispiel zu GRNN

In folgendem Abschnitt wird eine Regressionsanalyse aufbauend auf die Prinzipien von GRNN dargestellt. Für das bessere Verständnis der mathematischen Hintergründe wird kein neuronales Netz programmiert, sondern die zuvor erläuterte Theorie anhand eines Beispiels in Matlab[99], einer Software für mathematische Berechnungen, umgesetzt.

Als Aufgabenstellung wird das Finden einer Regressionsfunktion für die Punkte, welche in Tabelle 2-2 gezeigt werden, gesetzt. Weiters ist der Punkt X = 0 auszuwerten und dieses Ergebnis mit dem Wert der Ursprungsfunktion zu vergleichen.

Lfd. Nr.	X	Y
0	A	B
1	-10,00	1,36
2	-7,00	1,08
3	-5,00	0,71
4	-3,00	0,00
5	-2,00	-0,57
6	-1,00	-1,00
7	0,30	0,64
8	1,00	2,00
9	2,00	2,86
10	3,00	3,00
11	5,00	2,86
12	7,00	2,69
13	10,00	2,52

Tab. 2-2 Punkte für Regressionsfunktion

Die Zahlen der Spalte B, welche die Y-Koordinaten repräsentiert, sind nicht willkürlich gewählt, sondern ergeben sich durch Einsetzen der X-Werte in Gleichung (2-38). Somit kann anschließend ein Vergleich zwischen Regressionsfunktion und Ursprungsfunktion angestellt werden.

$$y(x) = \frac{2x^2 + 6x}{x^2 + 3} \tag{2-38}$$

Die folgende Abbildung 2-18 zeigt die Funktion (2-38) im Intervall von -10 bis +10. Sie weist ein Minimum an der Stelle x = -1 mit y(-1) = -1 und einen Hochpunkt an der Stelle x = 3 mit y(3) = 3 auf. Asymptotisch nähert sie sich auf beiden Seiten dem y-Wert 2 an.

99 https://de.mathworks.com/products/matlab.html, Datum des Zugriffs: 27.05.2018

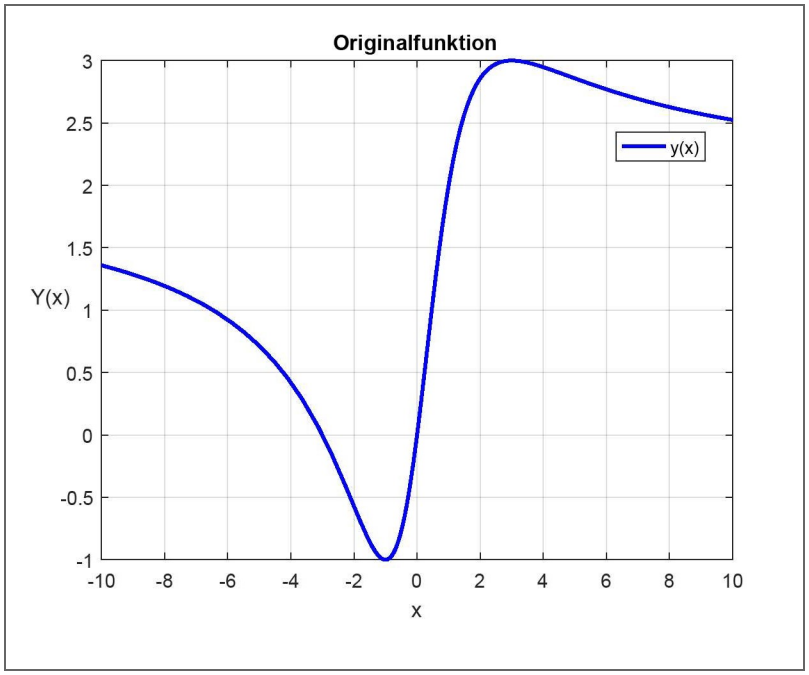


Abb. 2-18 Plot der Ursprungsfunktion

Um die Originalfunktion aus Gleichung (2-38) mit Hilfe der gegebenen Punkte nachzubilden wird folgender Programmiercode, welcher in Abbildung 2-19 dargestellt ist, benötigt.

```
function  bspgrnn
fA=@(x) (2*x^2+6*x)/(x^2+3);
X =[-10 -7 -5 -3 -2 -1 0.3 1 2 3 5 7 10 ];
Y =[1.359 1.077 0.714 0.000 -0.571 -1.000...
    0.641 2.000 2.857 3.000 2.857 2.692 2.524];

x=0;

for j = 0.1:0.1:3
Sigma = j;            %Sigma wird im Rahmen von 0.1-3 getestet
fexp=@(x) exp(-1/2*(x)/Sigma^2); %kein x^2, ist in D enthalten
D=[];
sumfx = 0;
sumyfx = 0;

for i = 1:length(X)
    D(i) = X(i)-x;              %Differenz
    D(i) = D(i)*D(i);           %Quadratische Distanz
    sumfx = sumfx + fexp(D(i));
    sumyfx = sumyfx + Y(i)*fexp(D(i));
end

y = sumyfx/sumfx;               %Y(X)
disp(sprintf('Y(X) beträgt bei Sigma = %g: %g',j, y));

end
end
```

Abb. 2-19 Programmiercode

Die Vektoren X und Y beschreiben die gegebenen Punkte. Die Variable x = 0 definiert jene Stelle, an der die Regressionsfunktion ausgewertet wird. Für

die Bestimmung des optimalen Sigmas wird die erste for-Schleife benötigt. Diese startet bei j = 0.1 und wird solange mit einer Schrittweite von 0.1 ausgeführt, bis 3 erreicht wird. Jedes einzelne j stellt hierbei ein zu untersuchendes Sigma dar. Die Variablen sumyfx und sumfx repräsentieren nach vollendeter zweiter for-Schleife den Zähler und Nenner aus Gleichung (2-37). Während jedem Durchlaufschritt wird zunächst die Differenz zwischen X_i und x gebildet, quadriert und anschließend in die Exponentialfunktion eingesetzt. Daraufhin wird zu sumyfx und sumfx, welche jeweils bei Null starten, der aktuelle Summand, sprich die Multiplikation aus Y_i mit der Dichtefunktion bzw. die gemeinsame Dichtefunktion, hinzugefügt. Nachdem jeder einzelne Eintrag im X-Vektor berücksichtigt worden ist, wird die Division durchgeführt, welche das gewünschte Ergebnis liefert.

Die innere for-Schleife wird somit bei jedem gewählten Sigma durchlaufen. Um die beste Regressionsfunktion für die Annäherung im Punkt x = 0 zu finden, wird jenes Sigma ausgewählt, bei welchem der geringste Fehler zwischen Y(x=0) und y(x=0) = 0 auftritt. Abbildung 2-20 verdeutlicht den Zusammenhang zwischen der Abweichung an der Stelle x = 0 und dem Glättungsparameter.

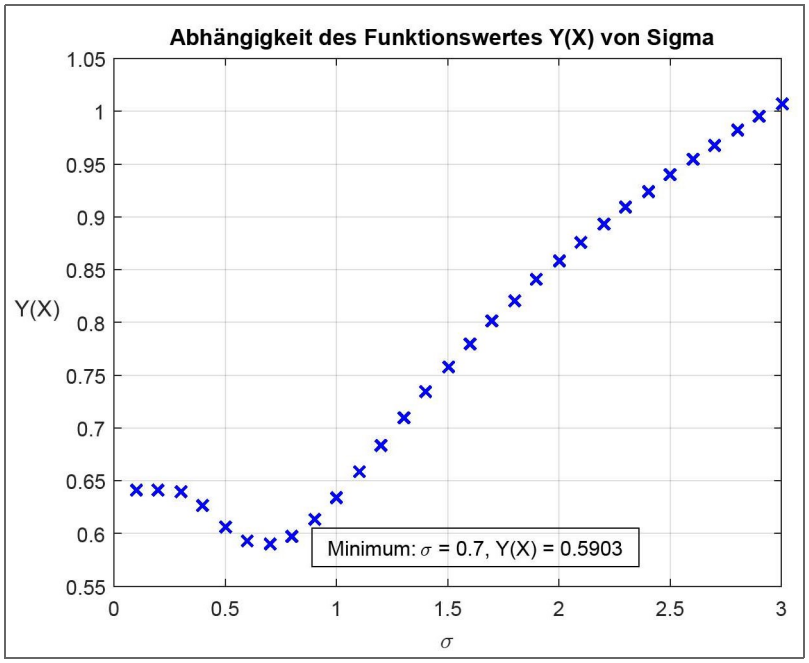


Abb. 2-20 Zusammenhang Glättungsfaktor und Abweichung bei x = 0

Es ist eindeutig zu erkennen, dass das Tal, in dem sich das Minimum mit Sigma = 0.7 befindet, relativ flach verläuft. Somit kann ohne weiterer Schrittgrößenverfeinerung relativ schnell ein passender Glättungsfaktor ausgewählt werden.

Mit einem Sigma von 0.7 beträgt die Abweichung 0.59. Die Ursprungsfunktion weist an der Stelle x = 0 einen Funktionswert von y(x=0) = 0 auf.

Die berechnete Regressionsfunktion Y(x) sowie die Ursprungsfunktion y(x) wird in Abbildung 2-21 gezeigt.

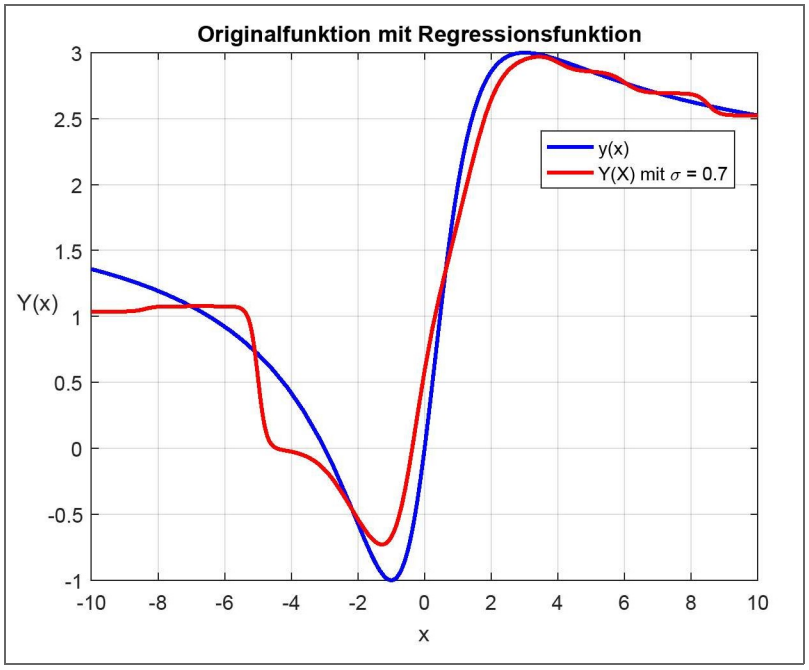


Abb. 2-21 Regressionsfunktion Y(x) und Ausgangsfunktion y(x)

Es ist zu erkennen, dass die Nachbildung im positiven x-Achsenbereich geringere Abweichungen als im negativen Bereich aufzeigt. Um die Schwankungen des roten Regressionsverlaufs zu verbessern müssten weitere Punkte in die Berechnung mitaufgenommen werden.

Für das bessere Verständnis bezüglich des Einflusses des Glättungsfaktors wird Abbildung 2-22 gezeigt. Je größer der Glättungsfaktor gewählt wird, desto glatter wird die Funktion.

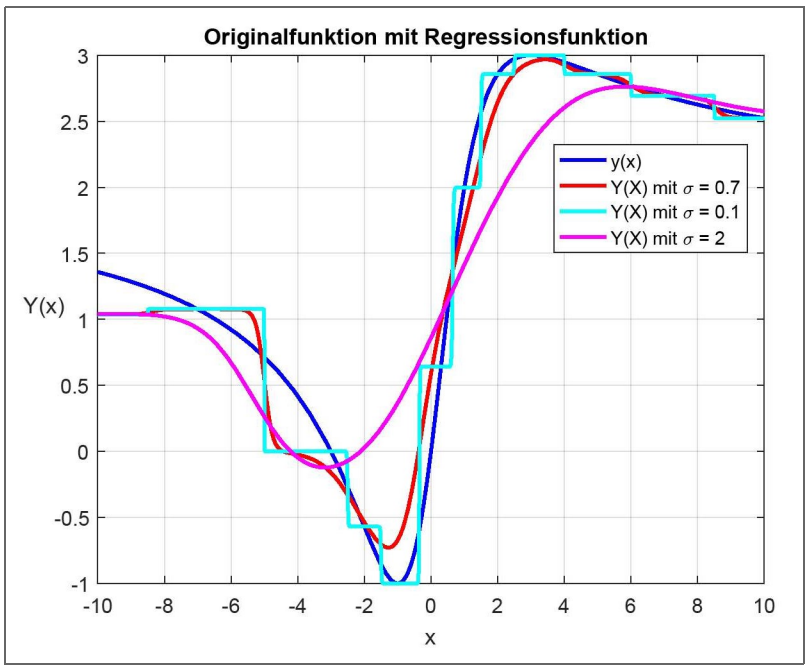


Abb. 2-22 Regressionsfunktionen Y(x) und Ausgangsfunktion y(x)

Nimmt Sigma überaus große Zahlenwerte an wird die Regressionsfunktion im gezeigten Intervall annähernd zu einer horizontalen Gerade, welche an jeder Stelle den Mittelwert von 1,39 aufweist. Aus dieser Darstellung geht ebenfalls hervor, dass bei einem Glättungsfaktor, der gegen Null geht, die Regressionsfunktion einen stufenartigen Verlauf aufweist. Somit wird als Funktionswert der des nächstgelegenen Datensatzes verwendet.

2.3.4 Vorteile und Anwendungsmöglichkeiten

Ein wesentlicher Vorteil von neuronalen Regressionsnetzen ist, dass diese gegen eine optimale Regressionsfläche konvergieren. Es werden lediglich genügend Trainingsdaten benötigt. Grundsätzlich gilt, je mehr bekannte Punkte gegeben sind, desto besser wird die Regressionsfunktion in jenem Bereich, welcher durch die Trainingsdaten abgedeckt ist, abgebildet. Außerhalb kann es zu großen Abweichungen kommen.[100]

Ein weiterer wichtiger Aspekt, welcher für den Einsatz von GRNN spricht, ist, dass diese Netze sehr schnell lernen. Jeder Input eines Trainingsdatensets wird in der zweiten Schichte durch ein Neuron repräsentiert. Somit können rasch neue Trainingsfälle berücksichtigt werden, ohne das gesamte Netz durch iterative Algorithmen, wie beispielsweise durch Backpropagation, anpassen zu müssen. Dazu kommt, dass die Regressionsfunktion unmittelbar nach dem ersten Durchrechnen des Netzes bekannt ist. Dies spielt bei der Echtzeit-Datenanalyse eine wichtige Rolle. Selbst bei einem Trainingsdatensatz ist die Regressionsfläche sogleich definiert.[101]

Mit Hilfe des Glättungsfaktors können Regressionsanalysen selbst bei verrauschten Daten zu passenden Ergebnissen führen. Je größer Sigma gewählt wird, desto weiter wird die Einflussspanne, wodurch die Nachbarwerte eine stärkere Gewichtung bekommen.[102]

Aufgrund der Tatsache, dass das Lernen des Netzes in einem Durchgang geschieht, ist es nicht möglich, ein schlechtes Ergebnis aufgrund eines lokalen Minimums zu erhalten. Jedoch geht hiermit auch eine längere Zeit bei der Berechnung eines neuen Wertes einher. Um Rechenzeit zu sparen ist es allerdings möglich Trainingsdaten, welche in einem ähnlichen Bereich situiert sind, zu einem Cluster zusammenzufassen. Dies kommt bei großen Datensätzen zur Anwendung.[103]

Das Einsatzgebiet von GRNN ist sehr groß. Sobald eine Aufgabenstellung vorliegt, bei der eine Funktion zu finden ist, welche gegebene Daten bestmöglich abbildet, um anschließend Funktionswerte an bestimmten Stellen auszuwerten, kann ein Regressionsnetz eingesetzt werden.

100 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572
101 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572
102 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 572
103 Vgl.: Specht,D.F.:A General Regression Neural Network (1991), Seite 576

3 NeuralTools

In diesem Kapitel wird das Programm NeuralTools[1], ein Produkt der Palisade Corporation, das ein Add-on zu Microsoft Excel[2] darstellt, erklärt. Dieser Excel-Aufsatz ermöglicht das Berechnen verschiedener neuronaler Netztypen, mit Hilfe deren anschließend Prognosen erstellt werden können.

3.1 Aufbau des Programms

Nach dem Starten von NeuralTools wird in der gewohnten Excel-Umgebung gearbeitet. Lediglich ein neuer Reiter „NeuralTools“ kommt in der Menüleiste hinzu. In diesem befinden sich die notwendigen Befehle, um mit Hilfe von neuronalen Netzen komplexe Datenbeziehungen zu analysieren. Die nachfolgende Abbildung 3-1 zeigt die neue Registerkarte. Um eine Aufgabenstellung mit NeuralTools bearbeiten zu können, müssen die jeweilgen Auswahlmöglichkeiten dieses Reiters von links nach rechts durchgegangen werden.

Abb. 3-1 Registerkarte Neural Tools

3.1.1 Datensatzmanager

Zu Beginn wird im Datensatzmanager, welcher in Abbildung 3-2 veranschaulicht wird, das zu untersuchende Datenmaterial definiert. Jede Zeile des Datensatzes repräsentiert einen Trainingsfall. Ob es sich um eine Input- oder Outputvariable handelt, muss separat über den Variablentyp festgelegt werden. Anfangs wird bereits eine Zuteilung von NeuralTools vorgeschlagen, die zu überprüfen ist. Sehr häufig wird die Zuordnung richtig vorgenommen. Folgende Auswahlmöglichkeiten stehen zur Verfügung: Kategorie unabhängig oder abhängig, numerisch unabhängig oder abhängig, unbenutzt und Tag. Die Bezeichnung abhängig deutet immer auf eine Outputvariable hin, weil diese aus verschiedenen Inputparametern berechnet wird. Unabhängig charakterisiert die Inputvariablen. Als numerisch wird ein Wert bezeichnet, der jede beliebige Zahl annehmen kann. Hingegen verkörpert der Parametertyp Kategorie die Zugehörigkeit zu einer bestimmten Klasse. Wird eine Spalte des Datensatzes mit unbenutzt versehen, wird diese in der Berechnung nicht miteinbezogen. Dies ist oftmals nach Sensitivitätsanalysen hilfreich, um jenen Inputparameter, welcher die geringste Auswirkung aufzeigt, aus dem Netz zu entfernen. Der Variablentyp Tag wird verwendet

1 http://www.palisade.com/neuraltools/de/, Datum des Zugriffs: 29.05.2018
2 https://products.office.com/de-at/excel, Datum des Zugriffs: 29.05.2018

um zuzuweisen, welche Fälle für das Trainieren, Testen und Vorhersagen verwendet werden sollen. Dies geschieht über eine eigene zusätzliche Spalte, die beispielsweise Verwendung genannt werden kann.

Nach Festlegung des Datensatzes wird dieser automatisch, falls die Zellformatierung aktiviert ist, als Tabelle mit einer hellblauen Überschrift formatiert und gespeichert. Somit kann dieser auch in anderen Excel-Sheets verwendet werden.

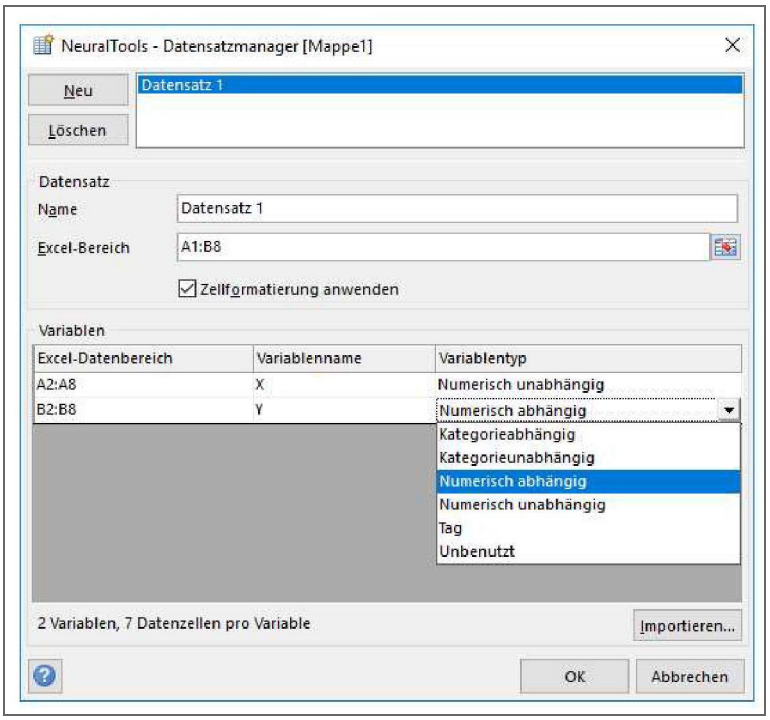


Abb. 3-2 Datensatzmanager

Für Prognose-, Test- oder Trainingsdaten muss nicht jeweils ein eigener Datensatz erstellt werden. Aus Gründen der Übersicht ist dies jedoch bei großen Datensätzen zu empfehlen. Hierbei ist allerdings darauf zu achten, dass die Beschriftungen der Spalten beim Trainieren, Testen und Vorhersagen den selben Namen aufweisen. Ansonsten erscheint eine Fehlermeldung.

Zusammenfassend kann gesagt werden, dass der Datensatzmanager das Anlegen und Bearbeiten, sowie die Bereichsauswahl von diversen Datensätzen ermöglicht.

3.1.2 Trainieren

Nachdem der Datensatz erfolgreich definiert worden ist, kann mit dem Vornehmen der Einstellungen für das Trainieren des Netzes begonnen werden. Hierfür ist auf das Symbol „Trainieren“, welches in Abbildung 3-1 ersichtlich ist, zu klicken. Es erscheint ein neues Fenster, welches in Abbildung 3-3 gezeigt wird. In den drei Registerkarten Trainieren, Netzkonfiguration und Ausführungszeit werden nun die Einstellungen vorgenommen.

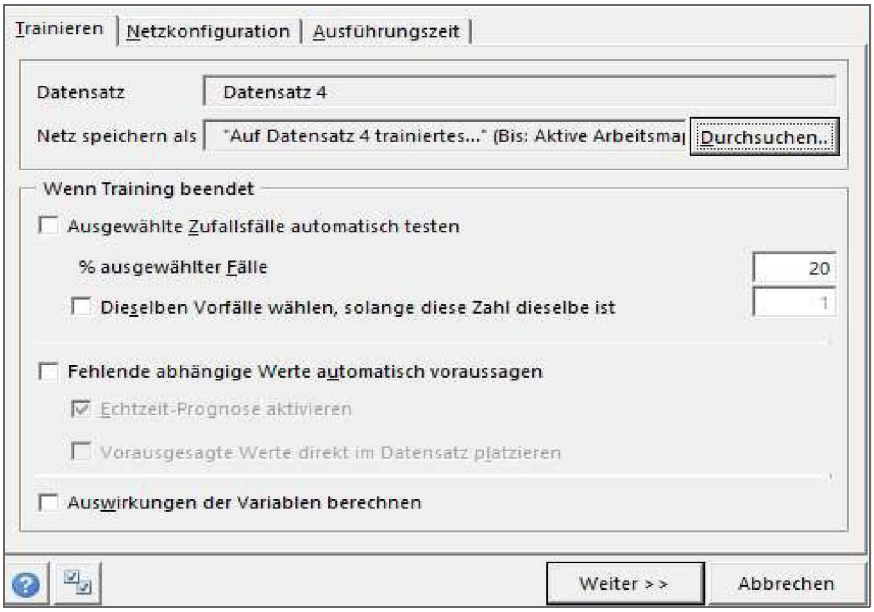


Abb. 3-3 Training – Trainieren

Zunächst werden die gewünschten Eigenschaften in der Registerkarte Trainieren ausgefüllt. Unter dem Namen des Datensatzes und des gespeicherten Netzes können die Prozentzahl der automatisch auszuwählenden Testfälle und die automatische Prognose bei fehlenden Werten eingestellt werden. Beides kann auch separat nach dem Trainieren durchgeführt werden.

Ist „Auswirkungen der Variablen berechnen“ ausgewählt, wird eine Analyse durchgeführt, welche die Auswirkung der einzelnen Parameter auf das Ergebnis untersucht. Daraufhin können, in der Hoffnung das Gesamtergebnis zu verbessern, Variablen mit einem geringen Einfluss aus dem Netz genommen werden.

In der Registerkarte „Netzkonfiguration“ ist der gewünschte Netztyp zu wählen. Grundsätzlich kann zwischen mehrschichtigen Feedforward-Netzen (MLFN) und Wahrscheinlichkeits-/Regressionsnetzen (PNN/GRNN) entschieden werden. Abhängig von der zu lösenden Aufgabenstellung ist eines dieser Netze auszuwählen. Die theoretischen Grundlagen wurden hierfür im vorherigen Kapitel diskutiert.

Sie Software NeuralTools beschreibt die beiden Netztypen auf folgende Art und Weise:

„PNN/GRNN:
Bei einer kategorieabhängigen Variable wird ein neuronales Wahrscheinlichkeitsnetz trainiert. Wenn die abhängige Variable dagegen numerisch ist, wird ein verallgemeinertes neuronales Regressionsnetz trainiert. PN-/GRN-Netze funktionieren in ähnlicher Weise. Jeder Trainingsfall wird durch ein Element (einen „Knoten“) im Netz dargestellt. In einem Fall mit unbekanntem abhängigen Wert wird die Prognose durch Interpolation der Trainingsfälle ausgeführt, und zwar wird dabei bei angrenzenden Fällen mit einem höheren Wahrscheinlichkeitsfaktor gearbeitet. Während des Trainings wird nach optimalen Interpolationsparametern gesucht.“[3]

[3] Palisade Corporation: Software NeuralTools 7.5, Registerkarte Netzwerkkonfiguration

„MLFN:

Ein mehrschichtiges Feedforward-Netzwerk besteht aus einer Eingabeschicht, einer oder zwei ausgeblendeten Schichten, sowie einer Ausgabeschicht. Ein Netz wird konfiguriert, indem die Anzahl der Knoten in beiden ausgeblendeten Schichten angegeben wird.“[4]

Es besteht jedoch auch die Möglichkeit nicht eines der beiden Netze explizit zu wählen, sondern das beste Netz auswählen zu lassen. Hierfür ist der Netztyp „Bestes Netz suchen“ gedacht. Ist genügend Zeit vorhanden[5], wird diese Auswahlmöglichkeit empfohlen, weil sowohl verschiedene MLF-Netze als auch ein PN-/GRN-Netz durchgerechnet werden. Aus den erstellten Netzen wird anschließend jenes ausgewählt, welches die geringste Abweichung im Bezug auf den Testdatensatz vorweist. Die Einstellungsmöglichkeiten für diesen Netztyp werden in Abbildung 3-4 gezeigt.

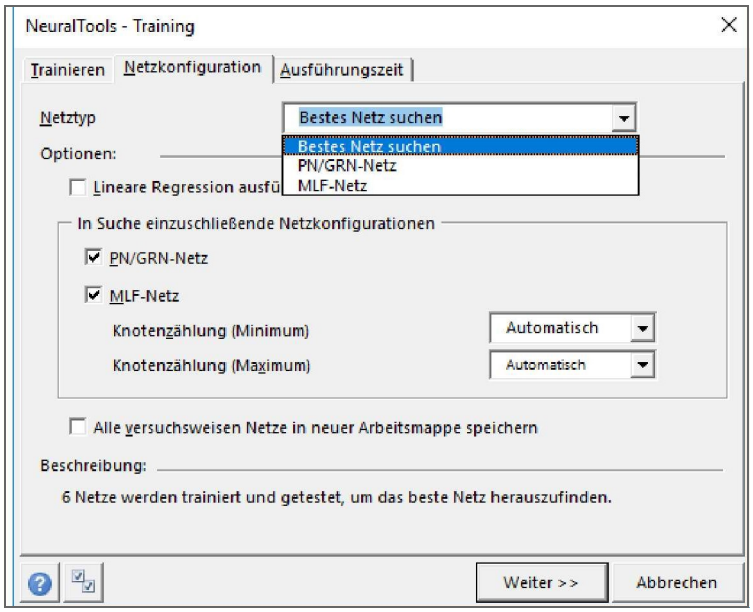


Abb. 3-4 Training – Netzkonfiguration

Schlussendlich müssen noch Einstellungen bezüglich der Ausführungszeit getroffen werden. Hierbei stehen folgende Abbruchkriterien zur Verfügung: Zeit in Stunden, die Anzahl der Versuche und Fortschritt bezüglich der Fehleränderung. Unter Versuch wird bei MLF-Netzen eine einzelne Zuweisung von Gewichtungen und bei PN-/GRN-Netzen eine Zuteilung von Glättungsfaktoren verstanden. Diese drei Möglichkeiten können sowohl kombiniert, als auch negiert werden. Wird keine Begrenzung der Ausführungszeit gewählt, kann das Training händisch im Dialogfeld Trainingsfortschritt abgebrochen werden. Bei PN-/GRN-Netzen ist grundsätzlich keine Beschränkung notwendig, weil diese sehr kurze Trainingsdauern aufweisen. Hingegen ist die Auswahl der Anhaltebedingungen umso wichtiger, wenn mit dem Netztyp „bestes Netz suchen“ gearbeitet wird, damit nicht zuviel Zeit bei einem bestimmten Netz verbraucht wird. Folgende Abbildung verdeutlicht das soeben Beschriebene.[6]

4 Palisade Corporation: Software NeuralTools 7.5, Registerkarte Netzwerkkonfiguration
5 Dauer hängt unter anderem von der Datenmenge, der Anzahl der Knoten und den ausgewählten Abbruchkriterien ab
6 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 51f

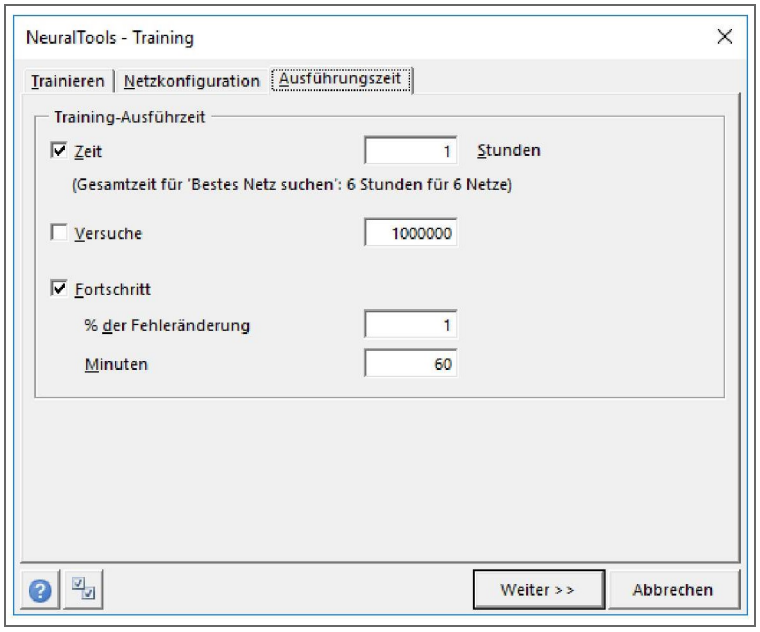


Abb. 3-5 Training – Ausführungszeit

Nachdem jede Registerkarte bearbeitet und die gewünschten Eigenschaften ausgewählt worden sind, kann auf „Weiter“ geklickt werden. Es öffnet sich ein neues Dialogfeld, welches den Titel Trainingsvorschau aufweist. Dieses wird in Abbildung 3-6 gezeigt. Durch den Befehl „Trainieren“ wird das Training mit den ausgewählten Daten begonnen. Jedoch bevor dieses gestartet wird, empfiehlt es sich die diversen Einstellungen in der Übersicht zu kontrollieren und die Fehlermeldungen sowie Warnhinweise sorgfältig zu lesen. Erscheint ein Fehler, kann das Training nicht gestartet werden.

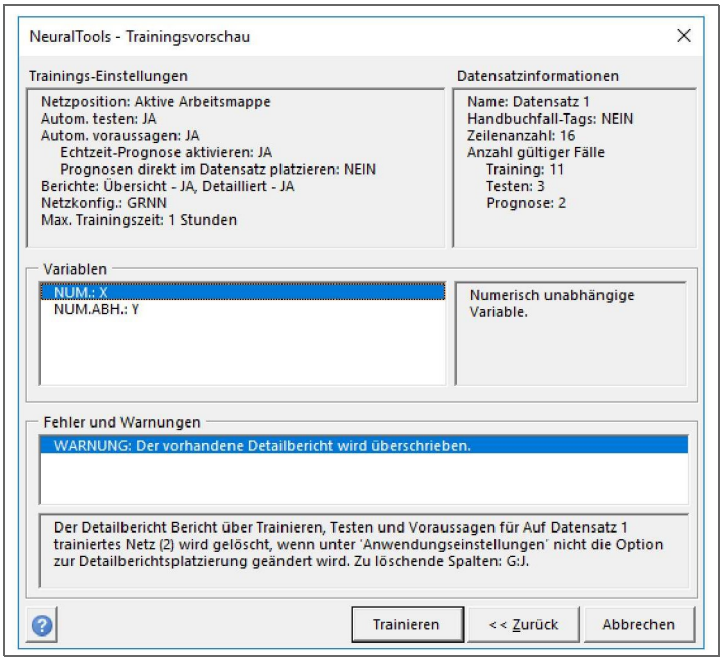


Abb. 3-6 Trainingsvorschau

Nach erfolgreichem Training wird ein Bericht gezeigt. Einstellungen über den gezeigten Inhalt können über den Button, welcher zwei kleine Kästchen aufweist und in der linken unteren Ecke von Abbildung 3-5 situiert ist, vorgenommen werden. Der Bericht besteht aus einer schriftlichen Zusammenfassung der wichtigsten Werte und mehreren Diagrammen, welche Aussagen über Abweichungen des aktuellen Netzes bezüglich der zum Training verwendeten Daten machen. Die nachfolgende Darstellung repräsentiert die schriftliche Zusammenfassung. Schon anhand der Übersicht ist zu sehen, dass der Fehler gegen Null konvergiert. Hierbei gilt es jedoch zu beachten, dass diese Abweichungen nur auf die Trainingsdaten bezogen sind. Erst beim Testen des Netzes kann durch die Anwendung von neuen Fällen eine allgemeine Aussage über das Netz getroffen werden.

NeuralTools: Neuronalnetz-Training
Ausgeführt durch: Ra
Datum: Dienstag, 29. Mai 2018 19:20:12
Datensatz: Datensatz 4
Netz: Auf Datensatz 4 trainiertes Netz

Übersicht	
Netzinformationen	
Name	Auf Datensatz 4 trainiertes Netz
Konfiguration	Numerischer GRNN-Prädiktor
Position	Diese Arbeitsmappe
Kategorieunabhängige Variablen	0
Numerisch unabhängige Variablen	1 (x)
Abhängige Variable	Numerische Var. (y)
Training	
Anzahl von Fällen	51
Trainingszeit	00:00:00
Anzahl der Versuche	55
Anhaltegrund	Autom. angehalten
% schlechter Prognosen (20 % Toleranz)	3,9216%
Mittlerer quadratischer Fehler	0,0000003003
Mittlerer absoluter Fehler	0,0000001308
Std. Abweichung vom ABS. Fehler	0,0000002703
Datensatz	
Name	Datensatz 4
Zeilenanzahl	51
Handbuchfall-Tags	NEIN

Abb. 3-7 Trainingsbericht

Der Anhaltegrund „Autom. angehalten“ bedeutet, dass ein optimales Netz bereits vor erreichen eines Anhaltekriteriums gefunden wurde. Ein Fall wird als schlecht bezeichnet, wenn der aus dem Netz berechnete Wert um eine gewisse Prozentzahl gegenüber dem Original abweicht. Das Ändern dieser Spanne erfolgt über das selbe Dialogfenster, wie beim Einstellen des Berichts.

3.1.3 Testen

Nach erfolgreichem Trainieren muss das neuronale Netz getestet werden. Dies kann sowohl automatisch als auch manuell erfolgen. Bei ersterem ist lediglich automatisch Testen im Dialogfenster der Trainingseinstellungen, welches in Abbildung 3-3 gezeigt wird, auszuwählen und eine Prozentzahl an

Fällen aus dem Datensatz einzugeben, die für das Testen herangezogen werden sollen. Oftmals kommen hierbei 20 % zum Einsatz. Sollen die Testfälle nicht zufällig ausgewählt werden, können diese auch über den Variablentyp Tag festgelegt werden. Bei zweiterem wird das Feld für automatisch Testen leer gelassen. Dadurch wird ein Netz nur trainiert. Anschließend kann durch klicken auf das Testen-Symbol aus Abbildung 3-1 mit einem eigenen Datensatz ein Netz auf seine Akkuratheit überprüft werden. Hierfür ist zunächst der Testdatensatz und das zu überprüfende Netz auszuwählen. Dieses Dialogfenster wird in Abbildung 3-8 gezeigt. Unter Variablenanpassung können Einstellungen bezüglich der Variablen vorgenommen werden. Dies kann automatisch oder benutzerdefiniert geschehen. Um Zeit zu sparen empfiehlt es sich, die Variablen des Testdatensatzes gleich wie im Trainingsdatensatz zu benennen und zu ordnen. Somit kann die Variablenanpassung automatisch erfolgen und muss nicht per Hand eingegeben werden.

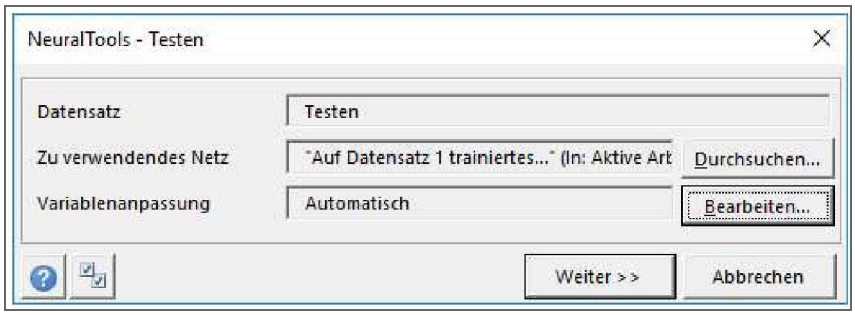


Abb. 3-8 Dialogfenster Testen

Nach betätigen des Weiter-Buttons erscheint das Dialogfenster Testvorschau. Es bietet die Möglichkeit nochmals alle Eingaben zu überprüfen. Abschließend kann mit dem Testen begonnen werden. Abbildung 3-9 veranschaulicht dieses Übersichtsfenster.

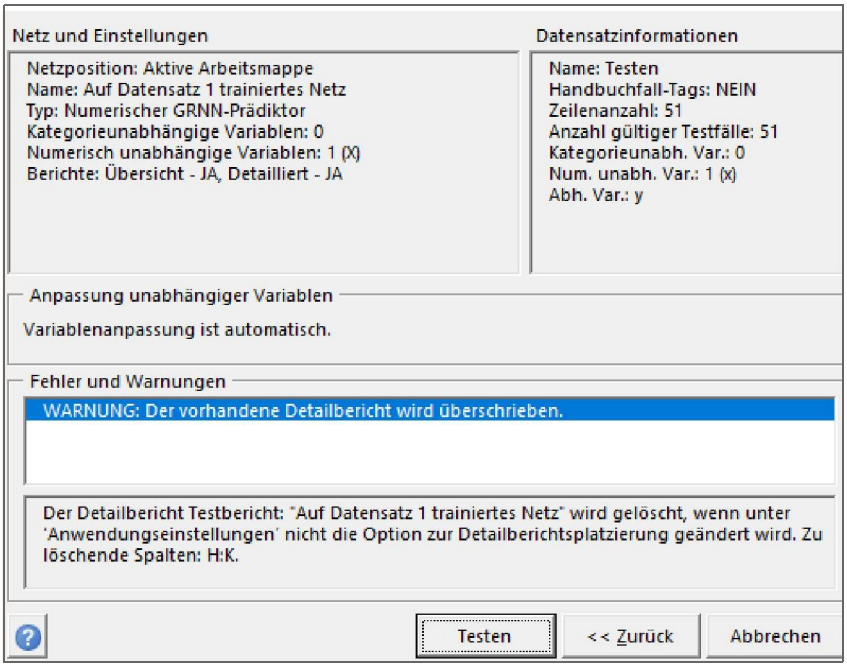


Abb. 3-9 Dialogfenster Testvorschau

Nach Abschluss des Testvorgangs werden wieder Berichte erstellt. Die gewünschten Eigenschaften des Reports können über die Anwendungseinstellungen definiert werden. Diese sind entweder über die Dienstprogramme aus Abbildung 3-1 oder über den Button mit den zwei kleinen Hackerl aus Darstellung 3-8 aufrufbar. Ein Testübersichtsbericht wird in der folgenden Illustrationion gezeigt. Die Übersicht gliedert sich in „Netzinformationen“, wo die wichtigsten Eckpunkte dargestellt werden, in „Testen“, bei dem die Testergebnisse präsentiert werden und in „Datensatz“, wo eine Übersicht über die verwendeten Daten gegeben wird.

NeuralTools: Testübersicht
Ausgeführt durch: Ra
Datum: Donnerstag, 31. Mai 2018 11:34:03
Datensatz: Testen
Netz: Auf Datensatz 1 trainiertes Netz

Übersicht	
Netzinformationen	
Name	Auf Datensatz 1 trainiertes Netz
Konfiguration	Numerischer GRNN-Prädiktor
Position	Diese Arbeitsmappe
Kategorieunabhängige Variablen	0
Numerisch unabhängige Variablen	1 (X)
Abhängige Variable	Numerische Var. (Y)
Testen	
Anzahl von Fällen	51
% schlechter Prognosen (20 % Toleranz)	37,2549%
Mittlerer quadratischer Fehler	0,3102
Mittlerer absoluter Fehler	0,2035
Std. Abweichung vom ABS. Fehler	0,2340
Datensatz	
Name	Testen
Zeilenanzahl	51
Handbuchfall-Tags	NEIN
Variablenanpassung	Automatisch
Verw. kategorieunabh. Variablen	Keine
Verw. numerisch unabh. Variablen	Namen aus dem Training
Abhängige Variable	Numerische Var. (y)

Abb. 3-10 Testübersichtsbericht

Die Anzahl der Fälle beschreibt wie viele Input-Output Zuweisungen im Training verwendet worden sind. Prozent schlechter Prognosen gibt an, wie viele Testfälle bezogen auf die Gesamtanzahl außerhalb des mit gut definierten Bereichs liegen. Der mittlere quadratische Fehler stellt die Quadratwurzel aus den durchschnittlichen quadratischen Abweichungen dar. Der mittlere absolute Fehler errechnet sich aus den durchschnittlichen unter Betrag gesetzten Differenzen von Prognose und aktuellem Wert.[7]

[7] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 63

„Die Standardabweichung ist ein Maß für die Streubreite der Werte eines Merkmals rund um dessen Mittelwert (arithmetisches Mittel). Vereinfacht gesagt, ist die Standardabweichung die durchschnittliche Entfernung aller gemessenen Ausprägungen eines Merkmals vom Durchschnitt.“[8] In diesem Fall repräsentiert der absolute Fehler das Merkmal.

Neben dem soeben besprochenen Report kann auch ein detaillierter Testbericht erstellt werden. Dieser wird direkt neben dem Testdatensatz eingefügt und gibt Aufschluss über die einzelnen Prognosen. In der nachfolgenden Abbildung 3-11 wird ein solcher gezeigt.

Testbericht: "Auf Datensatz 1 trainiertes Netz"

x	y	Verwendetes Tag	Prognose	Gut/Schlecht	Residual
-19,5	1,68	testen	1,08	Schlecht	0,60
-18,5	1,66	testen	1,08	Schlecht	0,58
-17,5	1,64	testen	1,08	Schlecht	0,56
-16,5	1,62	testen	1,08	Schlecht	0,54
-15,5	1,59	testen	1,08	Schlecht	0,51
-14,5	1,56	testen	1,08	Schlecht	0,49
-13,5	1,53	testen	1,08	Schlecht	0,45
-12,5	1,49	testen	1,08	Schlecht	0,41
-11,5	1,45	testen	1,08	Schlecht	0,37
-10,5	1,39	testen	1,08	Schlecht	0,31
-9,5	1,32	testen	1,08	Gut	0,25
-8,5	1,24	testen	1,08	Gut	0,17
-7,5	1,14	testen	1,08	Gut	0,06
-6,5	1,01	testen	1,08	Gut	-0,07
-5,5	0,83	testen	0,71	Gut	0,11
-5,1	0,74	testen	0,71	Gut	0,02
-4,7	0,64	testen	0,71	Gut	-0,08

Abb. 3-11 Ausschnitt aus detailliertem Bericht

Die rote Prognose stellt das aus dem trainierten Netz errechneten Ergebnis dar. Der Residualwert ergibt sich aus der Differenz von Soll-Output (y-Werte) und der Prognose. Wenn die vorhergesagte Zahl um mehr als 20 % abweicht, wird dieser Testfall mit schlecht bewertet.

Unter dem Testübersichtsbericht, der entweder in einer neuen oder in der aktuellen Arbeitsmappe erstellt wird, werden Diagramme angeordnet. Begonnen wird mit dem Histogramm der Residualwerte. Dieses gibt die Häufigkeit der einzelnen Residualwerte wieder und ist in Abbildung 3-12 dargestellt.

Darunter befindet sich das Diagramm „Vorausgesagt gegenüber Aktuell“, welches in der Darstellung 3-13 wiedergegeben wird. Dieses gibt Aufschluss über den Zusammenhang zwischen der Prognose und dem Soll-Output. Je dichter diese Punktwolke ist, desto besser performt das Netz. Im Idealfall liegen alle Punkte auf einer Geraden.

[8] https://de.statista.com/statistik/lexikon/definition/126/standardabweichung/, Datum des Zugriffs: 31.05.2018

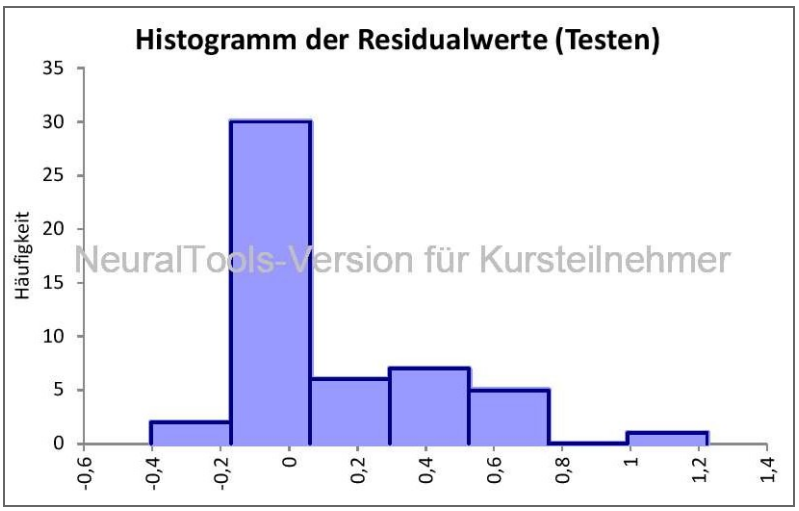


Abb. 3-12 Histogramm der Residualwerte

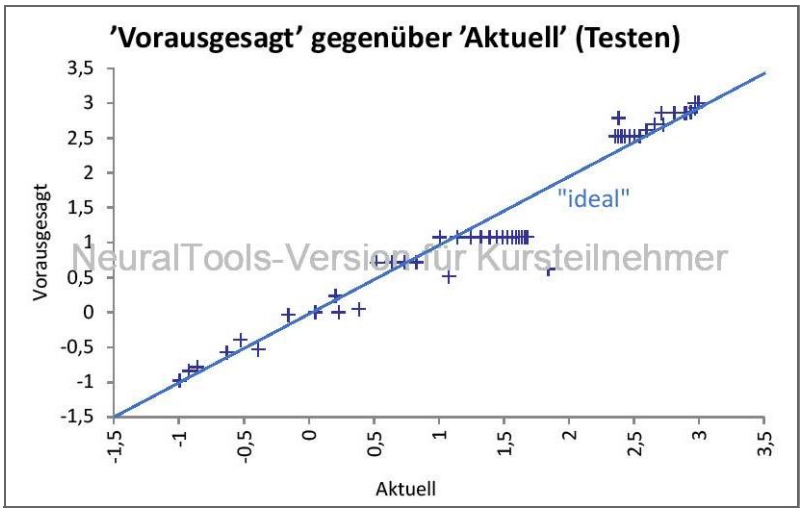


Abb. 3-13 Gegenüberstellung Vorausgesagt zu Aktuell

Schlussendlich werden noch zwei Diagramme, welche den Residualwert abbilden gezeigt. Abbildung 3-14 verdeutlicht den Zusammenhang zwischen Abweichung und aktuellem Wert. Hingegen verkörpert Abbildung 3-15 die Abhängigkeit zwischen Residualwert und der Prognose. Im besten Fall ordnen sich die Testfälle in einer horizontalen Linie durch die Null-Differenz an. Diese Diagramme sind wichtig um zu erkennen, ich welchen Bereichen die Regressionsfunktion die Daten gut abbildet und in welchen Abschnitten die Abweichungen zu groß sind. Das Netz kann daraufhin mit neuen Trainingsdaten, die aus dem schlechten Bereich stammen, verbessert werden.

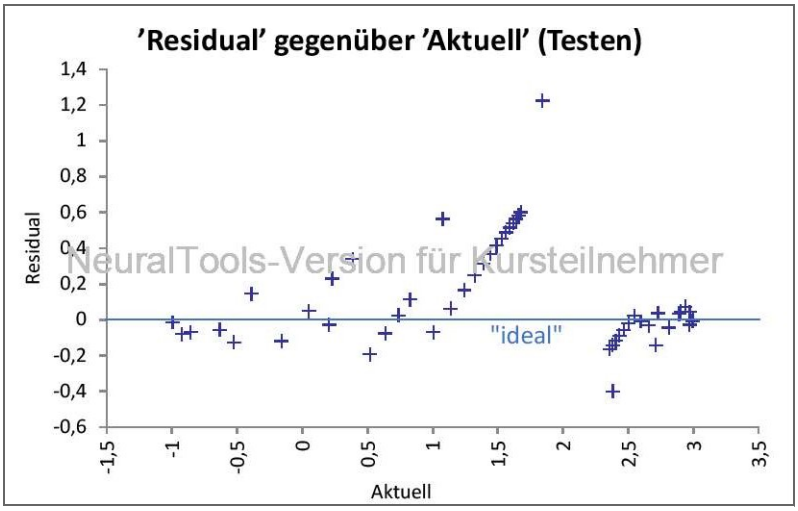


Abb. 3-14 Gegenüberstellung Residual zu Aktuell

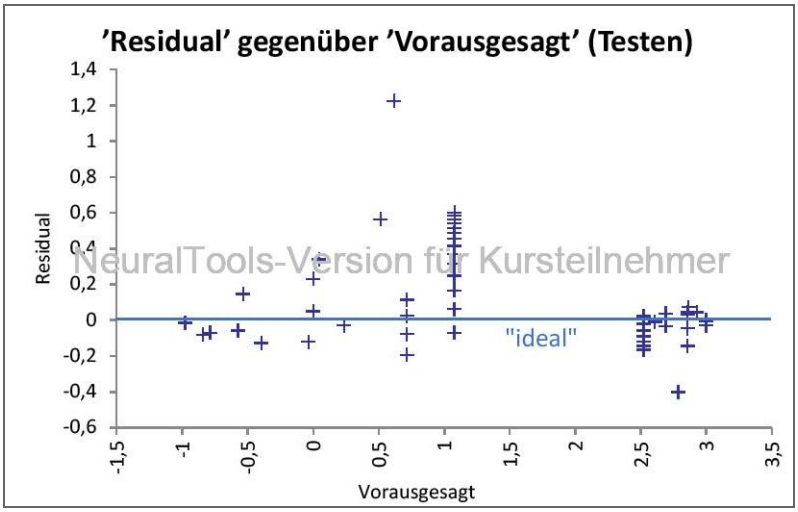


Abb. 3-15 Gegenüberstellung Residual zu Vorausgesagt

3.1.4 Voraussagen

Das Erstellen einer Prognose kann selbstständig nach dem Testen erfolgen, falls die automatische Vorhersage in Abbildung 3-3 ausgewählt ist oder kann manuell durchgeführt werden. Hierzu ist auf den Button „Voraussagen“, welcher in Abbildung 3-1 gezeigt wird, zu klicken.

Nach der Auswahl des Datensatzes sind daraufhin die gewünschten Einstellungen für das Vorhersagen in dem Prognose-Dialogfenster, welches in Darstellung 3-16 verdeutlicht wird, vorzunehmen. Wird ein eigener Prognose-Datensatz bereitgestellt, empfiehlt es sich die errechneten Werte gleich in diesem einzufügen. Hierbei gilt jedoch zu beachten, dass etwaig vorhandene Werte überschrieben werden.

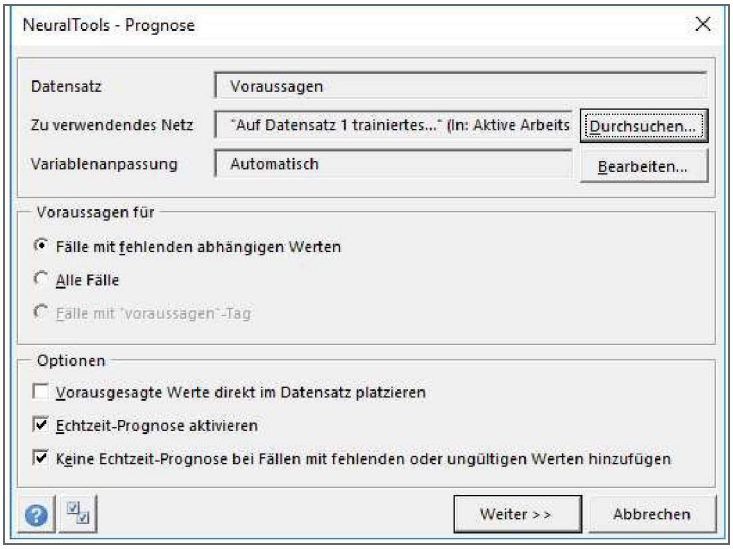


Abb. 3-16 Dialogfenster Prognose

Die Variablenanpassung kann automatisch oder benutzerdefiniert vorgenommen werden. Sie ermöglicht das Abgleichen zwischen den Variablen aus dem Trainingsdatensatz und aus dem Prognosedatensatz. Dies wird benötigt, falls die Variablen anders benannt worden sind.

Die Echtzeit-Prognose stellt eine leistungsstarke Funktion dar, welche das automatisierte Prognostizieren ermöglicht. Hierfür wird eine Funktion in der Zelle hinterlegt, welche sofort ein neues Ergebnis parat hat, falls sich der Datensatz ändert. Dies spart Zeit, weil keine weiteren Prognosen ausgeführt werden müssen. Für Datensätze, welche sich nicht ändern, sollte die Echtzeit-Prognose deaktiviert werden. Es gilt jedoch auch zu erwähnen, dass diese Funktion nur in NeuralTools Industrial zur Verfügung steht.[9]

Nach Klicken auf „Weiter“ öffnet sich ein Übersichtsfenster, welches die wichtigsten Eckpunkte sowie Warnhinweise und Fehlermeldungen beinhaltet. Diese Vorschau wird in Abbildung 3-17 verdeutlicht. Abschließend kann mit der Prognose durch Klicken auf Voraussagen begonnen werden.

Die Inhaltsangaben für den Bericht werden in den Anwendungseinstellungen vorgenommen. In der Rubrik „Spalten in Detailberichten“ können die gewünschten Spalten ausgewählt werden. Weiters werden die Prognosewerte pink eingefärbt.

Zusammenfassend kann gesagt werde, dass der Umgang mit NeuralTools sehr schnell erlernbar ist. Anfangs empfiehlt es sich die gewünschten Berichtseinstellungen einmal vorzunehmen, um anschließend die Konzentration auf das Trainieren, Testen und Prognostizieren legen zu können.

[9] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 67f

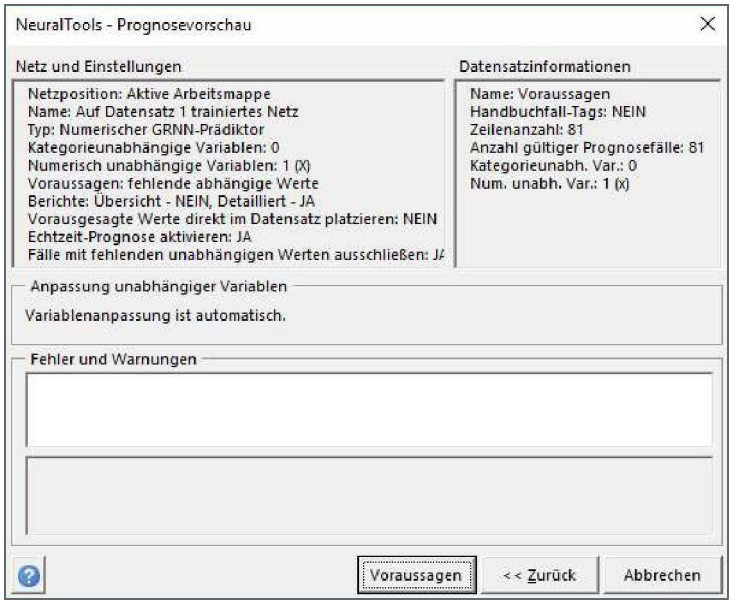


Abb. 3-17 Dialogfenster Prognose

Neben dem Erstellen von neuronalen Netzen bietet NeuralTools auch hilfreiche Dienstprogramme an. Beispielsweise können über den Neuronalnetzmanager trainierte Netze kopiert, verschoben oder gelöscht werden.

Ein weiteres praktisches Hilfsprogramm wirkt unterstützend bei dem Ersetzen von fehlenden Daten durch künstliche. Mit diesem Werkzeug ist es einfacher große Datenmengen zu verarbeiten. Erscheint im Dialogfeld Trainings-Vorschau ein Warnhinweis aufgrund unzureichender Datengrundlage, können mit dem Dienstprogramm zum Ersetzen fehlender Daten die Problemstellen leicht detektiert und behoben werden. Nachdem das Dialogfenster dieses Hilfsmittels geöffnet worden ist, wird unter „Art der Ersatzwerte“ festgelegt, was an Stelle der unerwünschten Daten platziert werden soll. Abhängig vom Variablen Typ gibt es hierfür verschiedene Möglichkeiten. Bei Kategorievariablen kann aus häufigster oder seltenster Kategorie, benachbarter Kategorie, Zufallskategorie oder spezieller Kategorie ausgewählt werden. Hingegen bietet das Dienstprogramm bei numerischen Variablen den durchschnittlichen Variablenwert, den mittleren Variablenwert, die Interpolation aus angrenzenden Werten oder einen Zufallswert an. Bei beiden Variablen besteht unter anderem auch die Möglichkeit Zellen zu löschen. Es sei jedoch darauf hingewiesen, dass es oftmals besser ist, Zellen leer zu lassen und diese dadurch in der Berechnung nicht miteinzubeziehen, als irgendwelche Werte einzusetzen.[10]

Ein weiteres Hilfsmittel stellt der Befehl Testempfindlichkeit dar. Dieser gibt Aufschluss darüber, ob die Testergebnisse eines trainierten Netzes auch bei zufälligen Auswahlen von Testfällen stabil bleiben. Dies ist besonders bei kleinen Datensätzen mit zufälliger Testvariablenbestimmung wichtig. Die nachfolgende Abbildung 3-18 zeigt das Dialogfeld Testempfindlichkeit.

[10] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 79

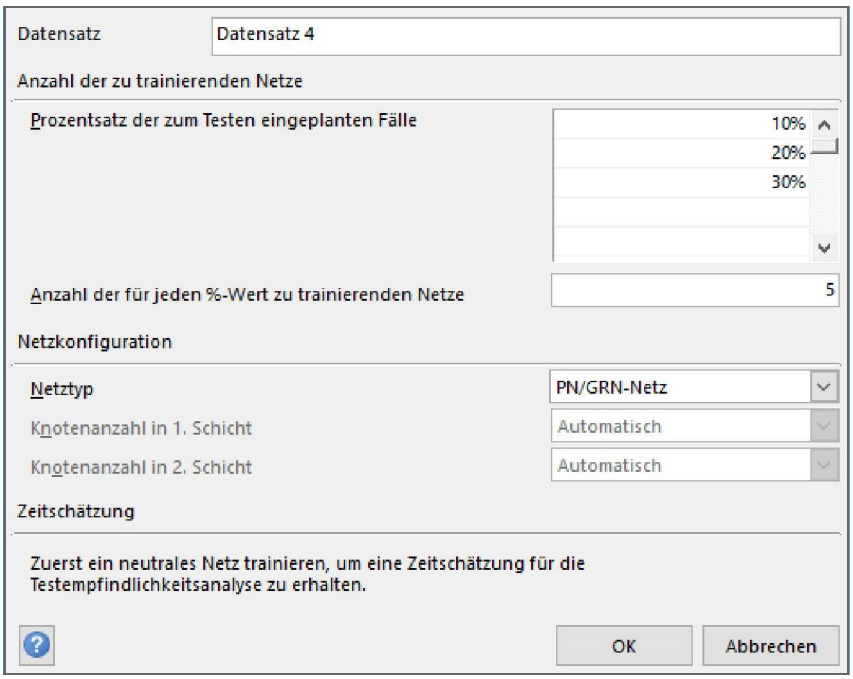


Abb. 3-18 Dialogfenster Testempfindlichkeit

Zunächst muss der Prozentsatz zum Testen der eingeplanten Fälle eingegeben werden. Anschließend ist die Anzahl der zu trainierenden Netze einzugeben. In diesem Beispiel werden 3 Testungen mit jeweils 5 verschiedenen Netzen durchgeführt. Bei der ersten werden 10 % der Daten zum Testen und 90 % zum Trainieren verwendet. Bei den anderen beiden beträgt das Verhältnis 2:8 und 3:7. Das Ergebnis dieser Analyse wird in Darstellung verdeutlicht. Es ist zu erkennen, dass der Bereich des mittleren quadratischen Fehlers, der anhand 5 verschiedener Netze berechnet worden ist, bei 10 % am kleinsten ist. Somit ist in diesem Fall das Training mit 90 % des Datensatzes zu bevorzugen.

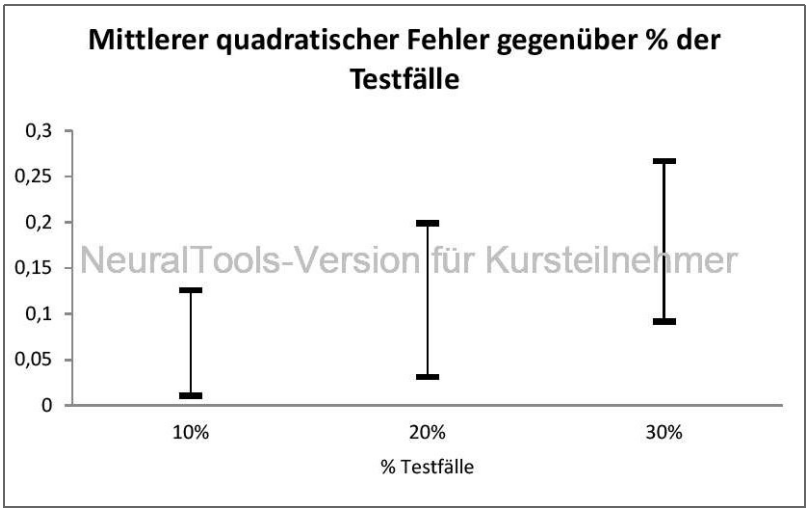


Abb. 3-19 Ergebnis Testempfindlichkeit

3.2 Netzauswahl

Neuronale Netze stellen heutzutage eine Alternative zu altbewährten statistischen Methoden dar. Mit unterschiedlichen Netzen ist es möglich sowohl Funktionsannäherungen als auch Klassifizierungen durchzuführen. Hierbei ist ein wesentlicher Vorteil gegenüber herkömmlichen linearen Techniken, dass mit relativ geringem Aufwand komplizierte Modelle abgebildet werden können.[11]

Mit NeuralTools können zwei verschiedene Netztypen berechnet werden. Auf der einen Seite sind das mehrschichtige Feedforward-Netze (MLFN) und auf der anderen Seite sind es verallgemeinerte neuronale Regressionsnetze (GRNN) und neuronale Wahrscheinlichkeitsnetze (PNN). Im folgenden Abschnitt wird auf die Umsetzung dieser Modellierungsmöglichkeiten im Programm NeuralTools eingegangen. Die zugehörigen theoretischen Grundlagen wurden bereits im vorherigen Kapitel erläutert.

3.2.1 Mehrschichtige Feedforward-Netze

Dieser Netztyp ist in der Lage sowohl komplexe Funktionen zu approximieren, als auch Zuordnungen durchzuführen. In den Trainingseinstellungen ist die Möglichkeit gegeben, die Anzahl der Neuronen je Schicht festzulegen. Es können höchstens 100 Units je Layer und maximal 2 Schichten ausgewählt werden. Der Benutzer kann somit durch Änderungen der Topologie das Verhalten des Netzes stark beeinflussen. Grundsätzlich gilt, je komplexer die Aufgabenstellung, desto mehr Neuronen werden benötigt. Weiters ist zu beachten, dass beim Voraussagen von Zahlenwerten Einzelausgabenetze zuverlässiger sind als Netze mit mehreren Ausgaben.[12] Müssen mehrere Werte prognostiziert werden ist es am besten, einen Datensatz mit allen Variablen zu definieren und anschließend ein Netz separat für jeden vorauszusagenden Wert zu erstellen. Dies gelingt am einfachsten, indem die übrigen gefragten Variablen im Datensatz zu „unbenutzt“ gesetzt werden.

Als Aktivierungsfunktion wird in NeuralTools die hyperbolische Tangentenfunktion, welche in Abbildung 3-20 dargestellt ist, eingesetzt. Diese wird auf die gewichtete Summe der Outputs der Neuronen aus der vorherigen Schicht angewandt.[13]

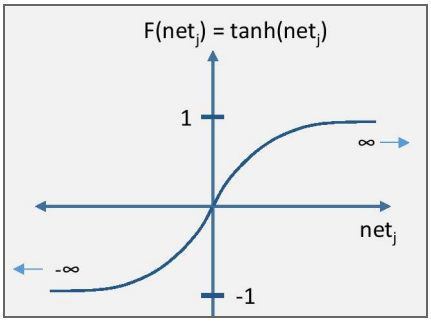


Abb. 3-20 Schematische Darstellung des Tangens-Hyperbolicus

[11] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 87
[12] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 86ff
[13] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 89

Das Ausgabeneuron weist jedoch nur die Einheit als Aktivierungsfunktion auf. Dies bedeutet es gibt die gewichtete Summe der Eingaben zurück. Somit kann auf ein zusätzliches Transformieren der Outputs, welches darauf beruht, dass der Ausgabebereich beim Tangenshyperbolicus nur zwischen -1 und 1 liegt, verzichtet werden.[14]

Trainiert wird ein MLFN in dem Gewichtungen, gesucht werden, die es dem Netz ermöglichen generell die richtige Antwort zu geben. Dieser Prozess hängt von den Trainingsdaten und vom Trainingsalgorithmus ab. Der Algorithmus wählt zunächst verschiedene Gewichtungen aus und wertet mit diesen die Trainingsfälle aus. Die errechneten Outputs werden von den Trainingsdaten abweichen. Auf Basis dieser Differenzen werden neue Gewichtungen ausgewählt. Das Ziel dieser Berechnungen ist somit durch Ändern der Gewichtungen den Fehlermesswert des gesamten Trainingssatzes, welcher sich aus den mittleren quadratischen Abweichungen der einzelnen Trainingsfälle errechnet, zu minimieren. Um dieses Optimierungsproblem zu lösen, kommt in NeuralTools das konjugierte Gradienten-Verfahren zur Anwendung.[15]

Auf der Suche nach dem Minimum folgt dieser Algorithmus dem Neigungsgrad der Fehlerfunktion. Zu Beginn wird der Gradient am gegebenen Startpunkt berechnet. Dieser gibt die erste Richtung vor. Anschließend gilt es herauszufinden, wie weit diesem vorgeschlagenen Weg gefolgt werden soll. Solange der Fehlerfunktionswert kleiner wird, ist die Richtung beizubehalten. Ein neuer Startpunkt ist gefunden, sobald dieser Wert wieder zunimmt. Als nächsten Schritt wird eine Richtung gewählt, die diesmal durch den konjugierten Gradienten vorgegeben wird. Die Besonderheit dieses Gradienten ist, dass die Anteile, welche parallel zu den vorherigen steilsten Anstiegen sind, zu Null werden. Darauf hin ist abermals die Distanz, welche auf dem neuen Weg zurück gelegt werden soll, zu bestimmen. Nachdem ein neuer Startpunkt gefunden worden ist, beginnt der Lösungszyklus durch Bilden eines weiteren konjugierten Gradienten von vorne. Der soeben besprochene Vorgang wird solange wiederholt, bis ein Minimum erreicht ist. Diese Vorgehensweise zeigten Johansson et al.[16] im Jahre 1992 anhand einer quadratischen Funktion. Aus diesem Grund wird die komplexe Fehlerfunktion in den zu untersuchenden Punkten quadratisch approximiert. Hierbei kommt ein mehrdimensionales Taylerpolynom, welches in folgende Gleichung (3-1) gezeigt wird, zum Einsatz.[17]

$$E(w) = E(\hat{w}) + (w - \hat{w})^T b + \frac{1}{2}(w - \hat{w})^T H(w - \hat{w}) \qquad (3\text{-}1)$$

Der zweite Summand repräsentiert den linearen Anteil und der dritte den quadratischen. Die Variable b verkörpert den Gradienten an der Stelle $\hat{w}$ und H stellt die Hesse-Matrix, welche alle zweiten partiellen Ableitungen beinhaltet, dar.[18]

Zusätzlich wird in NeuralTools Simulated Annealing eingesetzt, damit ein lokales Extremum wieder verlassen werden kann.[19] Dieses Verfahren beruht auf der Idee eines Abkühlvorganges und stammt aus dem Bereich der Metall-

14 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 89
15 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 89ff
16 Johansson, et al.: Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method
17 Vgl.: Bishop, C. M.: Neural Networks for Pattern Recognition, Seite 276ff
18 Vgl.: Bishop, C. M.: Neural Networks for Pattern Recognition, Seite 257
19 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 91

bearbeitung. Es imitiert das Tempern, einen metallurgischen Prozess, bei dem Atome durch Zuführen von Energie zunächst in ein höheres Energieniveau aufsteigen um anschließend durch langsames Abkühlen den optimalen Gitterplatz, sprich deren Energieminimum, finden.[20]

Umgesetzt wird dieses Prinzip auf folgende Art und Weise: Die neue Lösung, die besser als die vorherige ist, wird sofort in der nächsten Iteration verwendet. Eine verschlechternde Lösungsmöglichkeit wird mit einer bestimmten Wahrscheinlichkeit angenommen. Während dem Fortschreiten des Verfahrens wird diese Wahrscheinlichkeit allerdings sukzessive reduziert (Abkühlvorgang) bis am Ende nur noch Verbesserungen erlaubt sind. Es ist mathematisch bewiesen, dass dieses Verfahren bei einer unendlichen Abkühlzeit das globale Minimum erreicht.[21]

Bei endlicher Abkühlzeit ist dies jedoch nicht der Fall. Aus diesem Grund startet der Trainings-Algorithmus von NeuralTools die Berechnung des minimalen Fehlermesswertes mit unterschiedlichen Startfaktoren oftmals von vorne, in der Hoffnung das globale Minimum zu finden.[22]

3.2.2 Verallgemeinerte neuronale Regressionsnetze (GRNN) und Wahrscheinlichkeitsnetze (PNN)

Die Grundidee beider Netze ist sehr ähnlich und wurde bereits im vorherigen Kapitel erläutert. Der wesentliche Unterschied in der Verwendung ist, dass GRNN bei numerischen Prognosen und PNN bei Kategorieprognosen oder Klassifizierungen eingesetzt werden.[23]

Regressionsnetze berechnen bei Prognosen die Distanz zwischen vorauszusagendem Input und den Trainingsbeispielen. Abhängig von diesen Differenzen werden anschließend die bekannten Resultate gewichtet. Hierbei spielen Glättungsfaktoren eine wesentliche Rolle, weil diese den Verlauf der Regressionsfunktion beeinflussen. Je größer dieser Wert ist, desto weiter reicht der Einfluss der bekannten Wertepaare. Trainiert wird dieses Netz indem der Fehlermesswert, abermals der mittlere quadratische Fehler, mittels konjugiertem Gradientenverfahren minimiert wird. Beim Suchen des besten Glättungsparameters wird der zu testende Trainingsfall in der Musterschicht nicht eingebaut, damit keine Null Distanz berechnet wird. Keine Abweichung würde zu einem Bedeutungsverlust der anderen Neuronen führen.[24]

Bei neuronalen Wahrscheinlichkeitsnetzen wird ebenfalls der optimale Glättungsfaktor gesucht. Dieser wird benötigt um die Wahrscheinlichkeitsdichtefunktionen der einzelnen Klassen bestmöglich zu approximieren. Zu diesem Zweck kommt ebenfalls die Optimierungsmethode konjugiertes Gradientenverfahren zum Einsatz.[25]

Wesetliche Vorteile dieser zwei Netztypen sind, dass sie schnell trainiert werden können, keine speziellen Topologieangaben benötigen und bei technischen Problemen oftmals gute Lösungen anbieten.[26]

---

20 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 101f
21 Vgl.: Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, Seite 101f
22 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 91
23 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 94
24 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 96
25 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 99
26 Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 100

3.2.3 Eingabetransformation

Damit alle Variablen, die in NeuralTools verwendet werden, die gleiche Bandbreite aufweisen, müssen die verschiedenen Parameter zu Beginn skaliert werden. Einerseits ist dieser Prozess erforderlich, um ein Ausgleichen der einzelnen Auswirkungen der Variablen zu erzielen.[27] Andererseits wird bei der Bestimmung der Wahrscheinlichkeitsdichtefunktionen ein Kerndichteschätzer verwendet, dessen Bandbreite in allen Dimensionen gleich ist. Aus diesem Grund dürfen die Wertebereiche der Variablen nicht zu stark voneinander Abweichen.

Bei der Eingabetransformation stellen der aus dem Datensatz berechnete Mittelwert und die entsprechende Standardabweichung die Skalierungsparameter dar. Um einen ähnlichen Einflussbereich zu erhalten, wird von jeder Zahl der Mittelwert abgezogen und anschließend dieses Ergebnis durch die Standardabweichung dividiert. Diese Skalierung wird anfangs beim Trainieren, beim Testen und beim Vorhersagen durchgeführt.[28]

Aufgrund der Tatsache, dass symbolische Kategoriedaten nicht in einem neuronalen Netz verwendet werden können, müssen diese in Zahlen konvertiert werden. Dies geschieht völlig automatisch im Hintergrund. Nach der Eingabe einer kategorieunabhängigen Variablen wird dieses in verschiedene numerische Inputs umgewandelt. Hierbei wird für jede Kategorie eine eigene Spalte angelegt. Fällt die Eingabe in diese Kategorie wird der Wert in der dazugehörigen Spalte zu eins. Ansonsten ist Null einzusetzen. Die nachfolgenden Tabellen 3-1 und 3-2 verdeutlichen diese Konvertierungsmethode.[29]

Lfd. Nr.	Alter	Geschlecht	Größe
0	A	B	D
1	40	m	1,84 m
2	68	w	168,00 m
3	26	w	175,00 m
	w = weiblich, m = männlich		

Tab. 3-1 Datensatz mit kategorieunabhängigen Variablen

Lfd. Nr.	Alter	w	m	Größe
0	A	B	B	D
1	40	0	1	1,84 m
2	68	1	0	168,00 m
3	26	1	0	175,00 m
	w = weiblich, m = männlich			

Tab. 3-2 Umwandlung numerische Netzeinagebe

[27] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 102
[28] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 102
[29] Vgl.: Palisade Corporation: Benutzerhandbuch, NeuralTools, Seite 102

4 Anwendungsbeispiel

In diesem Kapitel wird eine Kategorisierung mit Hilfe von neuronalen Netzen durchgeführt. Den Netzen soll das Einteilen in drei verschiedene Klassen gelernt werden. Weiters wird gezeigt, welchen Einfluss die Anordnung und die Auswahl der Trainingsdaten hat.

4.1 Aufgabenstellung

Anhand von drei Trainingsdatensätzen zu je 80 Samples (x-Koordinate, y-Koordinate und Klassenzugehörigkeit) gilt es drei Netze zu konstruieren, welche weitere Datenpunkte bestmöglich in die Klassen A, B und C einteilen. Anschließend sind diese zu vergleichen.

Die verschiedenen Kategorien stellen jeweils eine Teilfläche in einem 15 x 15 großen Raster dar und werden in Abbildung 4-1 gezeigt. Die Klasse A beinhaltet jene Punkte (X/Y), die innerhalb oder auf dem orangen Quadrat liegen, sprich deren X- und Y-Koordinaten kleiner gleich sechs ist. Um der Klasse B zugeteilt zu werden muss ein Punkt im blau begrenzten Bereich zu liegen kommen. Dies ist der Fall, wenn die X-Koordinate einen Wert größer gleich 7 aufweist und sich die Y-Koordinate auf oder unter der Funktion, welche in Gleichung (4-1) dargestellt wird, befindet. Den Rest der Fläche nimmt die Klasse C ein.

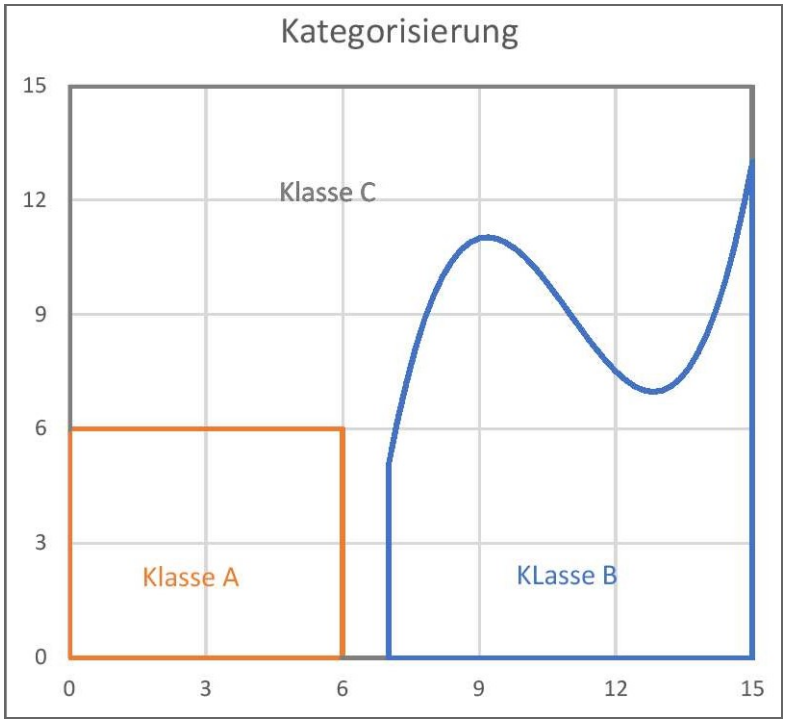


Abb. 4-1 Darstellung der Klassen A, B und C

$$y(x) = \frac{2x^2 + 6x}{x^2 + 3} \qquad (4\text{-}1)$$

Die drei Trainingsdatensätze unterscheiden sich in der Anordnung der bereits klassifizierten, somit bekannten, Punkte. Im ersten Datensatz werden die 80 Trainingsbeispiele zufällig angeordnet. Im zweiten sind die Koordinaten ebenfalls zufällig, jedoch weisen sie ganzzahlige Eigenschaften auf. Somit wird die Struktur innerhalb des Datensatzes erhöht. Im dritten Trainingsdatensatz wird ein Raster von 9 x 9 Punkten vorgegeben. Um 80 Beispiele zu erhalten, wird der Punkt P(0/15) eliminiert, weil dieser aufgrund seiner Position den geringsten Einfluss auf die Klassenauswahl hat.

Für jeden Datensatz werden neuronale Netze mit einem Hidden-Layer, der aus zwei, drei, vier, fünf, sechs, sieben oder acht Neuronen besteht berechnet. Weiters werden Netztypen mit zwei Hidden-Layern, die jeweils drei, sechs oder acht Neuronen innehaben, erstellt.

In den folgenden Abschnitten werden die für jeden Datensatz mit Hilfe der Software Neural Tools 7.5 berechneten Netze vorgestellt. Um die verschiedenen Netztypen bewerten zu können, wird ein weiterer Datensatz mit 40 zufällig gewählten und bereits klassifizierten Punkten zu Hilfe genommen. Abbildung 4-2 stellt die möglichen Klassen und die Testpunkte, welche zusätzlich in Tabelle 4-1 gezeigt werden, dar.

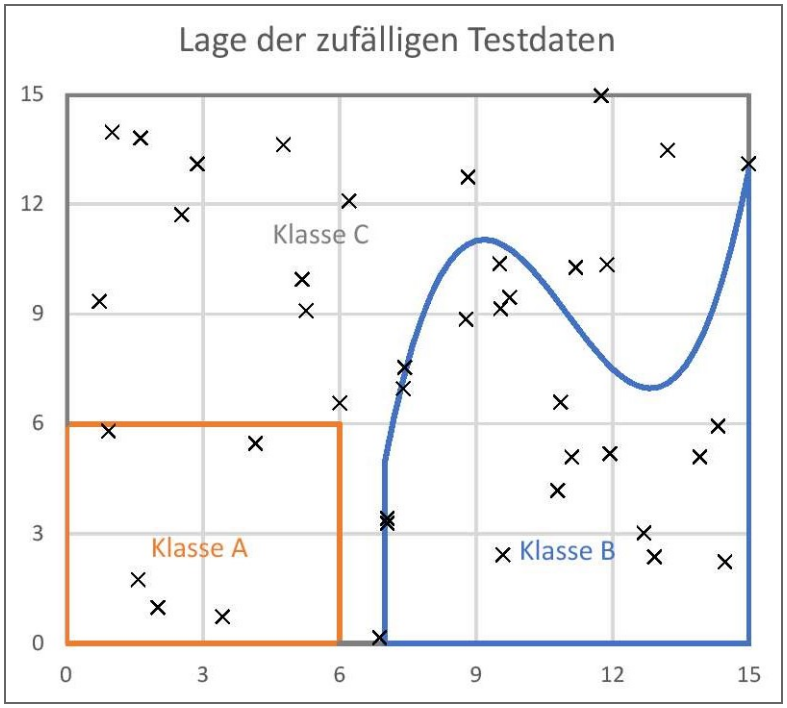


Abb. 4-2 Darstellung der 40 zufälligen Testdaten

X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse
8,78	8,86	B	14,47	2,23	B
12,69	3,02	B	10,79	4,18	B
3,43	0,74	A	13,92	5,1	B
14,99	13,09	C	7,44	7,54	C
13,21	13,47	C	4,77	13,62	C
2,02	0,99	A	14,32	5,94	B
0,93	5,8	A	1,58	1,75	A
8,82	12,74	C	7,05	3,29	B
9,54	9,15	B	6,88	0,16	C
11,88	10,35	C	1,01	13,96	C
2,88	13,09	C	9,59	2,43	B
1,64	13,8	C	5,18	9,95	C
7,05	3,42	B	7,4	6,96	B
0,73	9,34	C	6,21	12,09	C
6,01	6,57	C	11,75	14,96	C
4,16	5,47	A	2,54	11,71	C
10,86	6,59	B	9,52	10,37	B
11,94	5,19	B	11,19	10,27	C
5,27	9,09	C	9,74	9,45	B
11,1	5,09	B	12,92	2,37	B

Tab. 4-1 Liste der zufällig gewählten Testdaten (40 Samples)

4.2 Zufällig gewählte Trainingsdaten

Abbildung 4-3 zeigt die Positionierung der zufällig gewählten Trainingsdaten und die drei Kategorien A, B und C.

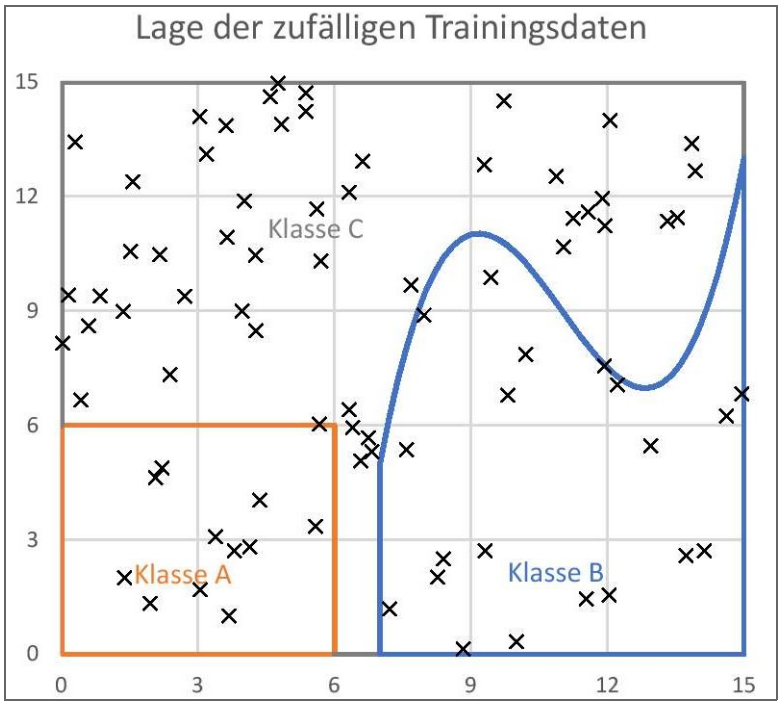


Abb. 4-3 Darstellung der 80 zufällig gewählten Trainingsdaten

Weiters werden diese zufälligen Trainingsdaten in der Tabelle 4-2 aufgelistet.

X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse
3,97	9	C	10	0,33	B	11,53	1,44	B
3,38	3,08	A	3,8	2,72	A	3,64	10,93	C
11,93	7,56	B	13,53	11,44	C	8,27	2,02	B
4,14	2,81	A	11,88	11,94	C	5,37	14,22	C
9,81	6,79	B	5,59	3,35	A	7,58	5,35	B
4,59	14,61	C	1,57	12,38	C	9,29	12,83	C
1,52	10,55	C	7,97	8,89	B	11,04	10,67	C
1,95	1,33	A	0,03	8,15	C	13,31	11,34	C
6,33	6,41	C	5,37	14,71	C	14,95	6,82	B
3,68	1	A	6,58	5,06	C	12,95	5,46	B
4,36	4,05	A	2,21	4,89	A	10,2	7,85	B
2,71	9,38	C	7,68	9,67	C	3,05	1,7	A
6,75	5,67	C	4,02	11,87	C	9,43	9,87	B
2,07	4,63	A	13,85	13,38	C	3,04	14,09	C
8,4	2,5	B	5,7	10,3	C	3,19	13,11	C
4,27	10,45	C	13,73	2,59	B	12,06	13,98	C
0,6	8,6	C	11,94	11,23	C	8,83	0,14	B
2,39	7,33	C	14,61	6,25	B	0,3	13,42	C
1,37	8,98	C	0,15	9,41	C	12,03	1,55	B
4,27	8,48	C	13,93	12,67	C	0,85	9,39	C
10,87	12,52	C	9,31	2,71	B	11,25	11,42	C
5,61	11,67	C	12,21	7,06	B	0,44	6,66	C
6,32	12,1	C	9,72	14,5	C	7,21	1,19	B
6,83	5,31	C	3,62	13,86	C	4,77	14,96	C
5,66	6,03	C	6,62	12,92	C	2,17	10,47	C
14,12	2,71	B	6,41	5,94	C	1,38	2	A
4,84	13,88	C	11,57	11,59	C			

Tab. 4-2 Liste der zufällig gewählten Trainingsdaten (80 Samples)

Anhand der Trainingsdaten aus Tabelle 4-2 werden die verschiedenen Netze erstellt und anschließend durch die 40 Testsamples aus Tabelle 4-1 bewertet. Die folgende Tabelle 4-3 gibt einen Überblick über die einzelnen Netztypen und deren prozentuellen Fehler bezogen auf den Testdatensatz und auf die 80 Trainingssamples. Hierbei geht hervor, dass bei einem mehrschichtigen Feedforward-Netz mit zwei Layern zu je acht Neuronen oder einem Layer mit drei Neuronen der geringste Fehler im Testdatensatz von 5 %, sprich 2 falschen Testbeispielen, vorhanden ist.

Lfd. Nr.	Netztypen	% falsch bei 80 Trainingsdaten	% falsch bei 40 Testfällen
0	A	B	C
1	MLFN, 8 Knoten, 8 Knoten	0,00 %	5,00 %
2	MLFN, 3 Knoten	3,75 %	5,00 %
3	MLFN, 6 Knoten, 6 Knoten	0,00 %	7,50 %
4	MLFN, 8 Knoten	0,00 %	7,50 %
5	MLFN, 3 Knoten, 3 Knoten	0,00 %	10,00 %
6	MLFN, 5 Knoten	0,00 %	12,50 %
7	PNN	1,25 %	12,50 %
8	MLFN, 4 Knoten	0,00 %	15,00 %
9	MLFN, 6 Knoten	0,00 %	15,00 %
10	MLFN, 7 Knoten	0,00 %	15,00 %
11	MLFN, 2 Knoten	6,25 %	17,50 %

Tab. 4-3 Prozent falscher Zuordnung[1]

Der 5 %ige Fehler deutet auf ein gutes Modell hin. Hierbei ist jedoch zu beachten, dass dieser Fehlerwert nur die zufällig gewählten Testdaten beschreibt. Bei anderen Testdatensätzen kann dieser durchaus höher sein. Um herauszufinden, wie das neuronale Netz die Grenzen zwischen den Klassen interpretiert, wird ein Raster mit einem Abstand von 0,2 über die gesamte Fläche gelegt und jeder Punkt vom Netz prognostiziert. Dadurch ist es möglich, das aus den 80 Trainingsdaten erstellte Netz zu beurteilen. Abbildung 4-4 zeigt das Prognoseergebnis des zweischichtigen Netzes mit je acht Neuronen.

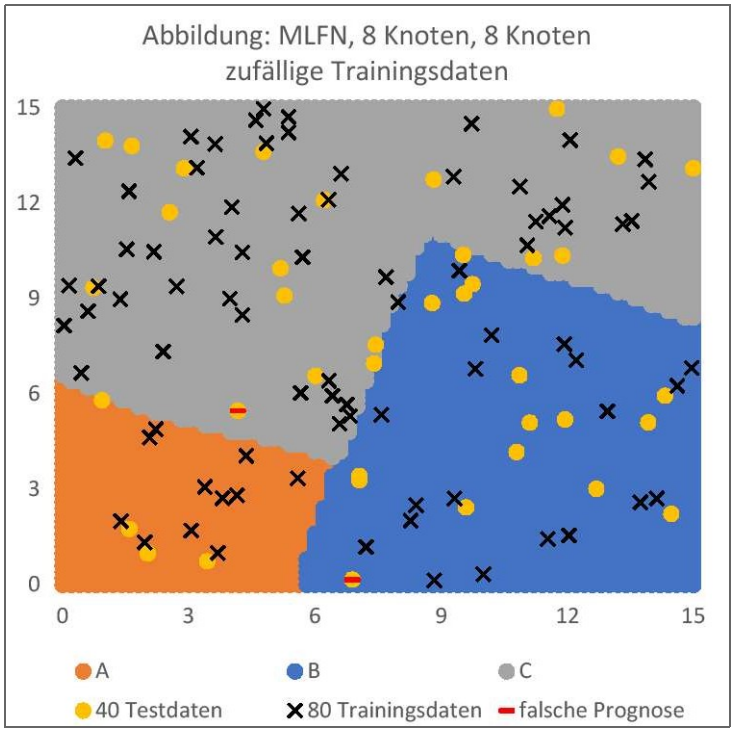


Abb. 4-4 Abbildung der Grenzen: MLFN, 8 Knoten, 8Knoten

[1] MLFN: mehrschichtiges Feedforward-Netz
PNN: Wahrscheinlichkeitsnetz

Hierbei wird ersichtlich, dass die Grenzen des trainierten Netzes einen anderen Verlauf als die aus Abbildung 4-1 aufweisen. Einen wesentlichen Grund für diese Abweichungen stellen die fehlenden Trainingsdaten in den Grenzbereichen dar. Die falschen Prognosen werden rot markiert.

Das neuronale Netz, welches eine Hidden-Schicht mit 3 Neuronen beinhaltet, wird ebenfalls mit Hilfe dieses Rasters analysiert. Die folgende Abbildung verdeutlicht, wie dieser Netztyp die Abgrenzung zwischen den Klassen A, B und C interpretiert. Im Gegensatz zu dem Grenzverlauf aus Abbildung 4-4 beginnt die Grenze zwischen Klasse A und C weiter oben und fällt stärker ab. Aus diesem Grund werden 3 Trainingssamples, welche rot markiert sind, nicht richtig beurteilt. Dies entspricht 3,75 % und wird in Tabelle 4-3 in Zeile B2 gezeigt. Im Testdatensatz werden die selben Punkte, wie beim vorherigen Netz nicht richtig erkannt. Diese werden ebenfalls mit rot gekennzeichnet.

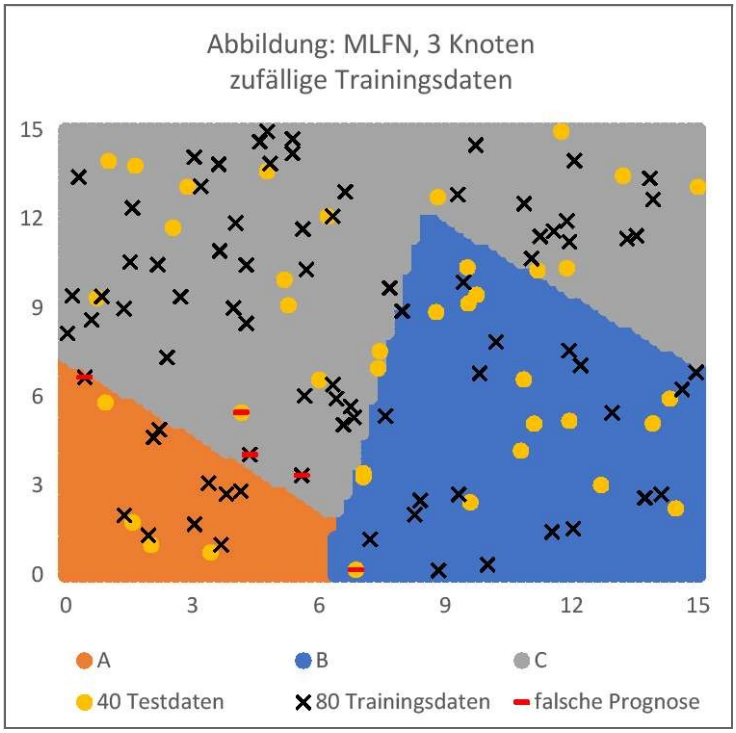


Abb. 4-5 Abbildung der Grenzen: MLFN, 3 Knoten

4.3 Ganzzahlig zufällig gewählte Trainingsdaten

In diesem Abschnitt werden anhand eines ganzzahligen, zufällig gewählten Datensatzes ein Wahrscheinlichkeitsnetz und mehrschichtige Feedforward-Netze mit einem Layer zu je 2 Knoten, 3 Knoten , 4 Knoten, 5 Knoten, 6 Knoten, 7 Knoten, 8 Knoten sowie mit zwei Layern aus 3 Knoten und 3 Knoten, 6 Knoten und 6 Knoten, 8 Knoten und 8 Knoten erstellt. Die Netze werden ebenfalls mit Hilfe des Testdatensatzes aus Tabelle 4-1 verglichen. Abschließend werden die besten Netze veranschaulicht.

Die folgende Abbildung 4-6 zeigt die ganzzahligen, zufällig gewählten Punkte aus dem Trainingsdatensatz. Weiters werden diese Trainingsdaten in den Tabellen 4-4 und 4-5 aufgelistet.

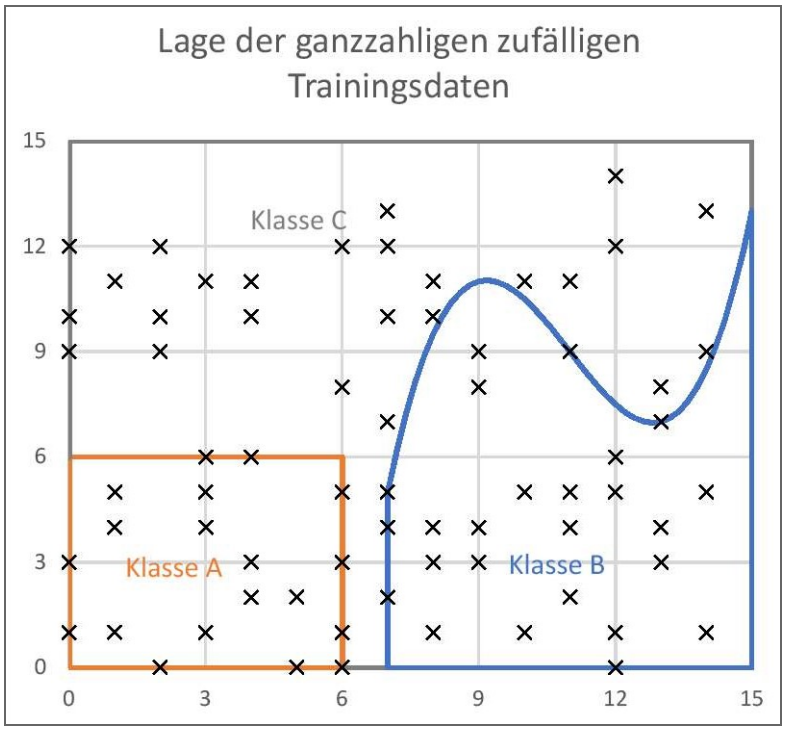


Abb. 4-6 Darstellung der 80 ganzzahligen, zufälligen Trainingsdaten

X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse
3	5	A	14	5	B	10	1	B
11	11	C	8	1	B	2	0	A
6	1	A	0	1	A	12	1	B
8	4	B	7	13	C	13	3	B
1	5	A	3	4	A	6	0	A
11	4	B	7	4	B	8	4	B
4	3	A	1	11	C	8	3	B
5	2	A	3	1	A	12	12	C
10	5	B	14	9	C	7	7	C
9	4	B	14	1	B	4	2	A
7	10	C	6	12	C	0	10	C
7	10	C	9	9	B	7	12	C
14	5	B	2	12	C	6	8	C
8	10	C	4	2	A	2	10	C
12	5	B	4	11	C	11	9	B
9	3	B	9	3	B	10	5	B

Tab. 4-4 Liste der ganzzahligen, zufällig gewählten Trainingsdaten – Teil 1

6	1	A	13	7	B	12	0	B
0	12	C	12	14	C	0	9	C
8	11	C	3	6	A	6	5	A
10	11	C	14	13	C	3	11	C
6	3	A	13	8	C	7	2	B
4	10	C	1	1	A	12	12	C
9	8	B	0	3	A	9	9	B
11	2	B	13	4	B	2	9	C
0	9	C	14	13	C	7	5	B
11	5	B	12	6	B	5	0	A
1	4	A	4	6	A			

Tab. 4-5 Liste der ganzzahligen, zufällig gewählten Trainingsdaten – Teil 2

Die Auswertung der verschiedenen Netze, welche aus den 80, ganzzahligen, zufällig gewählten Trainingsdaten erstellt worden ist, wird in Tabelle 4-6 gezeigt.

Lfd. Nr.	Netztypen	% falsch bei 80 Trainingsdaten	% falsch bei 40 Testfällen
0	A	B	C
1	MLFN, 3 Knoten, 3 Knoten	0,00 %	5,00 %
2	MLFN, 6 Knoten, 6 Knoten	0,00 %	5,00 %
3	MLFN, 8 Knoten, 8 Knoten	0,00 %	5,00 %
4	PNN	0,00 %	7,50 %
5	MLFN, 4 Knoten	1,25 %	12,50 %
6	MLFN, 7 Knoten	0,00 %	12,50 %
7	MLFN, 8 Knoten	0,00 %	12,50 %
8	MLFN, 5 Knoten	0,00 %	17,50 %
9	MLFN, 2 Knoten	7,50 %	20,00 %
10	MLFN, 6 Knoten	0,00 %	20,00 %
11	MLFN, 3 Knoten	3,75 %	22,50 %

Tab. 4-6 Prozent falscher Zuordnung

Es wird ersichtlich, dass die Netze, welche zwei versteckte Schichten aufweisen, bessere Ergebnisse erzielen, als die einschichtigen.

Die nächsten Abbildungen 4-7, 4-8 und 4-9 verdeutlichen, wie die drei besten Netze, welche jeweils einen Fehler von 5 % vorweisen, die Grenzlegung interpretieren. Die Netze, welche aus zwei Hidden-Layern mit je drei Neuronen und mit je sechs Neuronen bestehen, unterscheiden sich minimal. Auffallend ist, dass das Netz, welches zwei Schichten je acht Neuronen aufweist, die Klasse A sehr gut abbildet.

Der Punkt (6,88/0,16) wird bei beiden bisherigen Datensätzen nicht richtig identifiziert. Dies ist darauf zurückzuführen, dass keine Trainingsdaten der Klasse C in diesem Bereich vorhanden sind.

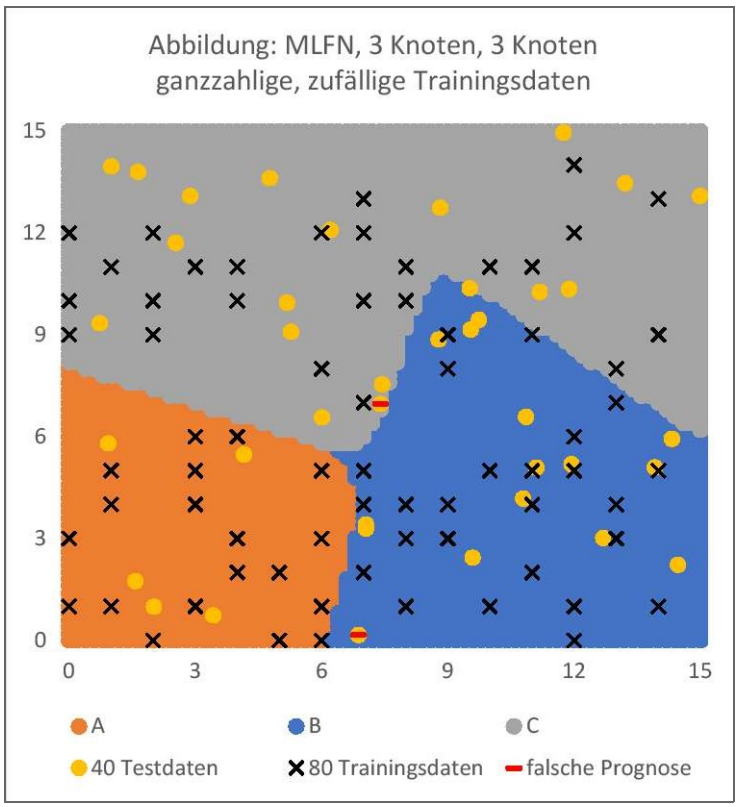


Abb. 4-7 Abbildung der Grenzen: MLFN, 3 Knoten, 3 Knoten

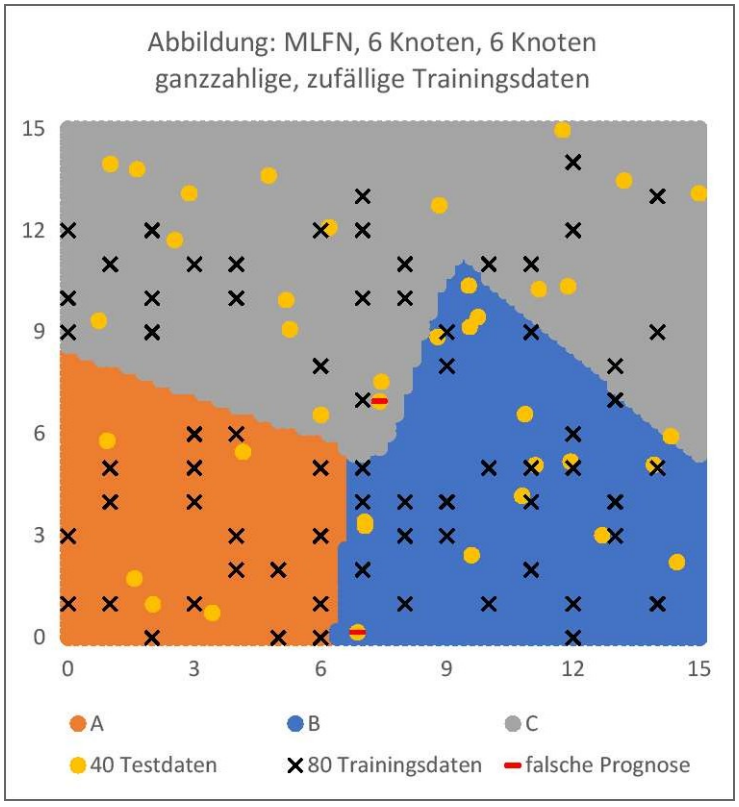


Abb. 4-8 Abbildung der Grenzen: MLFN, 6 Knoten, 6 Knoten

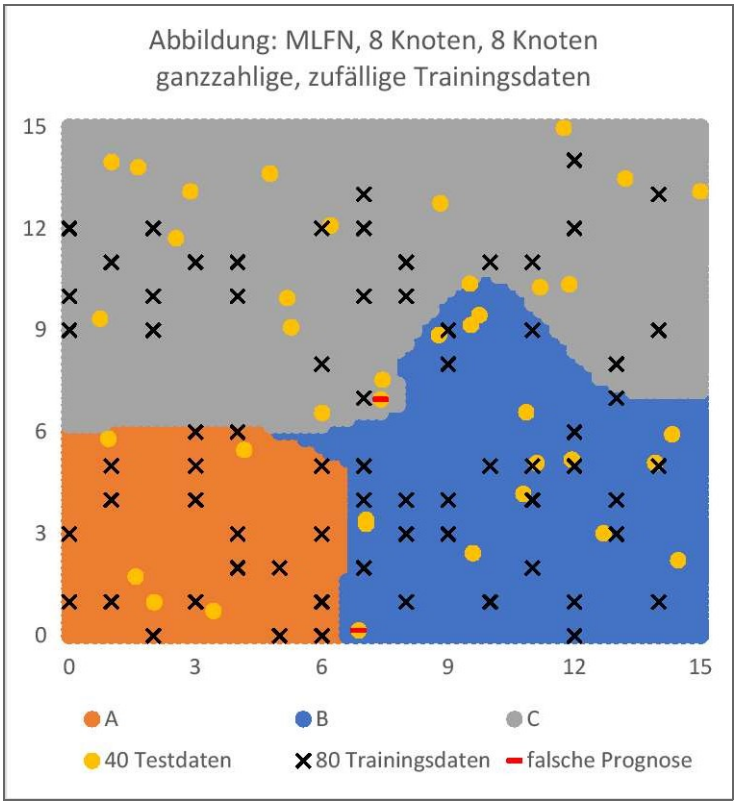


Abb. 4-9 Abbildung der Grenzen: MLFN, 8 Knoten, 8 Knoten

4.4 Im Raster angeordnete Trainingsdaten

Der dritte Datensatz besteht aus im Raster angeordneten Trainingsdaten, welche einen Abstand von 1,875 aufweisen. Um eine gleiche Anzahl an Trainingsbeispielen zu erhalten wird der Punkt (0/15) aus dem Raster herausgenommen, weil dieser einen äußerst geringen Einfluss auf die Klasse C hat.

Dieser Datensatz weist aus den drei zu untersuchenden Trainingsdaten die größte Ordnung auf. Jedoch hat er ebenfalls keine Punkte für die Klasse C zwischen 6 und 7.

In der nachfolgenden Abbildung 4-10 werden die Klassen A, B und C sowie die 80 im Raster angeordneten Trainingsdaten gezeigt. Weiters werden diese Punkte in den Tabellen 4-7 und 4-8 dargestellt.

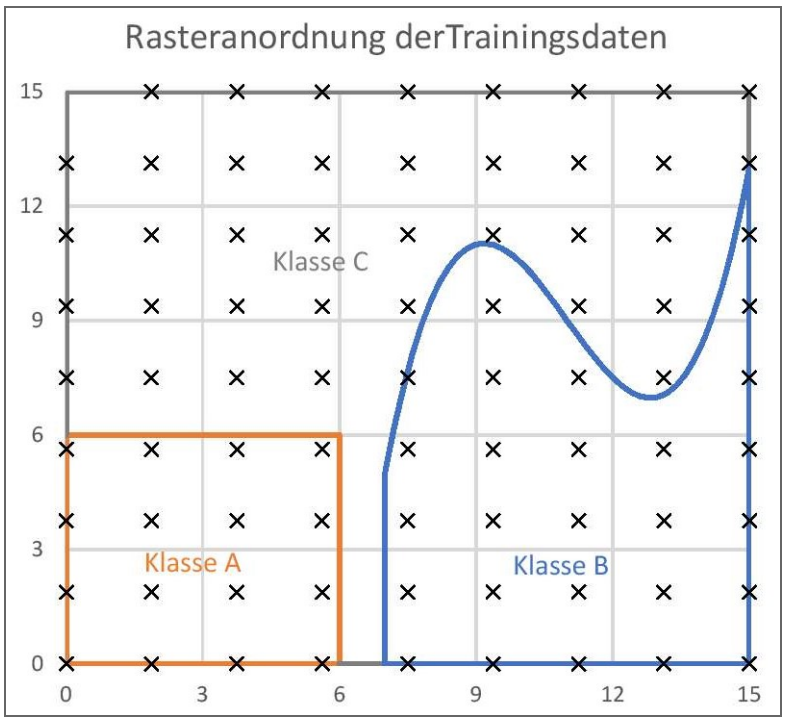


Abb. 4-10 Darstellung der 80 im Raster angeordneten Trainingsdaten

X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse	X-Werte	Y-Werte	Klasse
0	0	A	3,75	7,5	C	7,5	13,125	C
0	1,875	A	3,75	9,375	C	7,5	15	C
0	3,75	A	3,75	11,25	C	9,375	0	B
0	5,625	A	3,75	13,125	C	9,375	1,875	B
0	7,5	C	3,75	15	C	9,375	3,75	B
0	9,375	C	5,625	0	A	9,375	5,625	B
0	11,25	C	5,625	1,875	A	9,375	7,5	B
0	13,125	C	5,625	3,75	A	9,375	9,375	B
1,875	0	A	5,625	5,625	A	9,375	11,25	C
1,875	1,875	A	5,625	7,5	C	9,375	13,125	C
1,875	3,75	A	5,625	9,375	C	9,375	15	C
1,875	5,625	A	5,625	11,25	C	11,25	0	B
1,875	7,5	C	5,625	13,125	C	11,25	1,875	B
1,875	9,375	C	5,625	15	C	11,25	3,75	B
1,875	11,25	C	7,5	0	B	11,25	5,625	B
1,875	13,125	C	7,5	1,875	B	11,25	7,5	B
1,875	15	C	7,5	3,75	B	11,25	9,375	C
3,75	0	A	7,5	5,625	B	11,25	11,25	C
3,75	1,875	A	7,5	7,5	B	11,25	13,125	C
3,75	3,75	A	7,5	9,375	C	11,25	15	C
3,75	5,625	A	7,5	11,25	C	13,125	0	B

Tab. 4-7 Liste der im Raster angeordneten Trainingsdaten – Teil 1

13,125	1,875	B	13,125	13,125	C	15	7,5	B
13,125	3,75	B	13,125	15	C	15	9,375	B
13,125	5,625	B	15	0	B	15	11,25	B
13,125	7,5	C	15	1,875	B	15	13,125	C
13,125	9,375	C	15	3,75	B	15	15	C
13,125	11,25	C	15	5,625	B			

Tab. 4-8 Liste der im Raster angeordneten Trainingsdaten – Teil 2

Die nach den prozentuell falschen Prognosen gereihten Netztypen werden in Tabelle 4-9 dargestellt. Hieraus geht hervor, dass das Wahrscheinlichkeitsnetz und das Feedforward-Netz, welches zwei versteckte Layer zu je acht Neuronen (MLFN, 8 Knoten und 8 Knoten) hat, mit einem Fehler von 7,5 % bei der Klassifizierung der 40 Testfällen am besten abschneiden (siehe Tabelle 4-9 Zeile C1 und Zeile C2).

Lfd. Nr.	Netztypen	% falsch bei 80 Trainingsdaten	% falsch bei 40 Testfällen
0	A	B	C
1	PNN	0,00 %	7,50 %
2	MLFN, 8 Knoten, 8 Knoten	0,00 %	7,50 %
3	MLFN, 6 Knoten, 6 Knoten	0,00 %	10,00 %
4	MLFN, 3 Knoten, 3 Knoten	0,00 %	10,00 %
5	MLFN, 4 Knoten	2,50 %	15,00 %
6	MLFN, 5 Knoten	2,50 %	15,00 %
7	MLFN, 6 Knoten	1,25 %	15,00 %
8	MLFN, 8 Knoten	8,00 %	17,50 %
9	MLFN, 2 Knoten	8,75 %	20,00 %
10	MLFN, 7 Knoten	0,00 %	22,50 %
11	MLFN, 3 Knoten	7,50 %	27,50 %

Tab. 4-9 Prozent falscher Zuordnung

Die nachfolgenden Abbildungen 4-11 und 4-12 verdeutlichen die Grenzführung zwischen den einzelnen Klassen bei rasterförmig angeordneten Trainingsdaten. In diesem Datensatz wird die Funktion der Klasse B am besten nachgebildet. Hierbei ist jedoch zu erwähnen, dass in den anderen Datensätzen in diesem Bereich keine ausreichende Datengrundlage vorhanden ist.

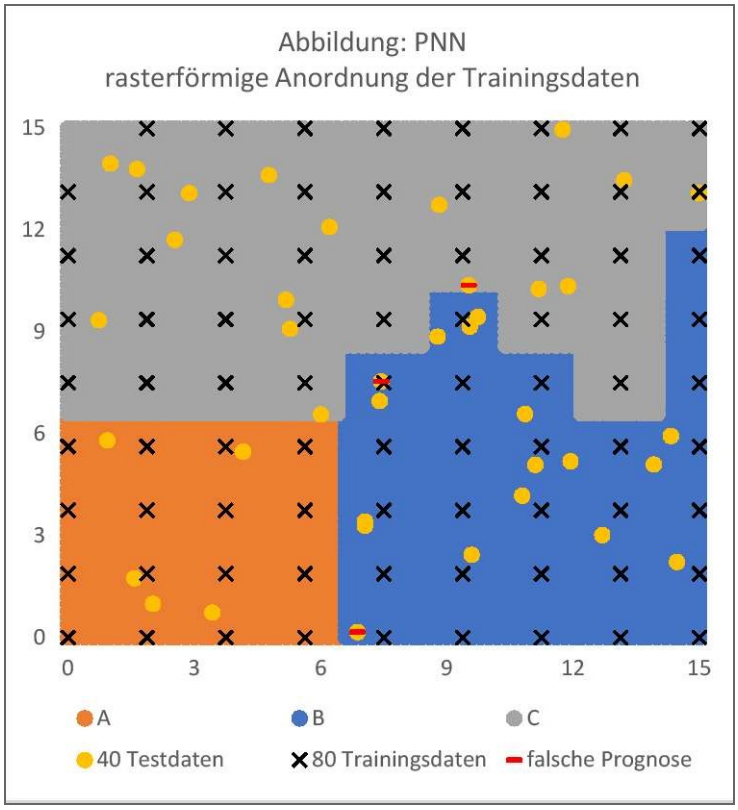


Abb. 4-11 Abbildung der Grenzen: PNN

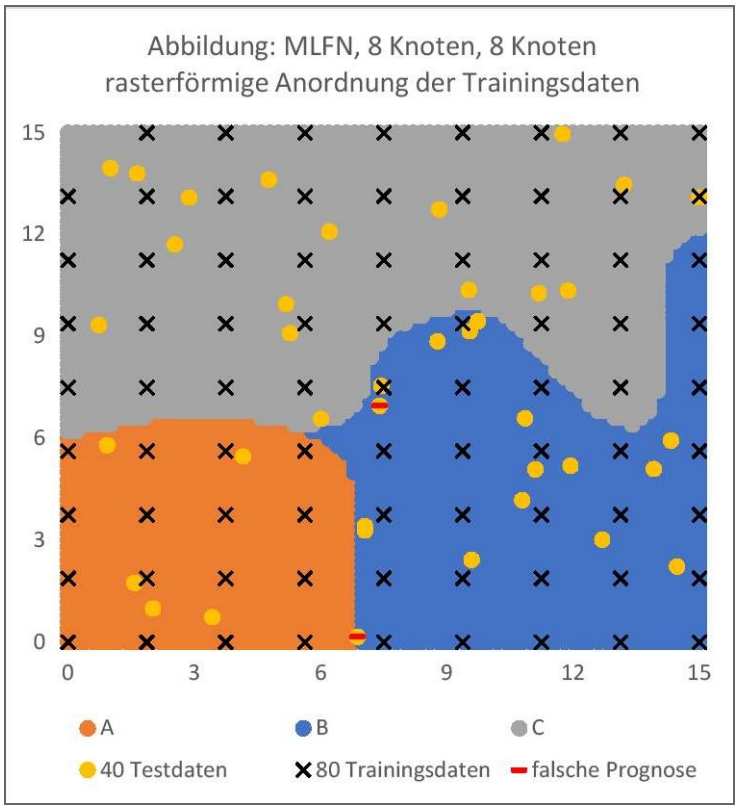


Abb. 4-12 Abbildung der Grenzen: MLFN, 8 Knoten, 8 Knoten

4.5 Vergleich der Testergebnisse

Bei einem neuen Datensatz kann von vorne herein nicht festgestellt werden, welcher Netztyp zum besten Ergebnis führt. Aus diesem Grund gilt es mehrere Netzkonfigurationen zu testen. Hierbei gilt es jedoch immer zu beachten, dass bei Vergleichsdatensätzen nur die einzelnen Punkte bewertet werden. Wie das neuronale Netz zwischen diesen Punkten klassifiziert, kann somit nicht beurteilt werden.

Aufgrund einer fehlenden Datengrundlage des Bereiches C zwischen 6 und 7 (Abszisse) wird der Testpunkt (6,88/0,16) in jedem Netz falsch vorhergesagt. Weiters gelingt es keinem Netz unter 5 % falscher Prognosen, sprich zwei nicht korrekt vorhergesagten Testfällen, zu gelangen. Obwohl im Rasterdatensatz von den erstplatzierten Netzen nur ein minimaler Fehler von 7,5 % falscher Klassifizierungen erreicht werden kann, bilden diese neuronalen Netze die Grenze besser nach, als die Netze mit 5 %igem Fehleranteil.

Aus den Datensätzen geht hervor, dass strukturiertere Trainingsdaten zu besseren Ergebnissen führen. Dies ist nachvollziehbar, weil es durch eine gleichmäßige Verteilung der bekannten Punkte zu keinen leeren Bereichen kommt. Aus diesem Grund sollten Messdaten möglichst gleichverteilt erhoben werden.

Weiters ist der Trend zu erkennen, dass bei steigender Anzahl von versteckten Schichten und von Neuronen bessere Grenzverläufe erzielt werden. Um diese Beobachtung zu überprüfen, wird ein weiteres Netz mit dem zufällig gewählten Datensatz aus Abbildung 4-3 trainiert. In diesem Fall weist das Netz zwei versteckte Schichten mit jeweils 20 Neuronen auf. Die interpretierten Grenzen werden in Abbildung 4-13 verdeutlicht. Es ist auffallend, dass trotz vier falscher Prognosen aus insgesamt 40 Testdaten die Grenzverläufe besser als bei den übrigen Netzen des gleichen Datensatzes ausgeführt sind.

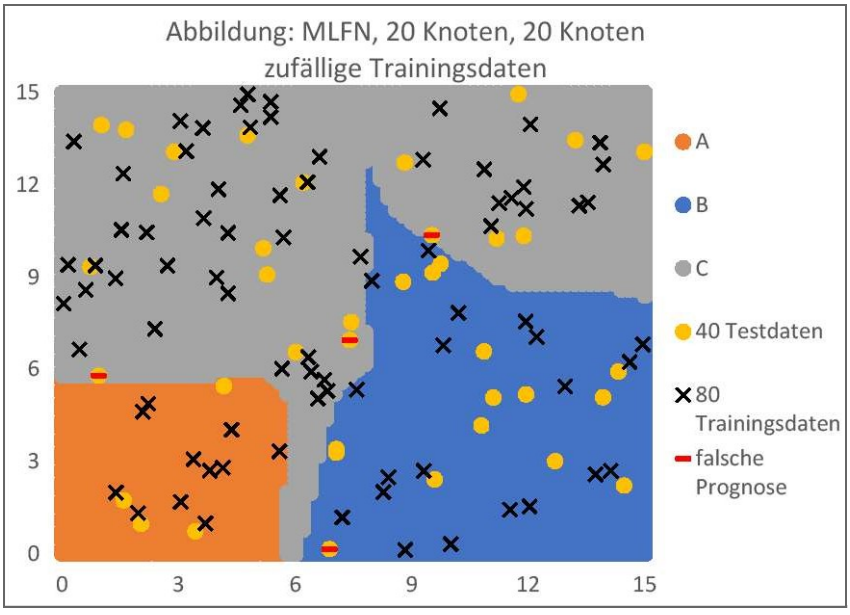


Abb. 4-13 Abbildung der Grenzen: MLFN, 20 Knoten, 20 Knoten

5 Zusammenfassung und Ausblick

Durch die stark angestiegene Rechenleistung der Computer sind neuronale Netze zu einem geeigneten Werkzeug für das Lösen von technischen Aufgaben aus dem Bereich der Klassifizierung und der Prognose geworden. Um Problemstellungen bestmöglich zu beantworten ist es wichtig, den Aufbau und die Anwendungsbereiche der einzelnen Netztypen zu kennen. Hierbei empfiehlt es sich immer in Fachliteratur nach ähnlichen Problemstellungen zu suchen und das beschriebene Netz auf die eigene Aufgabe anzupassen.

Ein neuronales Netz folgt der Grundidee der Funktionsweise des menschlichen Gehirns. Hierfür werden Neuronen, die auch Units oder Einheiten genannt werden, untereinander mit Verbindungen versehen und leiten, nach Überwindung der Aktivierungsenergie, Impulse an andere Einheiten weiter. Die einzelnen Neuronen werden durch mathematische Funktionen dargestellt und haben die Aufgabe abhängig von den zugesendeten Inputs einen Output zu generieren. Die Verbindungen zwischen den Units werden durch Gewichte repräsentiert. Je stärker der Einfluss eines Neurons auf ein anderes ist, desto größer wird die Gewichtung dieser Verbindung. An dieser Stelle ist es wichtig zu erwähnen, dass hinter dem Konzept neuronaler Netze nur mathematische Zusammenhänge stecken. Die schematische Darstellung aus Neuronen und Verbindungen ist lediglich eine Abbildung, die zu einem besseren Verständnis beiträgt.

Wie auch beim biologischen Vorbild, werden die Einheiten abhängig von deren Aufgabe in Input-, Hidden- und Output-Neuronen gegliedert. Input-Units nehmen die Daten aus der Umwelt auf und übergeben sie an Hidden-Units. Dort wird der Input modifiziert und entweder an andere Neuronen, der gleichen Art, oder an Output-Units weitergeleitet. Output-Einheiten haben die Aufgabe zugesendete Informationen an die Umwelt wieder abzugeben. Um eine bessere Struktur zu erhalten werden die Neuronen zusätzlich in Schichten gegliedert. Diese Schichten, die auch Layer genannt werden, fassen Neuronen gleicher Art zusammen und werden hintereinander angeordnet. Somit besteht ein Netz aus mindestens einem Input-Layer und einem Output-Layer. Dazwischen können sich ein oder mehrere Hidden-Layer befinden.

Das Vermächtnis des Lernens, beruht auf der Adaptionsfähigkeit der Netze. Zunächst werden Trainingssätze, welche aus Inputs mit zugehörigen Outputs bestehen, benötigt, um den Status des Netzes zu evaluieren. Hierfür greift ein Trainingsalgorithmus auf diese Daten zu und versucht ein neuronales Netz zu konstruieren, welches die Vorgaben bestmöglich abbildet. Weil im Trainingsprozess verschiedene Inputs mit zugehörigem Output vorgegeben werden, fällt diese Art des Lernens in die Kategorie „supervised learning“ (überwachtes Lernen). Um ein Netz an den Trainingsdatensatz anzupassen, können Verbindungen aufgebaut oder gelöscht, die einzelnen Gewichtungen verstärkt oder abgeschwächt, Neuronen hinzugefügt oder entfernt werden. Weiters besteht die Möglichkeit die Funktionen der Neuronen zu modifizieren. Um herauszufinden wie ein neuronales Netz anzupassen ist, werten Trainingsalgorithmen die Fehlerfunktion zwischen berechnetem Output und Soll-Output aus. Anschließend wird bei dieser Funktion das Fehlerminimum gesucht.

Aufgrund der Tatsachen, dass Neuronen auf unterschiedliche Arten verbunden und strukturiert werden können, existieren verschiedene Netztypen, die jeweils eigene Lernalgorithmen besitzen. In dieser Masterarbeit werden mehrschichtige Feedfoward-Netze, Wahrscheinlichkeitsnetze und Regressionsnetze diskutiert, weil diese der Software NeuralTools 7.5 zu Grunde liegen. Abhängig von der Aufgabenstellung werden die zuvor erwähnten Netze unterschiedlich eingesetzt.

Neuronale Netze können bei Aufgabenstellungen aus dem Bereich der Klassifizierung und bei numerischen Prognosen eingesetzt werden. Unter dem ersteren wird das Zuteilen eines gewissen Inputs in eine Klasse verstanden. Hingegen wird bei numerischen Prognosen für einen gewählten Input ein dazugehöriger Wert berechnet. Um ein neuronales Netz erfolgreich als Entscheidungsunterstützung einsetzen zu können, müssen jedoch genügend Trainingsdaten vorhanden sein. Grundsätzlich gilt, je mehr Trainingsdaten vorhanden sind, desto besser wird das Netz. Eine Mindestanzahl kann nicht festgelegt werden, weil diese bei jedem Netztyp unterschiedlich ist. Sie hängt unter anderem von den maximal erlaubten Fehlerwerten, der Komplexität des Modelles und der Aufgabenstellung ab.

Im Programm NeuralTools 7.5 kann eine Zuordnung in eine bestimmte Klasse mittels zwei verschiedener Netztypen durchgeführt werden. Entweder wird ein mehrschichtiges Feedforward-Netz oder ein Wahrscheinlichkeitsnetz eingesetzt. Bei numerischen Prognosen kann ebenfalls das mehrschichtige Feedforward-Netz oder ein Regressionsnetz angewendet werden. Von vorn herein ist es jedoch nicht möglich festzustellen, welcher Netztyp die geringeren Fehlerwerte aufweist. Aus diesem Grund besitzt diese Software den Netztyp „bestes Netz suchen“. Hierbei werden beide Netztypen getestet und anschließend das mit den geringeren Abweichungen ausgewählt.

Sowohl im konstruktiven Ingenieurbau als auch im Baubetrieb und der Bauwirtschaft können neuronale Netze eingesetzt werden. Sobald es sich um eine Problemstellung aus dem Bereich der Klassifizierung oder um eine numerische Prognose handelt und genügend Trainingsdaten vorhanden sind, eignen sich diese Netze hervorragend. Im Ingenieursbau werden sie beispielsweise bereits bei der Stahlträgermodellierung, in der Bruchmechanik zur Parameteridentifikation, bei Sicherheitsfragen bezüglich der Einsturzgefahr nach Erdbeben und zur Bewertung bestehender Betonbrücken eingesetzt. Hierbei gilt es jedoch zu beachten, dass die Ergebnisse lediglich Richtwerte darstellen, welche sich für Abschätzungen sehr gut eignen. Die notwendigen Nachweise müssen immer nach Norm geführt werden.

In der Bauwirtschaft eignen sich Aufgabenstellungen aus dem Bereich der Terminplanung, der Projektdauer, der Kostenschätzung, der Produktivität, oder der Aufwandswerte bzw. der Leistungswerte gut für die Verwendung von neuronalen Netzen. Die Erfahrungen aus vergangenen Projekten können in einer derartigen Form gespeichert werden, dass es möglich ist bei neuen Gegebenheiten Lösungsmöglichkeiten aufgrund dieses gesammelten Wissens zu finden. An dieser Stelle ist es wichtig anzumerken, dass die Lösungen nie besser als die einzelnen Outputs aus dem gegebenen Datenmaterial werden können. Eine Optimierungsaufgabe kann somit durch supervised learning nicht gelöst werden. Weiters eignen sich neuronale Netze ausgezeichnet in der Kalkulation. Rasch können für ein neues Projekt Aufwandswerte und Verfahren, auf Basis bereits abgeschlossener

Baustellen, ausgewählt werden.

Ein weiteres Einsatzgebiet stellt die Bilderkennung dar. Neuronale Netze besitzen die Fähigkeit in Bildern Informationen zu erkennen. Dies ermöglicht beispielsweise eine automatische Baustellen-Überwachung, weil die aufgestellte Schalung von der fertigen Betonwand unterschieden werden kann. Zusätzlich könnte völllig automatisch die Helmpflicht kontrolliert werden.

Aufgrund der Einzigartigkeit eines Bauwerks und unterschiedlichster Baustellenbedingungen stellt die Datenakquisition ein nicht triviales Unterfangen dar. Ist es möglich auf ausreichend Daten zurückzugreifen, welche die gesamte Bandbreite, in der sich die Lösungen befinden können, abdeckt, stellen neuronale Netze ein ausgezeichnetes Werkzeug für die Problemlösung dar.

Die vorgeschlagenen Anwendungsmöglichkeiten dienen als Beispiele und haben die Aufgabe den großen Einsatzbereich zu verdeutlichen. In den nächsten Jahren gilt es durch weitere Forschungsarbeiten die einzelnen Anwendungsbereiche zu analysieren, um die notwendigen Input-Parameter festzulegen. Nur durch weitere Forschungsarbeiten und der Bereitschaft vom traditionellen Weg abzuweichen, können die versteckten Potenziale der neuronalen Netze, als ein geeignetes Hilfsmittel zur Lösungs- und Entscheidungsfindung, aufgedeckt werden.

6 Literaturverzeichnis

Das Literaturverzeichnis ist in folgende Abschnitte unterteilt:

- Zeitschriften- und Fachartikel
- Bücher/Buchkapitel
- Masterarbeiten/Diplomarbeiten und Dissertationen
- Anwenderinformationen
- Seminar
- Internetquellen

6.1 Fachartikel

Adeli,H.: Neural Networks in Civil Engineerig:1989-200, In: Computer-Aided Civil and Infrastructure Engineering, Vol. 16, No. 2, page 126-142, March 2001. (Online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.872&rep=rep1&type=pdf, Datum des Zugriffs 21.05.2018)

Adeli, H.; Panakkat, A.: A probabilistic neural network for earthquake magnitude prediction, In: Neural Networks, Vol. 22, No. 7, page 1018-1024, September 2009.

Adeli, H.: Yeh, C.: Perceptron learning in engineering design, In: Microcomputers in Civil Engineering, Vol. 4, No.4, page 247-256, Dezember 1989.

Chen, C.;Lee, J.; Lin, M.: Classification of underwater signals using neural networks, In: Tamkang Journal of Science and Engineering, Vol. 3, No. 1, page 31-48, June 2000. (Online: http://www2.tku.edu.tw/~tkjse/3-1/3-1-4.pdf, Datum des Zugriffs 13.05.2018)

Hofstadler, C.; Schütz, M.: Anwendung des Systems Engineering auf die Arbeitsvorbereitung von Bauprojekten. In: Bautechnik, 11/2012.

Huang, C.;Liao, W.: Application of probabilistic neural networks to the class prediction of leukemia and embryonal tumor of the central nervous system, In Neural Processing Letters, Vol. 19, No. 3, page 211-226, June 2004. (ISSN: 1370-4621)

Johansson, E. M.; Dowla F. U.; Goodman D. M.: Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method, In: International Journal of Neural Systems Vol. 2, Nr. 4, page 291-301, 1992.

Lazarevska M. et al.: Application of artificial neural networks in civil engineering, In: Tehnicki vjesnik, Vol.21, No.6, page 1353-1359, 2014. (ISSN 1330-3651)

Kim, D.; Lee, J.; Chang, S.: Application of probability neural networks for prediction of concrete strength, In: Journal of Materials in Civil Engineering , ASCE, Vol. 17, No. 3, page 353-362; June 2005.(ISSN: 0899-1561)

Kiran S.; Lal B.; Tripathy S. S.: Shear Strength Prediction of Soil based on Probabilistic Neural Network, In: Indian Journal of Science and Technology, Vol. 9, No. 41, page 1-6, November 2016.(ISSN: 0974-6846, Online: http://www.indjst.org/index.php/indjst/article/viewFile/99188/74701, Datum des Zugriffs 13.05.2018)

Kulkarni, P., Londhe, S., Deo, M.: Artificial Neural Networks for Construction Management: A Review, In: Soft Computing in Civil Engineering, Vol. 1. No. 2, page 70-88, August 2017 . (Online: http://jsoftcivil.com/article_49580_de93dbb5e88d37842a26b0c1c53f17c5.pdf, Datum des Zugriffs 21.05.2018)

Parzen, E.: On estimation of a probability density function and mode, In: Annals of Mathematical Statistics , Vol. 33, No. 3, page 1065-1076, 1962 (Online: https://projecteuclid.org/download/pdf_1/euclid.aoms/1177704472, Datum des Zugriffs: 14.05.2018)

Specht, Donald F.: Probabilistic Neural Networks, In: Neural Networks, Vol. 3, page 109-118, 1990. (ISSN 0893-6080; Online: https://www.wi.hs-wismar.de/~cleve/vorl/projects/dm/ss13/PNN/Quellen/Specht_Probabilistic-NeuralNetworks.pdf, Datum des Zugriffs: 10.05.2018)

Specht, Donald F.: Generation of Polynomial Discriminant Functions for Pattern Recognition, In: IEEE Transactions on electronic computers, Vol. Ec.16, No. 3, page 308-319, June 1967. (ISSN: 0367-7508)

Specht, Donald F.: A General Regression Neural Network, In: IEEE Transactions on neural networks, VOl. 2, No. 6, page 568-576, November 1991. (Online: https://www.researchgate.net/publication/5569076_A_general_regression_neural_network, Datum des Zugriffs: 24.05.2018)

Zeinali,Yasha; Story, Brett A.: Competitive probabilistic neural network, In: Integrated Computer Aided Engineering, page 1-14, January 2017. (ISSN 1069-2509)

6.2 Bücher

Bishop, C. M.: Neural Networks for Pattern Recognition, Clarendon Press, oxford,1995. (Online: http://cs.du.edu/~mitchell/mario_books/Neural_Networks_for_Pattern_Recognition_-_Christopher_Bishop.pdf, Datum des Zugriffs: 03.06.2018)

Hofstadler, C.; Kummer, M.: Chancen- und Risikomanagement in der Bauwirtschaft, Graz, Springer-Verlag, 2017. (ISBN 978-3-662-54318-4)

Rich, E.; Knight, K.; Nair S.B.:Artificial Intelligence, 3. Auflage, Tata McGraw-Hill, 2009 (ISBN 978-0-07-008770-5)

Ertel, W.: Grundkurs Künstlicher Intelligenz, 3. Auflage, Springer Vieweg, 2013 (ISBN 978-3-8348-1677-1)

Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren, 1. Auflage, Vieweg+Teubner Verlag, 2010, (ISBN 978-3-8348-0822-6)

6.3 Masterarbeiten/Diplomarbeiten

Winzenborg, Insa.: Bayes'sche Schatztheorie und ihre Anwendung auf neuronale Daten zur Reizrekonstruktion, Diplomarbeit, Carl-von-Ossietzky-Universit¨at Oldenburg, Juni 2007. (Online: https://www.statistik.tu-dortmund.de/fileadmin/user_upload/Lehrstuehle/Ingenieur/Mueller/Diplomarbeiten/Winzenborg.pdf, Datum des Zugriffs: 10.05.2018)

6.4 Anwenderinformationen

Palisade Corporation: Benutzerhandbuch, NeuralTools, Neuroralnetz-Add-In für Microsoft Excel, Version 7 Juni, 2015.

6.5 Seminar

Hofstadler, C.: Baubetrieb Forschungsseminar: Systems Engineering Einführung, WS 2017

6.6 Internetquellen

https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm, Datum des Zugriffs: 07.05.2018

http://www.neuronalesnetz.de/. Datum des Zugriffs: 17.05.2018

https://de.statista.com/statistik/lexikon/definition/112/regression/, Datum des Zugriffs: 24.05.2018

https://www.uni-ulm.de/fileadmin/websi-te_uni_ulm/mawi.inst.110/lehre/ss13/Stochastik_I/Skript_4.pdf. Datum des Zugriffs: 03.05.2018

https://de.mathworks.com/products/matlab.html, Datum des Zugriffs: 27.05.2018

https://products.office.com/de-at/excel, Datum des Zugriffs: 29.05.2018

http://www.palisade.com/neuraltools/de/, Datum des Zugriffs: 29.05.2018

https://de.statista.com/statistik/lexikon/definition/126/standardabweichung/, Datum des Zugriffs: 31.05.2018

Abkürzungsverzeichnis

Im folgenden Abschnitt sind die wesentlichsten Abkürzungen dargestellt.

G

GRNN	General Regression Neural Network

M

MLFN	mehrschichtiges Feedforward-Netz

P

PNN	Probability Neural Network

W

W_i	Vektor aus Gewichtungen
W_{ij}	Gewichtsmatrix

Sonderzeichen

€	Euro
∫	Integral
%	Prozent
∞	Unendlich

Abbildungsverzeichnis

Tabellenverzeichnis